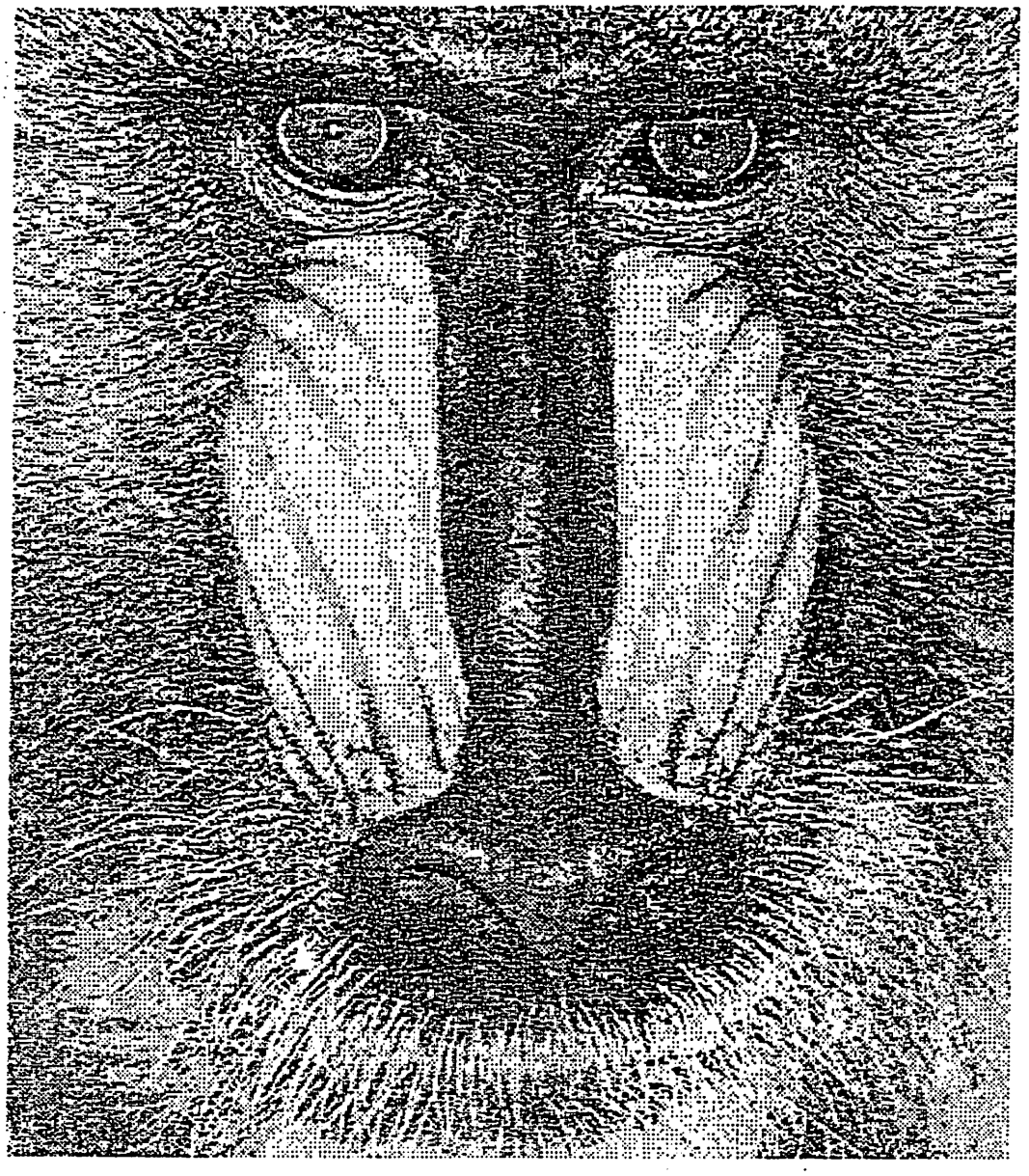
Guidance of library program use

(Numerical calculation: NUMPAC VOL. 2)

6. Interpolation, smoothing, and numerical differentiation and integration

7. Fourier analysis 8. Numerical integration

9. Ordinary differential equation 10. Elementary function



30/9/92

Nagoya University Computer Center

(Supervised by: Ichizo Ninomiya)

I. NUMPAC routine

Library programs of NUMPAC are roughly divided into two cathegories, ie., function subprograms and subroutine subprograms. There are some general rules for each of them and the rules are used in this manual for simple description. Please read the following explanations carefully before using NUMPAC.

(I) Function subprogram

(1) Function name and type

The function name of the real type follows the rule of the implicit type specification of FORTRAN.

Example : BJ0, ACND

The function name of the double precision real type consists of the function name of the corresponding real type with adding D to the head of it. The function name of the quadruple precision real number type (if exists) consists of the function name of the corresponding real type with adding Q to the head of it. However, there are some exceptions.

Example : SINHP, DSINHP, QSINHP

Example of exception : ALOG1, DLOG1, QLOG1

It is severely observed that the function name for double precision begins with D and that for quadruple precision begins with Q. Note that the function name should be declared with a suitable type in each program unit referring to the function.

```
Example : DOUBLE PRECISION DCOSHP, DJ1
          REAL*8 DCELI1,DCELI2
          REAL*16 QSINHP,QASINH
```

Because the function name of double precision always begins with D and that of quadruple precision with Q, it is convenient to use the IMPLICIT statement considering other variables.

```
Example : IMPLICIT REAL*8(D)
          IMPLICIT REAL*8(A-H,O-Z)
```

In this way, you need not declare the function name, separately.

(2) Accuracy of function value

Function routines are created aiming at the accuracy of full working precision as a rule. However, this cannot be achieved completely because of fundamental or technical difficulty [1].

2

Especially, it is not achieved for functions of two variables and functions of complex variable.

(3) Limit of argument

(a) The domain is limited.

Example : ALOG1

This function calculates $log(1+x)$. Therefore, $x>-1$ should be satisfied.

(b) The singular point exists.

Example : TANHP

This function calculates $\tan \pi x/2$. Therefore, an odd integer x is a sungularity.

(c) The function value overflows.

Example : BI0

This function is for modified Bessel function $I_0(x)$, and for big x, e^x is calculated referring to standard function EXP. Therefore, overflow limit $252log_e2 \fallingdotseq 174.673$ of EXP is the upper bound of the argument of this function.

(d) The function value becomes meaningless.

Example : BJ0

This function is for Bessel function $J_0(x)$, and standard functions SIN and COS are referred to for big x. Therefore, the argument limit $|x| \leqq 2^{18}\pi \fallingdotseq 8.23 \cdot 10^5$ of SIN and COS is the limit of the argument of this function.

There are many such examples. Note that the value $2^{18}\pi$ is not a sharp limit and that the number of significant digits for the function decreases gradually as approaching this limit even if within this limit.

When the function value underflows, it is set to 0 without special processing.

(4) Error processing

When the argument exceeds the limit, an message for the error is printed and the calculation is continued with the all function values set as 0. The message consists of the function name, the argument value, the function value (0) and the reason for the error.

Example : ALOG1 ERROR ARG=-0.2000000E+01 VAL=0.0 ARG.LT.-1

The error processing program counts the frequency of the errors and stops the calculation if the frequency exceeds a certain limit, considering the case that the calculation becomes meaningless when the error occurs one after another. Because all users do not want this, you can adopt or reject this processing including the print of the message. Subroutine FNERST is

provided for this purpose and you can use it in the following way.

CALL FNERST(IABORT, MSGPRT, LIMERR)

Argument	Type and kind	Attribute	Content
IABORT	Integer type	Input	IABORT=0 The calculation is not stopped. IABORT≠0 The calculation is stopped.
MSGPRT	Integer type	Input	MSGPRT=0 The message is not printed. MSGPRT≠0 The message is printed.
LIMERR	Integer type	Input	Upper bound of frequency of errors.

If this subroutine is not called, following values are set as a standard value.

IABORT=1, MSGPRT=1, LIMERR=10

(II) Subroutine subprogram

(1) Subroutine name and type

There is no meaning of the type in the head character of the subroutine name. Subroutines with the same purpose and the different type are distinguished by the ending character of the name. The principle is as follows.

Single precision : S Double precision : D Quadruple precision : Q	Complex number : C Double precision complex number : B Quadruple precision complex number : Z	Vector computer single precision : V Vector computer double precision : W Vector computer complex number : X Vector computer double precision complex number : Y

However, there are some exceptions.

Example	Example of exception
LEQLUS/D/Q/C/B RK4S/D/Q/C/B GJMNKS/D/Q	FFTR/FFTRD MINVSP/MINVDP

(2) Argument ··· The following four kinds are distinguished as an attribute of the argument.

Input	Users should set this data before calling the subroutine. As long as it is not especially noticed, the data is preserved as it is at the subroutine exit. This includes the case when the function name and the subroutine name are used as arguments. Note that those names should be declared with EXTERNAL.

4

Output	This data is created in the subroutine and is significant for the user.
Input/Output	Data is output in the same place as the input to save area. When input/output argument is a single variable, you should not specify a constant as a real argument. For instance, if LEQLUS is called with constant 1 specified in input/output argument and is ended normally, IND=0 is output, but all constants 1 are changed to 0.
Work area	It is an area necessary for calculation in a subroutine, and the content of the subroutine at exit is meaningless for users.

The type and attribute of the argument are explained for each subroutine group. The explanation is for single precision. For others, please read it with exchanging the type for the suitable one.

When a subroutine is called with an argument, but the argument is not used, the area for the argument need not be prepared, and anything can be written in that place. The same area can be allocated for the different arguments, only if it is pointed as it like SVDS. There is an example (FT235R) that special demand is requested for the argument.

It is requested for users to provide the function routine and the subroutine for the numerical integration routine and the routine for solving differential equations. In this case, the number, the type, and the order of the argument should be as specified. If parameters except a regulated argument are necessary, they are allocated in COMMON area to communicate with the main program. Refer to the explanation of an individual routine for the example.

---

1) Ichizo Ninomiya; "Current state, issues of mathematical software", information processing, Vol. 23 and pp. 109-117 (1982).

【 Opening source program to the public 】

The following source programs are published for users requesting them. Calculation can be requested directly, and the source list can be output or can be copied in the shared file. The copied program cannot be given to the third party without the permission of this center.

If you need to copy the source list in the card or the data set, please execute following procedures.

(1) Input the following command for TSS.

NLIBRARY ELM (library name) "DS (data set name)" "SLAVE(ON)"

When you need only the source list, you can omit DS and SLAVE. When SLAVE(ON) is specified, all slave routines of the program will be output.

(2) Execute the following job for BATCH.

//EXEC NLIBRARY, ELM=program names[, DS='data set names'][, SLAVE=ON]

You can have examples of the program usage with the following procedures.

(1) For TSS

EXAMPLE NAME (library name) [DS (data set name)]

(2) For BATCH

//EXEC EXAMPLE, NAME=program names[, DS='data set names']

Four kinds of manual listed below are prepared concerning library program.

Number	Manual title	Content
1	Library program and data list	All library programs and data which can be used in this center are listed. Additionally, "description format of the NUMPAC routine and notes on use", "How to choose the NUMPAC routine", and usage of error processing subroutine "FNERST" are described in this list.
2	Guidance to use library program (General volume : GENERAL VOL. 1)	This volume describes the general use of programs except NUMPAC, which can be used in this center.

3	Guidance to use library program (Numerical calculation : NUMPAC VOL. 1)	This volume describes how to use the following five kinds of programs. 1. Basic matrix operations 2. System of linear equations 3. Matrix inversion 4. Eigenvalue analysis 5. Polynomial equation and nonlinear equation
4	Guidance to use library program (Numerical calculation : NUMPAC VOL. 2)	This volume describes how to use the following five kinds of programs. 6. Interpolation, smoothing, and numerical differentiation and integration 7. Fourier analysis 8. Numerical quadrature 9. Ordinary differential equation 10. Elementary function
5	Guidance to use library program (Numerical calculation : NUMPAC VOL. 3)	This volume describes how to use the following nine kinds of programs. 11. Table functions 12. Orthogonal polynomial 13. Special functions 14. Bessel function and related function 15. Acceleration of convergence of sequences 16. Linear programming 17. Special data processing 18. Figure display application program 19. Others

All these manuals can be output by "MANUAL command". "PICKOUT command" is available if you need part of the usage of individual program.

For NUMPAC users

Please note the following and use NUMPAC effectively.

(1) The user has the responsibility for the result obtained by NUMPAC.

(2) When the trouble is found, please report it to the center program consultation corner (Extension 6530).

(3) Do not use NUMPAC in computer systems other than this center without permission.

(4) To publish the result obtained NUMPAC, the used program names (for instance, *** of NUMPAC) should be referred to.

This manual was translated using Fujitsu's machine translation system ATLAS.

II. Library and program itemized discussion

6. Interpolation, smoothing, and numerical differentiation and integration

[Method of choice of interpolation and smoothing routines]

NUMPAC offers a choice of routines depending on the way the data is given and whether the data contains error. When data is given in a form of a function and can be calculated with a function value at any point, the way of giving such data is called a function input. Chebyshev interpolation routines are suitable for such calculation. On the other hand, when discrete points are given as data without error or with a slight error, spline interpolation routines are recommended. If data contains error, the least square approximation routines can be used.

- Function input ——— Chebyshev interpolation FCHB1S, FCHB2S, and FCHB3S
- Discrete point input
  - Interpolation (Given points are passed through.)
    - When derivative values in high precision are required:
      - Spline interpolation DSCI3A and DSCI3D
    - When precision for graphics display is enough:
      - Quasi-Hermitian interpolation HERM31 and HERM32
  - Smoothing (when data includes an error)
    - Linear model: Polynomial approximation LSAICS
    - Non-linear model: NOLLS1 and SALS

10

AGFBS/D and AGFB2S/D (Automatic Grid-Fitting of Irregularly-Spaced Data by BRIGGS' Method)

Automatic Grid-Fitting of Irregularly-Spaced Data by BRIGGS' method

Programmed by	Akihiko Yamamoto in September 1980 and revised in October 1984
Format	Subroutine language: FORTRAN; size: 475, 476, 532, and 533 lines respectively

(1) Outline

AGFBS/D and AGFB2S/D obtain a function value at the mesh point in the rectangular region $S(XO \leqq X \leqq X1, YO \leqq Y \leqq Y1)$ using Briggs' method [1] when the irregularly-spaced function data $Zi(Xi,Yi),(i=1\sim N)$ is given. The region S can come off the region where the function value Zi is distributed. That is, the function values at mesh points that are off the data definition area are also extrapolated, but it is desirable that the region of the function values is as large as the one where $Zi(Xi,Yi)$ is distributed, to keep the reliability of the interpolated (extrapolated) values. If $Zi(Xi,Yi)$ is the function value $(i=1\sim N)$ at the point (Xi,Yi), and the rectangular region S is composed of IX meshes in the X direction and JY meshes in the Y direction, the coordinate (Xk,YL) at each mesh is as follows:

$$x_K=h_x(K-1)+x_0, \qquad (K=1\sim IX)$$

$$y_L=h_y(L-1)+y_0, \qquad (L=1\sim JY)$$

where

$$hx=\frac{x_1-x_0}{IX-1}, \quad hy=\frac{y_1-y_0}{JY-1}$$

Then, the function value at the mesh point (Kk,YL) is obtained with this subroutine, and entered in the array $U(K,L)$.

The calculation method is to solve the difference formula which is converted from a partial

differential equation with such conditions that the total curvature is minimized with the function value $Zi(Xi,Yi)$ as a boundary value, using the iteration method (see bibliography (1) for details).

In the iteration matrix $U(K,L)$, the function value after (P-1) iterations is defined as $U^{P-1}(K,L)$, the function value after P iterations is defined as $U^{P}(K,L)$, and the total of absolute correction values E^{P} is defined as

$$E^{P}=\sum_{i=1}^{IX}\sum_{j=1}^{JY}\left|U^{P}(i,j)-U^{P-1}(i,j)\right|$$

If

$$\frac{E^{P}}{E^{1}}\leq\varepsilon$$

is met for the given $\varepsilon(=E>0)$, the iteration terminates.

If ε takes a very small value, the calculation is terminated at the iteration count $(=IT)$. $U^{0}(K,L)$, that is, the initial state of the iteration matrix $U(K,L)$ can be selected from the following five:

(1) Quadratic surface obtained like least squares method

(2) First plane obtained like least squares method

(3) Average value of $Zi(Xi,Yi)$

(4) All 0.0

(5) $U(K,L)$ of the previous result of call

If the routine is iterated with the quadratic surface as the initial value from N to several tens, convergence often becomes fast. However, if N > several hundreds, it does not change considerably. However, this is not always true because convergence depends on the number of data and its distribution state.

In this method, however, the allowable number of data items Zi which depend essentially on each mesh is only one. Thus, the data Zi used to decide the function value of meshes is selected as

follows:Assume that more than one data item ($Zi;i=1\sim m$) are distributed in a mesh. At this time,

(i) The data $(Z1)$ that is nearest to $U(i,j)$ in AGFBS/D is assumed to be the typical value in the mesh.

(ii) In AGFB2S/D, it is assumed that the average value of Zi ($i=1\sim m$) exists in the their center of gravity and be the typical value in the mesh. However, $m\leq 999$ must be met.

Therefore, the following differences are found between AGFBS/D and AGFB2S/D when two or more data exist in each mesh.

(a) In AGFBS/D, all unnecessary data is rejected even if it exists in S. Thus, CPU time decreases a little as compared with AGFB2S/D. If the existing data containing an abnormal function value is accidentally rejected, grid fitting is normally done. Thus, this method is inadequate for grid fitting including abnormal data detection.

(b) Because in AGFB2S/D, all the data existing in S (more correctly, specified as $iLL=2$) is used, CPU time increases a little as compared with AGFBS/D. If the data containing an abnormal function value exists contrary to AGFBS/D, grid fitting is done with the abnormal state kept. Therefore, it is better to use AGFB2S/D for grid fitting such as checking all the data for abnormal function values. Generally, the rectangular region S is subdivided to an extent where up to two data items exist in a mesh. If no data contains an abnormal function value, the result is almost the same even if either of AGFBS/D and AGFB2S/D is used.

(2) Directions

CALL AGFBS/D(U, KU, NN, IX, JY, X, Y, Z, N, C, W, OM, E, JS, IT, ILL)

CALL AGFB2S/D(U, KU, NN, IX, JY, X, Y, Z, N, C, W, OM, E, JS, IT, ILL)

Argument	Type and kind (*1)	Attribute	Content
U	Real type Two-dimensional array	Input/output	Iteration matrix $U(K,2)$ where meshed function values are entered. It is useful as an input at JS=4 only, and starts the iteration with the result of the previous call as the initial value. The size is $IX*JY$.

Argument	Type and kind (*1)	Attribute	Content
KU	Integer type Two-dimensional array	Input/output	Work area. Iteration-related information is entered. It is useful as an input at JS=4 only, and can continue the iteration with the result of the previous call as an input as is. The size is $IX*JY$.
NN	Integer type	Input	The first subscript in the array declaration of U,KU.
IX	Integer type	Input	Number of mesh points in X direction. $X0$ is counted as 1, and $X1$ is counted as IX.
JY	Integer type	Input	Number of mesh points in Y direction. ($Y0$ is counted as 1, and $Y1$ is counted as JY.)
X	Real type One-dimensional array	Input	Value of irregularly-distributed discrete point input data Xi. The size is N.
Y	Real type One-dimensional array	Input	Value of irregularly-distributed discrete point input data Yi. The size is N.
Z	Real type One-dimensional array	Input	Function value Zi in irregularly-distributed discrete point input data (Xi,Yi). The size is N.
N	Integer type	Input	Number of discrete point input data items Xi,Yi,Zi. $N \geqq 4$

14

Argument	Type and kind (*1)	Attribute	Content
C	Real type one-dimensional array	Input	Rectangular region S where grid-fitting is done is specified. The size is four. $C(1)=X0, C(2)=Y0, C(3)=X1, C(4)=Y1$ should be input. $(X0<X1, Y0<Y1)$
W	Real type One-dimensional array	Input/output	Work area. Iteration-related information is entered. It is useful as an input at $JS=4$ only, and can continue the iteration with the result of the previous call as an input as is. The size is 8N or larger.
OM	Real type	Input	Convergence acceleration coefficient. The value to be input is 1 or 2. The appropriate value is about 1.7. Divergence may occur if the value is too close to 2. If $OM=1$, no acceleration is made (that is, convergence is slow), but no divergence may occur. The output ILL must be checked. $(1 \leq OM \leq 2)$
E	Real type	Input	Convergence criterion ε. If the total of absolute correction values for $U(K,L)$ at each iteration becomes smaller than ε times the first total, the iteration terminates. $\varepsilon=10^{-3}\sim10^{-4}$ is appropriate even though it depends on the case. The output ILL must be checked. $(E>0)$

Argument	Type and kind (*1)	Attribute	Content
JS	Integer type	Input/output	The initial state of U is specified as an input. ($0 \leqq JS \leqq 4$) JS=0・・・The average value of $Zi(Xi,Yi)$ is assumed to be an initial value. =1・・・ The first plane obtained by least squares method is assumed to be an initial value. =2・・・ The quadratic surface obtained by least squares method is assumed to be an initial value. =3・・・The value 0.0 is assumed to be an initial value. =4・・・ U of the result of the previous call is assumed to be an initial value. At this time, the iteration is executed reusing U,KU,W . Thus, U,KU,W must be retained just as it was called. This input is useful when the intermediate iteration process is checked. The total number of data items Zi that did not exist in the rectangular region S or were not used for grid-fitting even though they existed there is entered as an output.
IT	Integer type	Input/output	This argument has the following meanings as an input. ($IT \geqq -1$) IT=-1・・・Only the initialization of U is executed, and iteration is not executed. JS=0~3 must be specified. IT=0・・・ U is initialized, and iteration-related information is calculated, but iteration is not executed. Not only U but also KU,W is output. JS=0~3 must be specified.

Argument	Type and kind (*1)	Attribute	Content
			IT>0··· Iteration count. At least 200 iterations are required for the calculation to settle even though the count depends on OM. M at $E^M/E^1 \leq \varepsilon(=E)$ is entered as an output. If E is a very small value, it takes the value when it was input.
ILL	Integer type	Input/output	This argument has the following meaning as an input for AGFBS/D. When the discrete point input data $Zi(Xi,Yi)$ is near the edge of the region S (that is, $XO \leq Xi \leq XO+2hx, X1-2hx \leq Xi \leq X1, YO \leq Yi \leq YO+2hy$, or $Y1-2hy \leq Yi \leq Y1$ is met), whether to use Zi for fitting is specified as the one on the mesh point. ILL≦1··· The above Zi is not used as all. ILL≧2··· If Zi is within hx/ILL in the X direction and within hy/ILL in the y direction from one of the four corners (mesh points) of the meshes where Zi exists, it is used as one on the mesh point. That is, all of the above Zi is used at ILL=2, but it is rarely used at ILL≧3 depending on the distance. Generally, if the region S is subdivided enough by IX and JY, the data can be input with ILL=2. Represents the following termination statuses as an output. 0····· Normal termination. 10000··· Normal termination. Because an error occurred in the initialization specified with JS, the routine was executed as JS=0. 20000··· Normal termination. According to the determination by E, the iteration terminated at the count that is less than specified with IT. 30000··· Abnormal termination. Limits on the argument was exceeded. 40000··· Abnormal termination. $E^M \geq 10E^1$ is met, and divergence is judged to occur. The routine must be reexecuted with OM reduced a little. 50000··· Abnormal termination. $E^{20} \geq 0.5E^1$ is met, and divergence is judged to occur. The routine must be reexecuted with OM reduced a little.

*1 For double precision subroutines, all real types should be double precision real types.

(Note 1) All the $Zi(Xi,Yi)$ need not exist in the rectangular region S.

(1) Data reduction and sorting times are shortened.

(2) To reduce the size of the work area, only the data in the S should be input as much as possible.

(Note 2) If $Zi(Xi,Yi)$ is on the mesh point from the beginning, the value (function value) remains constant and does not change in the iteration process.

(Note 3) CPU time at $IX=JY\sim100, IT\sim100, N\sim24(5000)$ is about 1.68 (2.00) seconds. Generally, CPU time increases (decreases) in proportion to the second power with respect to IX, JY and the first power with respect to IT. The CPU time of $Zi(Xi,Yi)$ for sorting and reduction at N to 2000 is about 0.1 second, and increases (decreases) in proportion to the first power with respect to N.

(Note 4) If the number of data items in a mesh exceeds 999 in AGFB2S, the data item of later than the 1000-th is automatically rejected.

(Note 5) Iteration is forcedly terminated in either of the following cases:

(i) $\quad \frac{E^{20}}{E^{1}} \geqq 0.5 \quad (ILL=50000)$

(ii) $\quad \frac{E^{p}}{E^{1}} \geqq 10 \quad (ILL=40000)$

. If iLL≧40000, it is better to reduce and reissue OM.

Bibliography

1) Briggs, I.C. (1974), "Machine Contouring using Minimum Curvature", Geophysics, 39, 39-48.

(1987.08.10) (1988.06.01)

18

CFS1A and SFC1A (Curve Fitting by Splines)

Curve Fitting by Splines

Programmed by	Kazuo Hatano, January 1982
Format	Subroutine language: FORTRAN; size: 593 lines

(1) Outline

SFS1A and SFC1A apply $\overline{f}_r : 1 \leq r \leq N$ to least squares approximation using the *order* k (degree $k-1$) polynomial spline that has

$$a=\overline{x}_1=x_0<x_1<\cdots<x_n=\overline{x}_N=b \tag{1}$$

as nodes when the observation value $\overline{f}_r$ and observation error $\overline{\sigma}_r$ are given at N discrete points.

Let's define the normalized B-splines as follows:

$$N_j(x)=(t_{j+k}-t_j)g_k[t_j, t_{j+1}, \cdots, t_{j+k}:x]$$

$$g_k(t;x)=(t-x)_+^{k-1}=\begin{cases}(t-x)^{k-1} : t \leq x \\ 0 : t<x\end{cases} \tag{2}$$

$$t_j=\begin{cases}x_0 : -k+1 \leq j \leq -1 \\ x_j : \quad 0 \leq j \leq n \\ x_n : \quad n+1 \leq j \leq n+k-1\end{cases}$$

The coefficients $c_j : -k+1 \leq j \leq n-1$ of the linear combination

$$S(x)=\sum_{j=-k+1}^{n-1} c_j N_j(x) \tag{3}$$

of the normalized B-spline $N_j(x)$ is determined so that the square sum of residuals

$$J=\sum_{r=1}^{N} \frac{1}{\overline{\sigma}_r^2}\left\{\overline{f}_r-\sum_{j=-k+1}^{n-1} c_j N_j(\overline{x}_r)\right\}^2 \tag{4}$$

be minimized. (CFS1A)

Also, expression (3) is calculated for the given variable x. (SFC1A)

(2) Directions

CALL CPS1A(XR, FR, SIGMAR, XI, CJ, DRESP, STAT1, IHIST, PERCT, WORKC, IWORKC, N, K, KOSU, IWR, ICON)

Argument	Type and kind	Attribute	Content
XR	Real type One-dimensional array	Input	Discrete point $\overline{x}_r: 1 \leqq r \leqq N$. Size N.
FR	Real type One-dimensional array	Input	Observation value $\overline{f}_r: 1 \leqq r \leqq N$. Size N.
SIGMAR	Real type One-dimensional array	Input	Measurement error $\overline{\sigma}_r: 1 \leqq r \leqq N$. Size N or 1. If $\overline{\sigma}_r$ differs with r, IWR=1 is assumed. If $\overline{\sigma}_r$ is constant irrespective of r, it should be entered in SIGMAR(1), and IWR=0 is assumed. At this time, the size of the array SIGMAR can be 1 (array declaration is not required).
XI	Real type One-dimensional array	Input	Node $x_i: 0 \leqq i \leqq n$. Size $n+1$. x_i should be entered in $XI(i+1)$.
CJ	Real type One-dimensional array	Output	Coefficient assigned to B-spline $c_j: -k+1 \leqq j \leqq n-1$. Size $n+k-1$. c_j is entered in $CJ(j+k)$.
DRESP	Real type One-dimensional array	Output	Decrement by coefficient c_j in square sum of residuals. $d_j = c_j \sum_{r=1}^{N} 1/\overline{\sigma}_r^2 \ \overline{f}_r N_j(\overline{x}_r)$. Size $n+k-1$. d_j is entered in DRESP $(j+k)$.

Argument	Type and kind	Attribute	Content
STATI	Real type One-dimensional array	Output	Array of size 3. STATI(1): Square sum of residuals J (expression (4)) is entered. Generally, the relationship $J=\sum_{r=1}^{N}1/\sigma_r^2 f_r^2-\sum_{j=-k+1}^{n-1}dj$ exists. STATI(2): $\sigma=J/(N-(n+k-1))$ is entered. If this value is approximately 1, the result is assumed to be appropriate. STATI(3): The amount $AIC=Ne_w J+2(n+k-1)$ is entered.
IHIST	Integer type Two-dimensional array	Output	The residual histogram is entered. Size IHIST(2, 25). The number of r's that meets $-0.2i \geqq \{\bar{f}_r-\sum_{j=-k+1}^{n-1}c_j N_j(\bar{x}_r)\}/\bar{\sigma}_r > -0.2(i+1)$ is entered in IHIST(1, i+1). The number of r's that meets $0.2i < \{\bar{f}_r-\sum_{j=-k+1}^{n-1}c_j N_j(\bar{x}_r)\}/\bar{\sigma}_r \leqq 0.2(i+1)$ is entered in IHIST(2, i+1).
PERCT	Real type One-dimensional array	Output	The cumulative frequency distribution of residuals is entered. Size PERCT(10). If the number of r's that meets $i-1 \leqq |\bar{f}_r-\sum_{j=-k+1}^{n-1}c_j N_j(\bar{x}_r)|/\bar{\sigma}_r < i$ is $K(i)$, $PERCT(i)=\sum_{j=1}^{i}K(j)/N$: $i=1,2,\cdots,10$

Argument	Type and kind	Attribute	Content
WORKC	Real type One-dimensional array	Work area	Size $WORKC((n+k+N-1)(k+1)-1)$.
IWORKC	Integer type One-dimensional array	Work area	Size $IWORKC(N+n+k-1)$.
N	Integer type	Input	n. Number of nodes - 1.
K	Integer type	Input	k. Order of splines.
KOSU	Integer type	Input	N. The number of data items N.
IWR	Integer type	Input	0 or 1. Determines whether the measurement error $\overline{\sigma}_r$ is constant irrespective of the data. If $\overline{\sigma}_r$ is differs with r, IWR=1 is assumed. If $\overline{\sigma}_r$ is constant irrespective of r, IWR=0 is assumed.
ICON	Integer type	Output	ICON=0: Normal termination. ICON<0: Abnormal termination.

CALL SPC1A(XP, I, L, FP, N, K, XI, CJ, WORKF, ICON)

Argument	Type and kind	Attribute	Content
XP	Real type	Input	Point x where S(x) is to be calculated. $x_0 \leq x \leq x_n$ must be met.

22

Argument	Type and kind	Attribute	Content
I	Integer type	Input/output	i that meets $x_i \leq n < x_{i+1}$. $XI(I+1) \leq XP < XI(I+2)$.
L	Integer type	Input	This subroutine can calculate the l-th order derivative of $S(x)$. l in $S^{(l)}(x)$ to be evaluated. $0 \leq l \leq k-1$ must be met.
FP	Real type	Output	Calculated value of $S^{(l)}(x)$.
N	Integer type	Input	Same as CFS1A.
K	Integer type	Input	Same as CFS1A.
XI	Real type One-dimensional array	Input	Same as CFS1A.
CJ	Real type One-dimensional array	Input	Coefficient assigned to B-spline. $c_j: -k+1 \leq j \leq n-1$. Size $n+k-1$. Output of CFS1A.
WORKF	Real type One-dimensional array	Work area	Size k.
ICON	Integer type	Output	ICON=0: Normal termination. ICON<0: Abnormal termination.

(1987. 05. 20) (1987. 05. 20) (1988. 04. 22)

CFS2A and SFS1A (Surface Fitting by Splines)

Surface Fitting by Splines

Programmed by	Kazuo Hatano, January 1982
Format	Subroutine language: FORTRAN

(1) Outline

CFS2A and SFS1A apply $\overline{f}_{r,s}$: $1 \leqq r \leqq M$, $1 \leqq s \leqq N$ to least squares approximation using the k-th (Bi $k-1$-st) degree polynomial spline that has

$$a=\overline{x}_1=x_0<x_1<\cdots<x_\nu=\overline{x}_M=b \tag{1}$$

as the x direction node and

$$c=\overline{y}_1=y_0<y_1<\cdots<y_n=\overline{y}_N=d \tag{2}$$

as the y direction node when the observation value $\overline{f}_{r,s}$, and the observation error $\overline{\lambda}_r \cdot \overline{\mu}_s$ are give at M and N mesh points $(\overline{x}_r, \overline{y}_s): 1 \leqq r \leqq M$, $1 \leqq s \leqq N$. That is, the coefficient $C_{\alpha,\beta}: -k+1 \leqq \alpha \leqq m-1, -k+1 \leqq \beta \leqq n-1$ of the normalized B-spline bilinear combination

$$S(x,y)=\sum_{\alpha=-k+1}^{m-1} \sum_{\beta=-k+1}^{n-1} C_{\alpha,\beta} N_\alpha(x) N_{\beta(y)} \tag{3}$$

is determined so that the square sum of residuals

$$J=\sum_{r=1}^{M} \sum_{s=1}^{N} \frac{1}{\overline{\lambda}_r{}^2 \cdot \overline{\mu}_s{}^2} \left\{ \overline{f}_{r,s} - \sum_{\alpha=-k+1}^{m-1} \sum_{\beta=-k+1}^{n-1} C_{\alpha,\beta} N_\alpha(\overline{x}_r) N_\beta(\overline{y}_s) \right\}^2 \tag{4}$$

be minimized. (CFS2A)

Expression (3) is calculated for given variables x, y. (SFS1A)

(2) Directions

CALL CFS2A(XR, YS, FRS, SIGMXR, SIGMYS, XI, YJ, CAB, DRESP, STATI, IHIST, PERCT, WORKC, IWORKC, KOSUX, KOSUY, NX, NY, K, IWR, KOSXD, NXK1D, ICON)

Argument	Type and kind	Attribute	Content
XR	Real type One-dimensional array	Input	Coordinates $\overline{x}_r: 1 \leqq r \leqq M$ at mesh points in x direction. One-dimensional array of size M.
YS	Real type One-dimensional array	Input	Coordinates $\overline{y}_s: 1 \leqq s \leqq N$ at mesh points in y direction. One-dimensional array of size N.
FRS	Real type Two-dimensional array	Input	Observation value $\overline{f}_{r,s}: 1 \leqq r \leqq M, 1 \leqq s \leqq N$. Two-dimensional array of size NX*NY.
SIGMXR	Real type One-dimensional array	Input	The measurement error of $\overline{f}_{r,s}$ is given by the two-number product $\overline{\lambda}_r, \overline{\mu}_s$. $\overline{\lambda}_r: 1 \leqq r \leqq M$ should be entered in SIGMXR. Size M or 1. If $\overline{\lambda}_r, \overline{\mu}_s$ differ with r,s, WR=1 must be assumed. If $\overline{\lambda}_r, \overline{\mu}_s$ are constant irrespective of r,s, they should be entered in SIGMXR(1) and SIGMYS(1) respectively, and IWR=0 is assumed. At this time, the size of the arrays SIGMXR and SIGMYS can be 1 (array declaration is not required).
SIGMYS	Real type One-dimensional array	Input	$\overline{\mu}_s: 1 \leqq s \leqq N$ in $\overline{\lambda}_r, \overline{\mu}_s$, the measurement errors of $\overline{f}_{r,s}$, should be entered. Size N or 1.
XI	Real type One-dimensional array	Input	Node $x_i: 0 \leqq i \leqq m$ of x direction. Size $m+1$. x_i should be entered in $XI(i+r)$.

Argument	Type and kind	Attribute	Content
YJ	Real type One-dimensional array	Input	Node y_j: $0 \leq j \leq n$ of y direction. Size $n+1$. y_j should be entered in $YJ(j+1)$.
CAB	Real type Two-dimensional array	Output	Coefficients assigned to B-spline: $c_{\alpha,\beta}$: $-k+1 \leq \alpha \leq m-1$, and $-k+1 \leq \beta \leq n-1$. Size $(m+k-1)(n+k-1)$. $c_{\alpha,\beta}$ is entered in $CAB(\alpha+k, \beta+k)$.
DRESP	Real type Two-dimensional array	Output	Decrease by coefficient $c_{\alpha,\beta}$ in the square sum of residuals. $d_{\alpha,\beta}=c_{\alpha,\beta}\sum_{r=1}^{M}\sum_{s=1}^{N}1/\ \bar{\lambda}_r^2\bar{\mu}_s^2\bar{f}_{r,s}\times N_\alpha(\bar{x}_r)N_\beta(\bar{y}_s)$. The size $(m+k-1)\cdot(n+k-1)\alpha_{\alpha,\beta}$ is entered in DRESP $(\alpha+k, \beta+k)$.
STATI	Real type One-dimensional array	Output	Array of size 3 STATI(1): Squares sum of residuals, J (expression (4)), is entered. Generally, the relationship $J=\sum_{\alpha=-k+1}^{m-1}\sum_{\beta=-k+1}^{n-1}d_{\alpha,\beta}$ exists. STATI(2): The amount $\sigma=J/(MN-(m+k-1)(n+k-1))$ is entered. If this value is approximately 1, the result is assumed to be appropriate. STATI(3): The amount $AIC=M\cdot Nl_nJ+2(m+k-1)\times(n+k-1)$ is entered.

Argument	Type and kind	Attribute	Content
IHIST	Integer type Two-dimensional array	Output	A residual histogram is entered. Size IHIST(2, 25). The number of values (r,s) that meets $-0.2i \geq \{ \bar{f}_{r,s} - \sum_{\alpha=-k+1}^{m-1} \sum_{\beta=-k+1}^{n-1} C_{\alpha,\beta} N_\alpha(\bar{x}_r) N_\beta(\bar{y}_s) \} / \bar{\lambda}_r \cdot \bar{\mu}_s > -0.2(s+1)$ is entered in IHIST$(l, i+1)$. The number of values (r,s) that meets $-0.2i < \{ \bar{f}_{r,s} - \sum_{\alpha=-k+1}^{m-1} \sum_{\beta=-k+1}^{n-1} C_{\alpha,\beta} N_\alpha(\bar{x}_r) N_\beta(\bar{y}_s) \} / \bar{\lambda}_r \cdot \bar{\mu}_s \leq 0.2(i+1)$ is entered in IHIST$(2, i+1)$.
PERCT	Real type One-dimensional array	Output	The cumulative frequency distribution of residuals is entered. Size PERCT(10). If the number of values (r,s) that meets $i-1 \leq \left| \bar{f}_{r,s} - \sum_{\alpha=-k+1}^{m-1} \sum_{\beta=-k+1}^{n-1} C_{\alpha,\beta} \cdot N_\alpha(\bar{x}_r) N_\beta(\bar{y}_s) \right| / \bar{\lambda}_r \cdot \mu$ is $K(i)$, $PERCT(i) = \sum_{j=1}^{i} K(j)/(MN) : i = 1, 2, \cdots, 10$.
WORKC	Real type One-dimensional array	Work area	Size $k \times \{ m+n+min(M,N) \}$.
IWORKC	Integer type One-dimensional array	Work area	Size $M+N+m+n+2$.

Argument	Type and kind	Attribute	Content
KOSUX	Integer type	Input	M in the number of data items $M \cdot N$ (number of remainders in x direction).
KOSUY	Integer type	Input	N in the number of data items $M \cdot N$ (number of remainders in y direction).
NX	Integer type	Input	m. Number of nodes - 1 in x direction.
NY	Integer type	Input	n. Number of nodes - 1 in y direction.
K	Integer type	Input	k. Order of splines.
IWR	Integer type	Input	0 or 1. Whether the measurement error $\overline{\lambda}_r \cdot \overline{\mu}_s$ is constant irrespective of data is specified. If $\overline{\lambda}_r \cdot \overline{\mu}_s$ differs with r,s, IWR=1 is assumed. If $\overline{\lambda}_r \cdot \overline{\mu}_s$ is constant irrespective of r,s, IWR=0 is assumed.
KOSXD	Integer type	Input	The first subscript of adjustable array FRS. KOSXD≧KOSUX must be met.
NXK1D	Integer type	Input	The first subscript of adjustable arrays CAB and DRESP. NXK1D≦NX+K-1.
ICON	Integer type	Output	ICON=0: Normal termination. ICON<0: Abnormal termination.

CALL SFS1A(XP, YP, IX, IY, LX, LY, FP, NX, NY, K, XI, YJ, CAB, WORKF, NXK1D, ICON)

Argument	Type and kind	Attribute	Content
XP, YP	Real type	Input	Point (x,y) where (x,y) is to be calculated. $x_0 \leqq x \leqq x_m, y_0 \leqq y \leqq y_n$. x should be entered in XP, and y should be entered in YP.

Argument	Type and kind	Attribute	Content
IX, IY	Integer type	Input	i that meets $x_0 \leqq x < x_{\delta+1}$, and j that meets $y_j \leqq y < y_{\rho+1}$. i should be entered in IX, and j should be entered in IY. XI(IX+1)≦XP<XI(IX+2), YJ(IY+1)≦YP<YJ(IY+2)
LX, LY	Integer type	Input	This subroutine can calculate the partial differential of $S(x,y)$, $\partial^{\lambda+\mu}S(x,y)/\partial x^{\lambda}\partial y^{\mu}$. $\lambda,\mu 0 \leqq \lambda,\mu \leqq k-1$ in $S^{(\lambda,\mu)}(x,y)$ to be evaluated.
FP	Real type	Output	Calculated value of $S^{(\lambda,\mu)}(x,y)$.
NX, NY	Integer type	Input	Same as CFS2A.
K	Integer type	Input	Same as CFS2A.
XI, YJ	Real type One-dimensional array	Input	Same as CFS2A.
CAB	Real type Two-dimensional array	Input	Coefficient assigned to B-spline. $c_{\alpha,\beta}$: $-k+1 \leqq \alpha \leqq m-1$, $-k+1 \leqq \beta \leqq n-1$. Size $(m+k-1)(n+k-1)$. Output of CFS2A.
WORKF	Real type One-dimensional array	Work area	Size $m+5k-1$.
NXK1D	Integer type	Input	The first subscript of adjustable array CAB. NXK1D≧NX+K-1.
ICON	Integer type	Input	ICON=0: Normal termination. ICON<0: Abnormal termination.

(1987. 05. 28)

DCOMD1 and DCPFR1 (Curve Fitting by Composite Polynomials)

Curve Fitting by Composite Polynomials

Programmed by	Kazuo Hatano, January 1982
Format	Subroutine language: FORTRAN; size: 1491 lines

(1) Outline

DCOMD1 and DCPFR1 apply $f(x):0\leqq x\leqq 2\pi$ to least squares approximation using the composite polynomial

$$h(x)=\frac{1}{2}c_0+\sum_{j=1}^{n-1}(c_j\cos jx+b_j\sin jx)+\sum_{i=1}^{m}c_i q_i(x;n) \qquad (2)$$

when the function values $f(\bar{x}_r)$ are given at equally spaced N+1 discrete points

$$\bar{x}_r=\frac{2\pi r}{N}\ :\ r=0,1,\cdots,N \qquad (1)$$

Assume

$$\begin{cases} q_{2i}(x;n)=\sum_{j=n}^{\infty}\frac{(+)^{i-1}}{j^{2i}}\cos jx \\ q_{2i+1}(x;n)=\sum_{j=n}^{\infty}\frac{(+)^{i-1}}{j^{2i+1}}\sin jx \end{cases} \qquad (3)$$

The coefficients a_0, a_j, $b_j:j=1,2,\cdots,n-1$, $c_i:i=1,2,\cdots,m$ of $h(x)$ are determined so that the constant multiple of square sum of residuals

$$J=\frac{2}{N}\sum_{r=0}^{N}\{f(x_r)-h(x_r)\}^2 \qquad (4)$$

be minimized. (DCOMD1).

Expression (2), $h(\tilde{x}_s)$, at given equally spaced discrete points $\tilde{x}_s=2\pi s/K:s=0,1,\cdots,K$ is calculated. Suppose that K is a multiple of N.

(2) Directions

CALL DCOMD1(FR, NL, ABJ, CJ, NS, MDEG, WORK, ICON)

Argument	Type and kind	Attribute	Content
FR	Double precision real type One-dimensional array	Input	Function value $f(\bar{x}_r)$ at equally spaced discrete points. $0 \leq r \leq N$, size N+1
NL	Integer type	Input	N. Number of data items - 1. N must be an even number.
ABJ	Double precision real type One-dimensional array	Output	$a_0, a_j, b_j : 1 \leq j \leq n-1$ is entered. Size N (partly used as a work area).
CJ	Double precision real type One-dimensional array	Output	$c_i : 1 \leq i \leq m$ is entered. Size m. m must be an even number of up to 12.
NS	Integer type	Input	n is given.
MDEG	Integer type	Input	m is given. m must be an even number of up to 12.
WORK	Double precision real type One-dimensional array	Work area	The size depends on N. If v is an integer, and $N=2^v$, the size is 1. If v is not an integer, and N is an even number, the size is N.
ICON	Integer type	Output	ICON=0: Normal termination. ICON<0: Abnormal termination.

CALL DCPFR1(ABJ, CJ, NS, NCUT, MDEG, FR, NL, WORK, ICON)

Argument	Type and kind	Attribute	Content
ABJ	Double precision real type One-dimensional array	Input	$a_0, a_j, b_j : 1 \leq j \leq n-1$. Output of DCOMD1. Size N
CJ	Double precision real type One-dimensional array	Input	$c_i : 1 \leq i \leq m$. Output of DCOMD1, size m

Argument	Type and kind	Attribute	Content
NS	Integer type	Input	n is given.
NCUT	Integer type	Input	n is given. (Same value as NS is assigned.)
MDEG	Integer type	Input	m. Even number of up to 12.
FR	Double precision real type One-dimensional array	Output	Approximate value $f(\bar{x}_s): 0 \leq s \leq K$ at equally spaced discrete points. Size K+1
NL	Integer type	Input	K is given. Number of approximate values to be obtained - 1.
WORK	Double precision real type	Work area	The size depends on K. If v is an integer, and $K=2^v$, the size is 1. If v is not an integer, and K is an even, the size is K.
ICON	Integer type	Output	ICON=0: Normal termination. ICON<0: Abnormal termination.

(1987. 05. 15) (1987. 08. 08) (1987. 08. 10)

DSCI1A,DSCI2A,DSCI3A,DSCI4A,DSFI1A,DSFI2A,DSFI3A,DSFI4A

(Spline Interpolation (One Dimensional))

Spline Interpolation (One Dimensional)

Programmed by	Kazuo Hatano, June 1978
Format	Subroutine language: FORTRAN; size; 298, 141, 263, 141, 280, 150, 389, and 176 lines respectively

(1) Outline

If function values are given at each discrete points, and in some cases, end conditions are given at both ends

(1) Subroutines whose third character is C constitute the following $2m-1\,(m\geqq 2)$-th order polynomial splines that pass through given points and meet the end conditions.

(2) Subroutines whose third character is F obtain the function value (interpolated value) of the constituted $2m-1$-th order polynomial spline and $l\,(1\leqq l\leqq 2m-1)$-th order derivative, and calculate the integral from the left end to a given point.

The following four types are available depending on the conditions given at the end points.

(1) Type-I spline interpolation (2) Type-II spline interpolation

(3) Type-III spline interpolation (4) Periodic spline interpolation

1. Type-I spline interpolation

If the differential coefficients $f^{(l)}(x_0)$, $f^{(l)}(x_n)$, $(1\leqq l\leqq m-1)$ of up to the $m-1$-st order are given at both ends x_0, x_n of the function value $f(x_i)$ of $n+1$ points $x_0<x_1<\cdots<x_n$ $(n\geqq 1)$, $f(x)$ is interpolated with the $2m-1$-th order $(m\geqq 2)$ polynomial spline $S(x)=\sum_{j=-2m+1}^{n-1} c_j N_j(x)$ that assigns x_0, x_n as the $2m$ node and x_i, $(1\leqq i\leqq n-1)$ as a single node. $N_j(x)$, $(-2m+1\leqq j\leqq n-1)$ are normalized B-splines which are defined as follows:

$$g_{2m}(t;x)=(t-x)_+^{2m-1}=\begin{cases}(t-x)^{2m-1} & (t\geqq x)\\ 0 & (t<x)\end{cases}$$

$$N_j(x)=(t_{j+2m}-t_j)g_{2m}[\,t_j,t_{j+1},\cdots,t_{j+2m};x\,]$$

$$t_j=\begin{cases}x_0 & (-2m+1\leq j\leq -1)\\ x_j & (0\leq j\leq n)\\ x_n & (n+1\leq j\leq n+2m-1)\end{cases}$$

The interpolation coefficients $c_j,(-2m+1\leq j\leq n-1)$ are found out with the subroutine DSCI1A, and $S(x),S^{(l)}(x),\int_{x0}^{x}S(x)dx$ to $x_0\leq x\leq x_n$ are found out with DSFI1A. If the differential coefficients of up to the $m-1$-st order can be given at both ends, it is desirable to use these routines. The hightest precision may be expected by this subroutine among four types.

2. Type-II spline interpolation

If the differential coefficients $f^{(l)}(x_0)$, $f^{(l)}(x_n),(m\leq l\leq 2m-2)$ from m-th to $2m-2$-nd orders are given at both ends x_0,x_n of the function value $f(x_i)$ of $n+1$ points $x_0<x_1<\cdots<x_n$ $(n\geq m-1)$, $f(x)$ is interpolated with the $2m-1$-th order ($(m\geq 2)$) polynomial spline $S(x)=\sum_{j=-2m+1}^{n-1}c_jN_j(x)$ that assigns x_0,x_n as the $2m$ node and $x_i(1\leq i\leq n-1)$ as a single node. $N_j(x)$ is the same as with TYPE-1.

The interpolation coefficient c_j, $(-2m+1\leq j\leq n-1)$ are found out with the subroutine DSCI2A, and $S(x)$, $S^{(l)}(x)$, $\int_{x0}^{x}S(x)dx$ to $x_0\leq x\leq x_n$ are found out with DSFI2A.

The usefulness of this program may be the lowest of the four types. However, the $2m-1$-st order interpolation spline that can be obtained by assigning 0 to the differential coefficient from the m-th to $2m-2$-th orders at both ends is called "Natural spline" and most famous in spline applications. A natural spline can be constituted by using this routine. In most cases, however, it is large in error as compared with the following type-III spline interpolation:

3. Type-III spline interpolation

If the function values $f(x_i)$ are given at $n+1$ points $x_0<x_1<\cdots<x_n$ $(n\geq 2m)$, the equation $f(x)$ is interpolated with the $2m-1$-st order ($(m\geq 2)$) polynomial spline $S(x)=\sum_{j=-2m+1}^{n-2m+1}c_jN_j(x)$ that assigns x_0,x_n as $2m$ nodes and $x_i,(m\leq i\leq n-m)$ as a single node. $N_j(x),(-2m+1\leq j\leq n-2m+1)$ are the normalized B-splines which are defined as follows:

$$g_{2m}(t;x)=(t-x)_+^{2m-1}=\begin{cases}(t-x)^{2m-1} & (t\geqq x)\\ 0 & (t<x)\end{cases}$$

$$N_j(x)=(t_{j+2m}-t_j)g_{2m}[t_j,t_{j+1},\cdots,t_{j+2m};x]$$

$$t_j=\begin{cases}x_0 & (-2m+1\leqq j\leqq 0)\\ x_{j+m-1} & (1\leqq j\leqq n-2m+1)\\ x_n & (n-2m+2\leqq j\leqq n+1)\end{cases}$$

The Interpolation coefficients c_j, $(-2m+1\leqq j\leqq n-2m+1)$ are found out with the subroutine DSCI3A, and $S(x)$, $S^{(l)}(x)$, $\int_{x0}^{x}S(x)dx$ to $x_0\leqq x\leqq x_n$ are found out with DSFI3A. Because this type enables interpolation using only the function value, it is most useful if $f(x)$ is a non-periodic function.

4. Periodic spline interpolation

It is assumed that the interpolated function $f(x)$ is a periodic function of period x_n-x_0, and the function values $f(x_i)$ are given at $n+1$ points $x_0<x_1<\cdots<x_n$ $(n\geqq 2m)$. At this time, $f(x)$ is interpolated with the function $S(x)$ defined as follows:

$$g_{2m}(t;x)=(t-x)_+^{2m-1}=\begin{cases}(t-x)^{2m-1} & (t\geqq x)\\ 0 & (t<x)\end{cases}$$

$$N_j(x)=(t_{j+2m}-t_j)g_{2m}[t_j,t_{j+1},\cdots,t_{j+2m};x]$$

$$t_j=\begin{cases}x_{n+j}-(x_n-x_0) & (-2m+1\leqq j\leqq -1)\\ x_j & (0\leqq j\leqq n)\\ x_{j-n}+(x_n-x_0) & (n+1\leqq j\leqq n+2m-1)\end{cases}$$

$$S(x)=\sum_{j=-2m+1}^{n-1}c_jN_j(x)$$

$$\begin{cases}c_j=c_{j+n} & (-2m+1\leqq j\leqq -m)\\ c_j=c_{j-n} & (n-m+1\leqq j\leqq n-1)\end{cases}$$

The $2m-1$, $(m\geqq 2)$-st order polynomial spline $S(x)$ defined with the above expressions can be assumed to be a periodic function in the meaning that $S^{(l)}(x_0)=S^{(l)}(x_n)$ $(0\leqq l\leqq 2m-2)$ is

satisfied.

The interpolation coefficients c_j, $(-2m+1 \leq j \leq n-1)$ are found out with the subroutine DSCI4A, and $S(x)$, $S^{(l)}(x)$, $\int_{x_0}^{x} S(x)dx$ to $x_0 \leq x \leq x_n$ is found out with DSFI4A. It is recommended to use these routines if $f(x)$ is a periodic function.

(2) Directions

CALL DSCI1A(XI, F, DER, CJ, N, M, WORKC)

CALL DSFI1A(XP, I, L, FP, N, M, XI, CJ, WORKF)

Argument	Type and kind	Attribute	Content
XI	Double precision real type One-dimensional array	Input	Discrete point x_i. Array of size $n+1$. $x_i, (0 \leq i \leq n)$ should be entered in $XI(i+1)$.
F	Double precision real type One-dimensional array	Input	Function value $f(x_i), (0 \leq i \leq n)$ at discrete point x_i. Array of size $n+1$. $f(x_i)$ should be entered in $F(i+1)$.
DER	Double precision real type Two-dimensional array	Input	The l-th order differential coefficient $(1 \leq l \leq m-1)$ at the end point x_0, x_n. Two-dimensional array of size $(2, m-1)$. $f^l(x_n)$ should be entered in $DER(2, l)$ in $f^{(l)}(x_0) DER(1, l)$.
CJ	Double precision real type One-dimensional array	Input/output	Output in DSCI1A. Input in DSCFIA. Interpolation coefficient c_j, $(-2m+1 \leq j \leq n-1)$. Array of size $n+2m-1$. c_j is entered in $CJ(j+2m)$.

Argument	Type and kind	Attribute	Content
N	Integer type	Input	Number of discrete points. n in $n+1$ should be entered.
M	Integer type	Input	m in the order $2m-1$ of splines should be entered.
WORKC	Double precision real type One-dimensional array	Input/output	Work area. Array of size $(n-1)(2m-1)+2m^2+2m$.
XP	Double precision real type	Input	Point x where interpolated values are to be evaluated. XI(1)≦XP≦XI(N+1) must be met. If XP in the outside of this range is given, error messages are printed, and FP=0.0 is assigned.
I	Integer type	Input	The integer I that meets XI(I+1)≦XP<XI(I+2) should be entered. Even if I does not meet the above condition, the computation is normally executed. However, the calculation time is required a little more for search.
L	Integer type	Input	Integer that complies with -1≦L≦2*M-1. A calculation type is given. L=-1: Indefinite integral from MI(1) to XP is calculated and output to FP. L=0: Interpolation value at XP is calculated and output to FP. 1≦L≦2*M-1: L-th order differential coefficient at XP is calculated and output to FP. L<-1 and L>2*M-1: Error messages are printed, and FP=0.0 is assigned.

Argument	Type and kind	Attribute	Content
FP	Double precision real type	Output	Calculation results such as interpolation values are entered.
WORKF	Double precision real type One-dimensional array	Input/output	Work area. Array of size $2m$.

CALL DSCI2A(XI, F, DER, CJ, N, M, WORKC)

CALL DSFI2A(XP, I, L, FP, N, M, XI, CJ, WORKF)

Argument	Type and kind	Attribute	Content
DER	Double precision real type Two-dimensional array	Input	Two-dimensional array at the point x_0, x_n of l-th order differential coefficient $(m \leq l \leq 2m-2)$ and size x_0, x_n. $f^{(l)}(x_0)$ should be entered in $DER(1, l-m+1)$, and $f^{(l)}(x_n)$ should be entered in $DER(2, l-m+1)$.
WORKC	Double precision real type One-dimensional array	Input/output	Work area. Array of size $(n+2m-3)(2m-1)+4m$.
For other arguments, see the Type-I spline. However, CJ is an output in DSCI2A and an input in DSFI2A.			

CALL DSCI3A(XI, F, CJ, X30, N, M, WORKC)

CALL DSFI3A(XP, I, L, FP, N, M, XI, CJ, X30, WORKF)

Argument	Type and kind	Attribute	Content
CJ	Double precision real type One-dimensional array	Input/output	Output in DSCI3A. Input in CSFI3A. Interpolation coefficient c_j, $(-2m+1 \leq j \leq n-2m+1)$. Array of size $n+1$. c_j is entered in $CJ(j+2m)$.
X30	Double precision real type One-dimensional array	Input/output	Output in DSCI3A. Input in DSFI3A. Nodes $x_0, x_m, x_{m+1}, \cdots, x_{n-m}, x_n$ of splines are entered. Array of size $n-2m+3$.
WORKC	Double precision real type First column array	Input/output	Work area. Array of size $(n-1)(2m-1)+4m$.
Other arguments are the same as with the Type-I spline.			

CALL DSCI4A(XI, F, CJ, N, M, WORKC)

CALL DSFI4A(XP, I, L, FP, N, M, XI, CJ, WORKF)

Argument	Type and kind	Attribute	Content
WORKC	Double precision real type One-dimensional array	Input/output	Work area. Array of size $n(2m-1)+2n(m-1)+2m$.
For other arguments, see the type I spline. (CJ is an output in DSCI4A, and an input in DSFI4A.)			

(3) Note

It is recommended to use the four subroutines properly as described below depending on the characteristics of the function $f(x)$.

1. If $f(x)$ is a periodic function, DSCI4A and DSCI4F are used.

2. If the differential coefficients $f^{(l)}(x_0)$, $f^{(l)}(x_n)$, $(1 \leq l \leq m-1)$ of $f(x)$ can be given at both ends, DSCI1A and DSFI1A are used. For instance, the first order differential coefficient at both ends is given for the cubic spline $(m=2)$ interpolation.

3. If only the function values are given, DSCI3A and DSFI3A are used.

4. For interpolation with the so-called "Natural spline," DSCI2A and DSFI2A are used.

(1987. 05. 15)

DSCI1D, DSCI2D, DSCI3D, DSCI4D, DSCI5D, DSCI6D, DSCI7D, DSFI1D, DSFI2D, DSFI3D, DSFI4D, DSFI5D, DSFI6D, and DSFI7D

(Spline interpolation (two-dimensional))

Spline Interpolation (Two Dimensional)

Programmed by	Kazuo Hatano; June 1978
Format	Subroutine language; FORTRAN, Size; About 300 lines each

(1) Outline

When function values are given in grid points in a rectangular region and required boundary conditions are given at the boundary, the subroutine (with the third character of its name being C) makes polynomial spline $S(x,y)$ at dual degree $2\nu-1\,(\nu\geq 2)$ which passes the given points and satisfies the boundary conditions.

The subroutine (with the third character of its name being F) evaluates the function values (interpolation values), partial derivatives $\partial^{\lambda+\mu}S(x,y)/\partial^{\lambda}x\partial^{\mu}y\,(0\leq\lambda\leq 2\nu-1, 0\leq\mu\leq 2\nu-1)$, and indefinite integral $\int_{y_0}^{y}\int_{x_0}^{x}S(x,y)dxdy$ of the polynomial spline $S(x,y)$ of bi- $2\nu-1$ degree

The following seven types of interpolations are available depending on the conditions given at the boundary:

(1) (Type-Ⅰ)×(Type-Ⅰ) spline interpolation (5) (Type-Ⅰ)×(periodic) spline interpolation

(2) (Type-Ⅱ)×(Type-Ⅱ) spline interpolation (6) (Type-Ⅱ)×(periodic) spline interpolation

(3) (Type-Ⅲ)×(Type-Ⅲ) spline interpolation (7) (Type-Ⅲ)×(periodic) spline interpolation

(4) (periodic)×(periodic) spline interpolation

1. (Type-I) × (Type-I) spline interpolation

$$\begin{cases} a=x_0<x_1<\cdots<x_m=b \\ c=y_0<y_1<\cdots<y_n=d \end{cases} \tag{1}$$

is given. When the following is given for two-dimensional function $f(x,y)$:

(1) $f_{i,j}=f(x_i,y_j) \quad (0\leq i\leq m),(0\leq j\leq n)$

(2) $f_{i,j}^{(\lambda,0)}=f^{(\lambda,0)}(x_i,y_j) \quad (i=0,m),(0\leq j\leq n)$

(3) $f_{i,j}^{(0,\mu)}=f^{(0,\mu)}(x_i,y_j) \quad (j=0,n),(0\leq i\leq m)$ (2)

(4) $f_{i,j}^{(\lambda,\mu)}=f^{(\lambda,\mu)}(x_i,y_j) \quad (i=0,m),(j=0,n)$

$(1\leq\lambda\leq\nu-1),(1\leq\mu\leq\nu-1)$

That is, the following conditions are met:

(1) Function values are given for all grid points.

(2) Normal derivative $\partial^{\lambda}f/\partial x^{\lambda}(1\leq\lambda\leq\nu-1)$ to the degree of $\nu-1$ in the x direction is given for the grid point on $x=x_0=a, x=x_m=b$.

(3) Normal derivative $\partial^{\mu}f/\partial y^{\mu}(1\leq\mu\leq\nu-1)$ to the degree $\nu-1$ in the y direction is given for the grid point on $y=y_0=c, y=y_n=d$.

(4) Partial derivative $\partial^{\lambda+\mu}/\partial x^{\lambda}\partial y^{\mu}(1\leq\lambda,\mu\leq\nu-1)$ is given for four corners $(x_0,y_0),(x_m,y_0),(x_0,y_n),(x_m,y_n)$.

Then, $f(x,y)$ is interpolated by the polynomial spline

$$S(x,y)=\sum_{\beta=-2\nu+1}^{n-1}\sum_{\alpha=-2\nu+1}^{m-1}c_{\alpha,\beta}N_{\alpha}(x;\Delta_x)N_{\beta}(y;\Delta_y) \tag{3}$$

at dual degree $2\nu-1$. $c_{\alpha,\beta}(-2\nu+1\leq\alpha\leq m-1),(-2\nu+1\leq\beta\leq n-1)$ is an interpolation coefficient. Also,

$$N_{\alpha}(x;\Delta_x)=(s_{\alpha+2\nu}-s_{\alpha})g_{2\nu}[s_{\alpha},s_{\alpha+1},\cdots,s_{\alpha+2\nu};x]$$

$$g_{2\nu}(s;x)=(s-x)_{+}^{2\nu-1}=\begin{cases}(s-x)^{2\nu-1} & (s\geq x)\\ 0 & (s<x)\end{cases} \tag{4}$$

$$s_{\alpha}=\begin{cases}x_0 & (-2\nu+1\leq\alpha\leq-1)\\ x_{\alpha} & (0\leq\alpha\leq m)\\ x_m & (m+1\leq\alpha\leq m+2\nu-1)\end{cases}$$

and

$$N_{\beta}(y;\Delta_y)=(t_{\beta+2\nu}-t_{\beta})g_{2\nu}[t_{\beta},t_{\beta+1},\cdots,t_{\beta+2\nu};y]$$

$$g_{2\nu}(t;y)=(t-y)_+^{2\nu-1}$$

$$t_\beta=\begin{cases} y_0 & (-2\nu+1\leq\beta\leq-1) \\ y_\beta & (0\leq\beta\leq n) \\ y_n & (n+1\leq\beta\leq n+2\nu-1) \end{cases} \qquad (5)$$

When interpolation condition (2) is applied to expression (3), the linear equations of order $(m+2\nu-1)\cdot(n+2\nu-1)$ which use interpolation coefficients $c_{\alpha,\beta}$ as unknown are obtained. By assigning the interpolation coefficients obtained by solving the equations to expression (3), interpolation values for arbitrary $a\leq x\leq b, c\leq y\leq d$ can be calculated.

Interpolation coefficients $c_{\alpha,\beta}(-2\nu+1\leq\alpha\leq m-1),(-2\nu+1\leq\beta\leq n-1)$ are calculated by subroutine DSCI1D and $S^{(\lambda,\mu)}(x,y)(-1\leq\lambda,\mu\leq2\nu-1)$ is used by subroutine DSFI1D so as to evaluate interpolation values. Here,

$$S^{(-1,-1)}(x,y)=\int_c^y\int_a^x S(x,y)dxdy$$

$$S^{(-1,\mu)}(x,y)=\int_a^x \frac{\partial^\mu S(x,y)}{\partial^\mu y}dx \qquad (0\leq\mu\leq2\nu-1)$$

$$S^{(\lambda,-1)}(x,y)=\int_c^y \frac{\partial^\lambda S(x,y)}{\partial x^\lambda}dy \qquad (0\leq\lambda\leq2\nu-1) \qquad (6)$$

$$S^{(\lambda,\mu)}(x,y)=\frac{\partial^{\lambda+\mu}S(x,y)}{\partial x^\lambda\partial y^\mu} \qquad (0\leq\lambda,\mu\leq2\nu-1)$$

2. (Type-Ⅱ)×(Type-Ⅱ) spline interpolation

$$\begin{cases} a=x_0<x_1<\cdots<x_m=b \\ c=y_0<y_1<\cdots<y_n=d \end{cases} \qquad (7)$$

is given. When the following is given for two-dimensional function $f(x,y)$:

(1) $f_{i,j}=f(x_i,y_j) \qquad (0\leq i\leq m),(0\leq j\leq n)$

(2) $f_{i,j}^{(\lambda,0)}=f^{(\lambda,0)}(x_i,y_j) \qquad (i=0,m),(0\leq j\leq n)$

(3) $f_{i,j}^{(0,\mu)}=f^{(0,\mu)}(x_i,y_j)$ $(j=0,n),(0\leq i\leq m)$ (8)

(4) $f_{i,j}^{(\lambda,\mu)}=f^{(\lambda,\mu)}(x_i,y_j)$ $(i=0,m),(j=0,n)$

$(\nu\leq\lambda\leq 2\nu-2),(\nu\leq\mu\leq 2\nu-2)$

That is, the following conditions are met:

(1) Function values are given for all grid points.

(2) Normal derivative $\partial^{\lambda}f/\partial x^{\lambda}(\nu\leq\lambda\leq 2\nu-2)$ from degree ν to degree $2\nu-2$ in the x direction is given on the grid points of $x=x_0=a, x=x_m=b$.

(3) Normal derivative $\partial^{\mu}f/\partial y^{\mu}(\nu\leq\mu\leq 2\nu-2)$ from degree ν to degree $2\nu-2$ in the y direction is given on the grid points of $y=y_0=c, y=y_n=d$.

(4) Partial derivative $\partial^{\lambda+\mu}/\partial x^{\lambda}\partial y^{\mu}(\nu\leq\lambda,\mu\leq 2\nu-2)$ is given at four corners $(x_0,y_0),(x_m,y_0),(x_0,y_n),(x_m,y_n)$.

Then, $f(x,y)$ is interpolated by polynomial spline

$$S(x,y)=\sum_{\beta=-2\nu+1}^{n-1}\sum_{\alpha=-2\nu+1}^{m-1}c_{\alpha,\beta}N_{\alpha}(x;\Delta_x)N_{\beta}(y;\Delta_y) \qquad (9)$$

of bi-$2\nu-1$ degree. $C_{\alpha,\beta}(-2\nu+1\leq\alpha\leq m-1),(-2\nu+1\leq\beta\leq n-1)$ are interpolation coefficients. $N_{\alpha}(x;\Delta_x),N_{\beta}(y;\Delta_y)$ are functions given by expressions (4) and (5). When interpolation conditions (8) are applied to expression (9), the linear equations of order $(m+2\nu-1)\cdot(n+2\nu-1)$ which use interpolation coefficients $C_{\alpha,\beta}$ as unknown are obtained. By assigning the interpolation coefficients obtained by solving the equations to expression (9), interpolation values for arbitrary $a\leq x\leq b, c\leq y\leq d$ can be calculated.

Interpolation coefficients $c_{\alpha,\beta}(-2\nu+1\leq\alpha\leq m-1),(-2\nu+1\leq\beta\leq n-1)$ are calculated by subroutine DSCI2D and $S^{(\lambda,\mu)}(x,y)(-1\leq\lambda,\mu\leq 2\nu-1)$ is used by subroutine DSFI2D so as to determine interpolation values. $S^{(\lambda,\mu)}(x,y)$ is given by expression (6).

3. (Type-Ⅲ)×(Type-Ⅲ) spline interpolation

$$\begin{cases}a=x_0<x_1<\cdots<x_m=b\\ c=y_0<y_1<\cdots<y_n=d\end{cases} \qquad (11)$$

is given. When values $f_{i,j}=f(x_i,y_j)(0\leq i\leq m),(0\leq j\leq n)$ on grid points of two-dimensional function $f(x,y)$ are given, $f(x,y)$ is interpolated by polynomial spline

$$S(x,y)=\sum_{\beta=-2\nu+1}^{n-2\nu+1}\sum_{\alpha=-2\nu+1}^{m-2\nu+1}c_{\alpha,\beta}N_\alpha(x;\Delta_x)N_\beta(y;\Delta_y) \qquad (12)$$

of bi-$2\nu-1$ degree. $c_{\alpha,\beta}(-2\nu+1\leq\alpha\leq m-2\nu+1),(-2\nu+1\leq\beta\leq n-2\nu+1)$ are interpolation coefficients. Also,

$$N_\alpha(x;\Delta_x)=(s_{\alpha+2\nu}-s_\alpha)g_{2\nu}[s_\alpha,s_{\alpha+1},\cdots,s_{\alpha+2\nu};x] \qquad (13)$$

$$g_{2\nu}(s;x)=(s-x)_+^{2\nu-1}$$

$$s_\alpha=\begin{cases}x_0 & (-2\nu+1\leq\alpha\leq 0)\\ x_{\alpha+\nu-1} & (1\leq\alpha\leq m-2\nu+1)\\ x_m & (m-2\nu+2\leq\alpha\leq m+1)\end{cases}$$

and

$$N_\beta(y;\Delta_y)=(t_{\beta+2\nu}-t_\beta)g_{2\nu}[t_\beta,t_{\beta+1},\cdots,t_{\beta+2\nu};x]$$

$$g_{2\nu}(t;y)=(t-y)_+^{2\nu-1}$$

$$t_\beta=\begin{cases}y_0 & (-2\nu+1\leq\beta\leq 0)\\ y_{\beta+\nu-1} & (1\leq\beta\leq n-2\nu+1)\\ y_n & (n-2\nu+2\leq\beta\leq n+1)\end{cases} \qquad (14)$$

When the interpolation conditions are applied to expression (12), linear equations

$$\sum_{\beta=-2\nu+1}^{n-2\nu+1}\sum_{\alpha=-2\nu+1}^{m-2\nu+1}c_{\alpha,\beta}N_\alpha(x_i;\Delta_x)N_\beta(y_j;\Delta_y)=f_{i,j}$$

$$(i=0,1,\cdots,m),(j=0,1,\cdots,n) \qquad (15)$$

of order $(m+1)\cdot(n+1)$ which use interpolation coefficients $c_{\alpha,\beta}$ as unknown are obtained. By assigning the interpolation coefficients obtained by solving the equations to expression (12), interpolation values for arbitrary $a\leq x\leq b, c\leq y\leq d$ can be calculated.

Interpolation coefficients $c_{\alpha,\beta}(-2\nu+1\leq\alpha\leq m-2\nu+1)\cdot(-2\nu+1\leq\beta\leq n-2\nu+1)$ are calculated by subroutine DSCI3D and $S^{(\lambda,\mu)}(x,y)(-1\leq\lambda,\mu\leq 2\nu-1)$ is calculated by subroutine DSFI3D so as to determine interpolation values. $S^{(\lambda,\mu)}(x,y)$ is given by expression (6).

4. (periodic)×(periodic) spline interpolation

$$\begin{cases} a=x_0<x_1<\cdots<x_m=b \\ c=y_0<y_1<\cdots<y_n=d \end{cases} \tag{16}$$

is given. Two-dimensional function $f(x,y)$ is supposed to be a periodic function with period $b-a$ for variable x and also a periodic function with period $d-c$ for variable y. When values $f_{i,j}=f(x_i,y_j)\,(0\leq i\leq m),\,(0\leq j\leq n)$ on the grid point of $f(x,y)$ are given, $f(x,y)$ is interpolated by the following polynomial splines at dual degree $2\nu-1$:

$$S(x,y)=\sum_{\beta=-2\nu+1}^{n-1}\sum_{\alpha=-2\nu+1}^{m-1} c_{\alpha,\beta}N_\alpha(x;\Delta_x)N_\beta(y;\Delta_y) \tag{17}$$

$$\begin{cases} c_{\alpha,\beta}=c_{\alpha+m,\beta} & (-2\nu+1\leq\alpha\leq-\nu) \\ c_{\alpha,\beta}=c_{\alpha-m,\beta} & (m-\nu+1\leq\alpha\leq m-1) \end{cases} \tag{18}$$

$(-2\nu+1\leq\alpha\leq m-1)$

$(-2\nu+1\leq\beta\leq n-1)$

$$\begin{cases} c_{\alpha,\beta}=c_{\alpha,\beta+n} & (-2\nu+1\leq\beta\leq-\nu) \\ c_{\alpha,\beta}=c_{\alpha,\beta-n} & (n-\nu+1\leq\beta\leq n-1) \end{cases} \tag{19}$$

$$N_\alpha(x;\Delta_x)=(s_{\alpha+2\nu}-s_\alpha)g_{2\nu}[\,s_\alpha,s_{\alpha+1},\cdots,s_{\alpha+2\nu};x\,]$$

$$g_{2\nu}(s;x)=(s-x)_+^{2\nu-1}$$

$$s_\alpha=\begin{cases} x_{m+\alpha}-(x_m-x_0) & (-2\nu+1\leq\alpha\leq-1) \\ x_\alpha & (0\leq\alpha\leq m) \\ x_{\alpha-m}+(x_m-x_0) & (m+1\leq\alpha\leq m+2\nu-1) \end{cases} \tag{20}$$

$$N_\beta(y;\Delta_y)=(t_{\beta+2\nu}-t_\beta)g_{2\nu}[\,t_\beta,t_{\beta+1},\cdots,t_{\beta+2\nu};y\,]$$

$$g_{2\nu}(t;y)=(t-y)_+^{2\nu-1}$$

$$t_\beta=\begin{cases} y_{n+\beta}-(y_n-y_0) & (-2\nu+1\leq\beta\leq-1) \\ y_\beta & (0\leq\beta\leq n) \\ y_{\beta-n}+(y_n-y_0) & (n+1\leq\beta\leq n+2\nu-1) \end{cases} \tag{21}$$

$c_{\alpha,\beta}(-2\nu+1\leq\alpha\leq m-1),(-2\nu+1\leq\beta\leq n-1)$ are interpolation coefficients. $S(x,y)$ given by expressions (17) to (21) can be considered as a periodic function in a sense that it satisfies

$$\begin{cases} S^{(\lambda,\mu)}(x_0,y)=S^{(\lambda,\mu)}(x_m,y) \\ S^{(\lambda,\mu)}(x,y_0)=S^{(\lambda,\mu)}(x,y_n) \end{cases} \quad (0\leq\lambda,\mu\leq 2\nu-1),(x_0\leq x\leq x_m),(y_0\leq y\leq y_n) \tag{22}$$

When interpolation conditions are applied to expression (17), linear equations at degree $m\cdot n$ which use interpolation coefficients $c_{\alpha,\beta}(-\nu+1\leq\alpha\leq m-\nu),(-\nu+1\leq\beta\leq n-\nu)$ as unknown are obtained. By assigning the interpolation coefficients obtained by solving the equations to expression (17), interpolation values for arbitrary $a\leq x\leq b, c\leq y\leq d$ can be calculated.

Interpolation coefficients $c_{\alpha,\beta}(-2\nu+1\leq\alpha\leq m-1),(-2\nu+1\leq\beta\leq n-1)$ are calculated by subroutine DSCI4D and $S^{(\lambda,\mu)}(x,y)(-1\leq\lambda,\mu\leq 2\nu-1)$ is calculated by subroutine DSFI4D so as to determine interpolation values. $S^{(\lambda,\mu)}(x,y)$ is given by expression (6).

5. (Type-I)×(periodic) spline interpolation

$$\begin{cases} a=x_0<x_1<\cdots<x_m=b \\ c=y_0<y_1<\cdots<y_n=d \end{cases} \tag{23}$$

is given. Two-dimensional function $f(x,y)$ is supposed to be a periodic function with period $d-c$ concerning variable y. For $f(x,y)$;

(1) $f_{i,j}=f(x_i,y_i) \quad (0\leq i\leq m),(0\leq j\leq n)$ (24)

(2) $f_{i,j}^{(\lambda,0)}=f^{(\lambda,0)}(x_i,y_j) \quad (1\leq\lambda\leq\nu-1),(i=0,m),(0\leq j\leq n)$

are given. That is, when;

(1) Function values are given on all grid points;

(2) Normal derivative $\partial^{\lambda}f/\partial x^{\lambda}(1\leq\lambda\leq\nu-1)$ up to degree $\nu-1$ in the x direction is given on the grid point on $x=x_0=a, x=x_m=b$;

$f(x,y)$ is interpolated by the following polynomial splines of bi-$2\nu-1$ degree:

$$S(x,y)=\sum_{\beta=-2\nu+1}^{n-1}\sum_{\alpha=-2\nu+1}^{m-1}c_{\alpha,\beta}N_{\alpha}(x;\Delta_x)N_{\beta}(y;\Delta_y) \tag{25}$$

$$\begin{cases} c_{\alpha,\beta}=c_{\alpha,\beta+n} & (-2\nu+1\leq\beta\leq-\nu) \\ c_{\alpha,\beta}=c_{\alpha,\beta-n} & (n-\nu+1\leq\beta\leq n-1) \end{cases} \tag{26}$$

$$(-2\nu+1\leq\alpha\leq m-1)$$

$c_{\alpha,\beta}(-2\nu+1\leq\alpha\leq m-1),(-2\nu+1\leq\beta\leq n-1)$ are interpolation coefficients. $N_{\alpha}(x;\Delta_x)$ is given by expression (4) and $N_{\beta}(y;\Delta_y)$ is given by expression (21). When interpolation condition (24) is applied to expression (25), linear equations of order $(m+2\nu-1)\cdot n$ which use

interpolation coefficients $c_{\alpha,\beta}(-2\nu+1\leq\alpha\leq m-1),(-\nu+1\leq\beta\leq n-\nu)$ as unknown are obtained. By assigning the interpolation coefficients obtained by solving the equations to expression (25), interpolation values for arbitrary $a\leq x\leq b, c\leq y\leq d$ can be calculated. Interpolation coefficients $c_{\alpha,\beta}(-2\nu+1\leq\alpha\leq m-1),(-2\nu+1\leq\beta\leq n-1)$ are calculated by subroutine DSCI5D and $S^{(\lambda,\mu)}(x,y)(-1\leq\lambda,\mu\leq 2\nu-1)$ is calculated by subroutine DSFI5D so as to determine interpolation values. $S^{(\lambda,\nu)}(x,y)$ is given by expression (6).

6. (Type-Ⅱ)×(periodic) spline interpolation

$$\begin{cases} a=x_0<x_1<\cdots<x_m=b \\ c=y_0<y_1<\cdots<y_n=d \end{cases} \tag{27}$$

is given. Two-dimensional function $f(x,y)$ is supposed to be a periodic function with period $d-c$ concerning variable y. For $f(x,y)$;

(1) $f_{i,j}=f(x_i,y_i)$ $(0\leq i\leq m),(0\leq j\leq n)$ (28)

(2) $f_{i,j}^{(\lambda,0)}=f^{(\lambda,0)}(x_i,y_j)$ $(1\leq\lambda\leq\nu-1),(i=0,m),(0\leq j\leq n)$

are given. That is, when;

(1) Function values are given on all grid points;

(2) Normal derivative $\partial^\lambda f/\partial x^\lambda(\nu\leq\lambda\leq 2\nu-1)$ from degree ν to degree $2\nu-2$ in the x direction is given on the grid point on $x=x_0=a, x=x_m=b$;

$f(x,y)$ is interpolated by the following polynomial splines of bi-$2\nu-1$ degree:

$$S(x,y)=\sum_{\beta=-2\nu+1}^{n-1}\sum_{\alpha=-2\nu+1}^{m-1}c_{\alpha,\beta}N_\alpha(x;\Delta_x)N_\beta(y;\Delta_y) \tag{29}$$

$$\begin{cases} c_{\alpha,\beta}=c_{\alpha,\beta+n} & (-2\nu+1\leq\beta\leq-\nu) \\ c_{\alpha,\beta}=c_{\alpha,\beta-n} & (n-\nu+1\leq\beta\leq n-1) \end{cases} \tag{30}$$

$(-2\nu+1\leq\alpha\leq m-1)$

$c_{\alpha,\beta}(-2\nu+1\leq\alpha\leq m-1),(-2\nu+1\leq\beta\leq n-1)$ are interpolation coefficients. $N_\alpha(x;\Delta_x)$ is given by expression (4) and $N_\beta(y;\Delta_y)$ is given by expression (21). When interpolation condition (28) is applied to expression (29), linear equations of order $(m+2\nu-1)\cdot n$ which use interpolation coefficients $c_{\alpha,\beta}(-2\nu+1\leq\alpha\leq m-1),(-\nu+1\leq\beta\leq n-\nu)$ as unknown are obtained. By assigning the interpolation coefficients obtained by solving the equations to expression (29), interpolation values for arbitrary $a\leq x\leq b, c\leq y\leq d$ can be calculated.

Interpolation coefficients $c_{\alpha,\beta}(-2\nu+1\leq\alpha\leq m-1),(-2\nu+1\leq\beta\leq n-1)$ are calculated by

subroutine DSCI6D and $S^{(\lambda,\mu)}(x,y)(-1\leq\lambda,\mu\leq 2\nu-1)$ is calculated by subroutine DSPI6D so as to determine interpolation values. $S^{(\lambda,\mu)}(x,y)$ is given by expression (6).

7 (Type-Ⅲ)×(periodic) spline interpolation

$$\begin{cases} a=x_0<x_1<\cdots<x_m=b \\ c=y_0<y_1<\cdots<y_n=d \end{cases} \tag{31}$$

is given. Two-dimensional function $f(x,y)$ is supposed to be a periodic function with period $d-c$ concerning variable y. When values $f_{i,j}=f(x_i,y_j)(0\leq i\leq m),(0\leq j\leq n)$ on the grid point of $f(x,y)$ are given, $f(x,y)$ is interpolated by the following polynomial splines of bi-$2\nu-1$ degree:

$$S(x,y)=\sum_{\beta=-2\nu+1}^{n-1}\sum_{\alpha=-2\nu+1}^{m-2N+1}c_{\alpha,\beta}N_\alpha(x;\Delta_x)N_\beta(y;\Delta_y) \tag{32}$$

$$\begin{cases} c_{\alpha,\beta}=c_{\alpha,\beta+n} & (-2\nu+1\leq\beta\leq-\nu) \\ c_{\alpha,\beta}=c_{\alpha,\beta-n} & (n-\nu+1\leq\beta\leq n-1) \end{cases} \quad (-2\nu+1\leq\alpha\leq m-2\nu+1) \tag{33}$$

$c_{\alpha,\beta}(-2\nu+1\leq\alpha\leq m-2\nu+1),(-2\nu+1\leq\beta\leq n-1)$ are interpolation coefficients. $N_\alpha(x;\Delta_x)$ is given by expression (13) and $N_\beta(y;\Delta_y)$ is given by expression (21). When interpolation conditions are applied to expression (32), linear equations of order $(m+1)\cdot n$ which use interpolation coefficients $c_{\alpha,\beta}(-2\nu+1\leq\alpha\leq m-2\nu+1),(-\nu+1\leq\beta\leq n-\nu)$ as unknown are obtained. By assigning the interpolation coefficients obtained by solving the equations to expression (32), interpolation values for arbitrary $a\leq x\leq b, c\leq y\leq d$ can be calculated.

Interpolation coefficients $c_{\alpha,\beta}(-2\nu+1\leq\alpha\leq m-2\nu+1),(-2\nu+1\leq\beta\leq n-1)$ are calculated by subroutine DSCI7D and $S^{(\lambda,\mu)}(x,y)(-1\leq\lambda,\mu\leq 2\nu-1)$ is calculated by subroutine DSPI7D so as to determine interpolation values. $S^{(\lambda,\mu)}(x,y)$ is given by expression (6).

(2) Directions

```
CALL DSCI1D(XI, YJ, F, CAB, NX, NY, M, WORKC, NXM2D)
CALL DSPI1D(XP, YP, IX, IY, LX, LY, FP, NX, NY, M, XI, YJ, CAB, WORKP, NXM2D)
```

Argument	Type and kind	Attribute	Content
XI	Double precision real type One-dimensional array	Input	Grid point x_i in the x direction. Array of size $m+1$. $x_i(0 \leq i \leq m)$ is put in $XI(i+1)$.
YJ	Double precision real type One-dimensional array	Input	Grid point y_j in the y direction. Array of size $n+1$ $y_j(0 \leq j \leq m)$ is put in $YJ(j+1)$.
F	Double precision real type Two-dimensional array	Input	$\overline{f}_{i,j}$ like function values in grid point. Two-dimensional array of size $(m+2v-1) \times (n+2v-1)$ $\overline{f}_{i,j}(1 \leq i \leq m+2v-1), (1 \leq j \leq n+2v-1)$ are put in $F(i,j)$.
CAB	Double precision real type Two-dimensional array	Input/output	Output for DSCI1D. Input for DSFI1D. Interpolation coefficients $c_{\alpha,\beta}(-2v+1 \leq \alpha \leq m-1), (-2v+1 \leq \beta \leq n-1)$ Two-dimensional array of size $(m+2v-1) \times (n+2v-1)$. $c_{\alpha,\beta}$ is put in CAB$(\alpha+2v, \beta+2v)$.
NX	Integer type	Input	The number of grid squares m in the x direction is put.
NY	Integer type	Input	The number of grid squares n in the y direction is put.
M	Integer type	Input	v in order $2v-1$ of spline is put.
WORKC	Double precision real type One-dimensional array	Input/output	Work area. The size is $(k-1)(2v-1)+2v^2+6v+2k-2$ as $k=_{max}(m,n)$.

Argument	Type and kind	Attribute	Content
NXM2D	Integer type	Input	Size of adjustable array. Size of the first subscript of F and CAB. NXM2D≧$m+2v-1$ must be satisfied.
XP, YP	Double precision real type	Input	Point (x,y) at which we want to evaluate interpolation values or other values. x is put in XP and y is put in YP. XI(1) ≦XP≦XI(NX+1) and YJ(1) ≦YP≦YJ(NY+1) must be satisfied. If XP and YP outside this range are given, an error message is printed and FP is set to 0.0.
IX, IY	Integer type	Input	Integers IX and IY which respectively satisfy XI(IX+1) ≦XP≦XI(IX+2) and YJ(IY+1) ≦YP≦YJ(IY+2) are put. Even if IX and IY do not satisfy the above requirements, calculation is performed normally but it takes a little more time than usual because of the need for search.
LX, LY	Integer type	Input	Integer which satisfies -1≦LX and LY≦$2v-1$. A kind of calculation is given. That is, λ in quantity $S^{(\lambda,\mu)}(x,y)$ to be evaluated is put in LX and μ is put in LY.
FP	Double precision real type	Output	The calculation result of $S^{(\lambda,\mu)}(x,y)$ such as for an interpolation value is generated.
WORKF	Double precision real type One-dimensional array	Input/output	Work area. The size is $m+6v-1$.

To simplify the explanation of the syntax, $\overline{f}_{i,j}(1\leq i\leq m+2v-1),(1\leq j\leq n+2v-1)$ are defined for interpolated function $f(x,y)$ as follows:

(1) $\overline{f}_{i,j}=f^{(\nu-i,\nu-j)}(x_0,y_0)$ $(1\leq i,j\leq\nu-1)$

(2) $\overline{f}_{i,j}=f^{(0,\nu-j)}(x_{i-\nu},y_0)$ $(\nu\leq i\leq m+\nu),(1\leq j\leq\nu-1)$

(3) $\overline{f}_{i,j}=f^{(i-m-\nu,\nu-j)}(x_m,y_0)$ $(m+\nu+1\leq i\leq m+2\nu-1),(1\leq j\leq\nu-1)$

(4) $\overline{f}_{i,j}=f^{(\nu-i,0)}(x_0,y_{j-\nu})$ $(1\leq i\leq\nu-1),(\nu\leq j\leq n+\nu)$

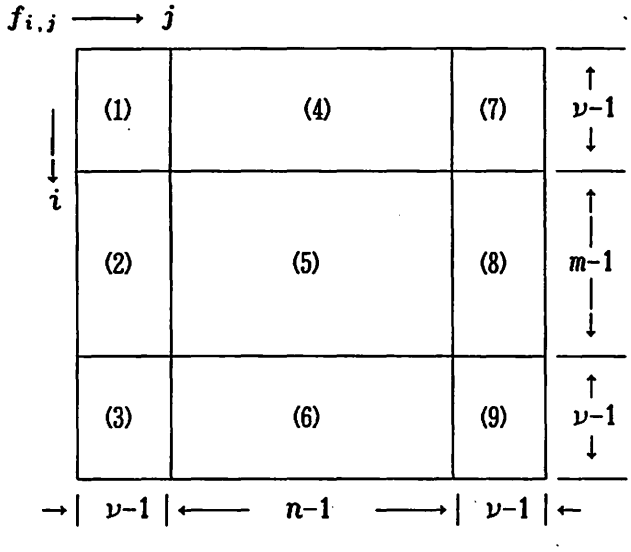
(5) $\overline{f}_{i,j}=f(x_{i-\nu},y_{j-\nu})$ $(\nu\leq i\leq m+\nu),(\nu\leq j\leq n+\nu)$

(6) $\overline{f}_{i,j}=f^{(i-m-\nu,0)}(x_m,y_{j-\nu})$ $(m+\nu+1\leq i\leq m+2\nu-1),(\nu\leq j\leq n+\nu)$

(7) $\overline{f}_{i,j}=f^{(\nu-i,j-n-\nu)}(x_0,y_n)$ $(1\leq i\leq\nu-1),(n+\nu+1\leq j\leq n+2\nu-1)$

(8) $\overline{f}_{i,j}=f^{(0,j-n-\nu)}(x_{i-\nu},y_n)$ $(\nu\leq i\leq m+\nu),(n+\nu+1\leq j\leq n+2\nu-1)$

(9) $\overline{f}_{i,j}=f^{(i-m-\nu,j-n-\nu)}(x_m,y_n)$ $(m+\nu+1\leq i\leq m+2\nu-1),(n+\nu+1\leq j\leq n+2\nu-1)$



For instance, the list of $\overline{f}_{i,j}$ is as follows when $\nu=2$, $m=2$, $n=3$.

j=1 to 6

	$f_{00}^{(1,1)}$	$f_{00}^{(1,0)}$	$f_{01}^{(1,0)}$	$f_{02}^{(1,0)}$	$f_{03}^{(1,0)}$	$f_{03}^{(1,1)}$
	$f_{00}^{(0,1)}$	f_{00}	f_{01}	f_{02}	f_{03}	$f_{03}^{(0,1)}$
i=1 to 5	$f_{10}^{(0,1)}$	f_{10}	f_{11}	f_{12}	f_{13}	$f_{13}^{(0,1)}$
	$f_{20}^{(0,1)}$	f_{20}	f_{21}	f_{22}	f_{23}	$f_{23}^{(0,1)}$
	$f_{20}^{(1,1)}$	$f_{20}^{(1,0)}$	$f_{21}^{(1,0)}$	$f_{22}^{(1,0)}$	$f_{23}^{(1,0)}$	$f_{23}^{(1,1)}$

CALL DSCI2D(XI, YJ, F, CAB, NX, NY, M, WORKC, NXM2D)

CALL DSPI2D(XP, YP, IX, IY, LX, LY, PP, NX, NY, M, XI, YJ, CAB, WORKP, NXM2D)

Argument	Type and kind	Attribute	Content
P	Double precision real type Two-dimensional array	Input	Function values etc. at grid points $f_{i,j}$ (quantity given by expression (10)). Two-dimensional array of size $(m+2\nu-1)\times(n+2\nu-1)$. $\overline{f}_{i,j}(1\leq i\leq m+2\nu-1),(1\leq j\leq n+2\nu-1)$ are put in $F(i,j)$.
WORKC	Double precision real type One-dimensional array	Input/output	Work area. The size is $(k+2\nu-3)(2\nu-1)+2\nu^2+6\nu+2k-2$ as $k=_{max}(m,n)$.
The other arguments are the same as for the (Type-I)×(Type-I) spline. (However, CAB is input for DSCI2D and output for DSPI2D.)			

To simplify the explanation of the syntax above, $\overline{f}_{i,j}(1\leq i\leq m+2\nu-1)(1\leq j\leq n+2\nu-1)$ are defined for interpolated function $f(x,y)$ as follows:

(1) $\overline{f}_{i,j}=f^{(2\nu-1-i,2\nu-1-j)}(x_0,y_0)$ $(1\leq i,j\leq\nu-1)$

(2) $\overline{f}_{i,j}=f^{(0,2\nu-1-j)}(x_{i-\nu},y_0)$ $(\nu\leq i\leq m+\nu),(1\leq j\leq\nu-1)$

(3) $\overline{f}_{i,j}=f^{(i-m-1,2\nu-1-j)}(x_m,y_0)$ $(m+\nu+1\leq i\leq m+2\nu-1),(1\leq j\leq\nu-1)$

(4) $\overline{f}_{i,j}=f^{(2\nu-1-i,0)}(x_0,y_{j-\nu})$ $(1\leq i\leq\nu-1),(\nu\leq j\leq n+\nu)$

(5) $\overline{f}_{i,j}=f(x_{i-\nu},y_{j-\nu})$ $(\nu\leq i\leq m+\nu),(\nu\leq j\leq n+\nu)$

(6) $\overline{f}_{i,j}=f^{(i-m-1,0)}(x_m,y_{j-\nu})$ $(m+\nu\leq i\leq m+2\nu-1),(\nu\leq j\leq n+\nu)$

(7) $\overline{f}_{i,j}=f^{(2\nu-1-i,j-n-1)}(x_0,y_n)$ $(1\leq i\leq\nu-1),(n+\nu\leq j\leq n+2\nu-1)$

(8) $\overline{f}_{i,j}=f^{(0,j-n-1)}(x_{i-\nu},y_n)$ $(\nu\leq i\leq m+\nu),(n+\nu\leq j\leq n+2\nu-1)$

(9) $\overline{f}_{i,j}=f^{(i-m-1,j-n-1)}(x_m,y_n)$ $(m+\nu\leq i\leq m+2\nu-1),(n+\nu\leq j\leq n+2\nu-1)$

For instance, the list of $\overline{f}_{i,j}$ is as follows when $\nu=2, m=2, n=3$.

	j=1 to 6					
	$f_{00}^{(2,2)}$	$f_{00}^{(2,0)}$	$f_{01}^{(2,0)}$	$f_{02}^{(2,0)}$	$f_{03}^{(2,0)}$	$f_{03}^{(2,2)}$
	$f_{00}^{(0,2)}$	f_{00}	f_{01}	f_{02}	f_{03}	$f_{03}^{(0,2)}$
i=1 to 5	$f_{10}^{(0,2)}$	f_{10}	f_{11}	f_{12}	f_{13}	$f_{13}^{(0,2)}$
	$f_{20}^{(0,2)}$	f_{20}	f_{21}	f_{22}	f_{23}	$f_{23}^{(0,2)}$
	$f_{20}^{(2,2)}$	$f_{20}^{(2,0)}$	$f_{21}^{(2,0)}$	$f_{22}^{(2,0)}$	$f_{23}^{(2,0)}$	$f_{23}^{(2,2)}$

CALL DSCI3D(XI, YJ, F, CAB, XY3O, NX, NY, M, WORKC, NXP1D)

CALL DSFI3D(XP, YP, IX, IY, LX, LY, FP, NX, NY, M, XI, YJ, CAB, XY3O, WORKF, NXP1D)

Argument	Type and kind	Attribute	Content
F	Double precision real type Two-dimensional array	Input	Function value $f_{i,j}$ in grid point. Two-dimensional array of size $(m+1)\times(n+1)$. $f_{i,j}(0\leq i\leq m),(0\leq j\leq n)$ are put in $F(i+1,j+1)$.
CAB	Double precision real type Two-dimensional array	Input/output	Output for DSCI3D. Input for DSFI3D. Interpolation coefficients $c_{\alpha,\beta}(-2v+1\leq\alpha\leq m-2v+1),(-2v+1\leq\beta\leq n-2v+1)$. Two-dimensional array of size $(m+1)\times(n+1)$. $c_{\alpha,\beta}$ is put in $CAB(\alpha+2v,\beta+2v)$.
XY3O	Double precision real type One-dimensional array	Input/output	Output for DSCI3D. Input for DSFI3D. Spline knots $x_0,x_v,x_{v+1},\cdots,x_{m-v},x_m;y_0,y_v,y_{v+1},\cdots,y_n$ are put. Array of size $m+n-4v+6$
WORKC	Double precision real type One-dimensional array	Input/output	Work area. The size is $(k-1)(2v-1)+4v+2(k+1)$ as $k=_{max}(m,n)$ is assumed.
NXP1D	Integer type	Input	Size of adjustable array. Size of the first subscript of F and CAB. $NXP1D\geq m+1$ must be satisfied.
The other arguments are the same as for the (Type-I)×(Type-I) spline.			

CALL DSCI4D(XI, YJ, F, CAB, NX, NY, M, WORKC, NXP1D, NXM2D)

CALL DSFI4D(XP, YP, IX, IY, LX, LY, FP, NX, NY, M, XI, YJ, CAB, WORKF, NXM2D)

Argument	Type and kind	Attribute	Content
F	Double precision real type Two-dimensional array	Input	Function value $f_{i,j}$ in grid point. Two-dimensional array of size $(m+1)\times(n+1)$. $f_{i,j}(0\leq i\leq m),(0\leq j\leq n)$ are put in $F(i+1,j+1)$.
CAB	Double precision real type Two-dimensional array	Input/output	Output for DSCI4D. Input for DSFI4D. Interpolation coefficient $c_{\alpha,\beta}(-2v+1\leq\alpha\leq m-1),(-2v+1\leq\beta\leq n-1)$. Two-dimensional array of size $(m+2v-1)\times(n+2v-1)$. $c_{\alpha,\beta}$ is put in $(\alpha+2v,\beta+2v)$.
WORKC	Double precision real type One-dimensional array	Input/output	Work area. The size is $k(4v-1)+4v$ as $k=_{max}(m,n)$ is assumed.
NXP1D	Integer type	Input	Size of adjustable array. Size of the first subscript of array F. NXP1D$\geq m+1$ must be satisfied.
NXM2D	Integer type	Input	Size of adjustable array. Size of the first subscript of array CAB. NXM2D$\geq m+2v-1$ must be satisfied.
The other arguments are the same as for the (Type-I)×(Type-I) spline.			

```
CALL DSCI5D(XI, YJ, F, CAB, NX, NY, M, WORKC, NXM2D)

CALL DSFI5D(XP, YP, IX, IY, LX, LY, FP, NX, NY, M, XI, YJ, CAB, WORKF, NXM2D)
```

Argument	Type and kind	Attribute	Content
F	Double precision real type Two-dimensional array	Input	Function value etc. $f_{i,j}(1\leq i\leq m+2v-1),(1\leq j\leq n+1)$ in grid point. Two-dimensional array of size $(m+2v-1)\times(n+1)$. $\overline{f}_{i,j}$ are put in $F(i,j)$.
CAB	Double precision real type Two-dimensional array	Input/output	Output for DSCI5D. Input for DSFI5D. Interpolation coefficient $c_{\alpha,\beta}(-2v+1\leq\alpha\leq m-1),(-2v+1\leq\beta\leq n-1)$. Two-dimensional array of size $(m+2v-1)\times(n+2v-1)$. $c_{\alpha,\beta}$ is put in CAB$(\alpha+2v,\beta+2v)$.
WORKC	Double precision real type One-dimensional array	Input/output	Work area. The size is a larger one of k_1,k_2 given by the following expression: $k_1=(m-1)(2v-1)+2v^2+6v+2m-2$, $k_2=n(4v-1)+4v$.
The other arguments are the same as for the (Type-I)×(Type-I) spline.			

To simplify the explanation of the syntax above, $\overline{f}_{i,j}(1\leq i\leq m+2v-1),(1\leq j\leq n+1)$ are defined for interpolated function $f(x,y)$ as follows:

(1) $\overline{f}_{i,j}=f^{(v-i,0)}(x_0,y_{j-1})$ $\quad(1\leq i\leq v-1)$

(2) $\overline{f}(x_{i-v},y_{j-1})$ $\quad(v\leq i\leq m+v)$

(3) $\overline{f}_{i,j}=f^{(i-m-v,0)}(x_m,y_{j-1})$ $\quad(m+v+1\leq i\leq m+2v-1)$

$(1\leq j\leq n+1)$

For instance, the list of $f_{i,j}$ is as follows when $\nu=2$, $m=2$, $n=3$.

	j=1 to 4			
	$f_{00}^{(1,2)}$	$f_{01}^{(1,0)}$	$f_{02}^{(1,0)}$	$f_{03}^{(1,0)}$
	f_{00}	f_{01}	f_{02}	f_{03}
i=1 to 5	f_{10}	f_{11}	f_{12}	f_{13}
	f_{20}	f_{21}	f_{22}	f_{23}
	$f_{20}^{(1,0)}$	$f_{21}^{(1,0)}$	$f_{22}^{(1,0)}$	$f_{23}^{(1,0)}$

CALL DSCI6D(XI, YJ, F, CAB, NX, NY, M, WORKC, NXM2D)

CALL DSFI6D(XP, YP, IX, IY, LX, LY, FP, NX, NY, M, XI, YJ, CAB, WORKF, NXM2D)

Argument	Type and kind	Attribute	Content
F	Double precision real type Two-dimensional array	Input	Function value etc. $\overline{f}_{i,j}(1\leq i\leq m+\nu-1),(1\leq j\leq n+1)$ in grid point. Two-dimensional array of size $(m+2\nu-1)\times(n+1)$. $\overline{f}_{i,j}$ are put in $F(i,j)$.
CAB	Double precision real type Two-dimensional array	Input/output	Output for DSCI6D. Input for DSFI6D. Interpolation coefficients $c_{\alpha,\beta}(-2\nu+1\leq\alpha\leq m-1)(-2\nu+1\leq\beta\leq n-1)$. Two-dimensional array of size $(m+2\nu-1)\times(n+2\nu-1)$. $c_{\alpha,\beta}$ is put in CAB$(\alpha+2\nu,\beta+2\nu)$.
WORKC	Double precision real type One-dimensional array	Input/output	Work area. The size is a larger one of k_1,k_2 given by the following expression: $k_1=(m+2\nu-3)(2\nu-1)+2\nu^2+6\nu+2m-2$, $k_2=n(4\nu-1)+4\nu$

Argument	Type and kind	Attrib ute	Content
The other arguments are the same as for the (Type-I)×(Type-I) spline.			

To simplify the explanation of the syntax above, $f(x,y)$ are defined for interpolated function $\overline{f}_{i,j}(1\leq i\leq m+2\nu-1),(1\leq j\leq n+1)$ as follows:

(1) $\overline{f}_{i,j}=f^{(2\nu-1-i,0)}(x_0,y_{j-1})$ $\quad(1\leq i\leq \nu-1)$

(2) $\overline{f}_{i,j}=f(x_{i-\nu},y_{j-1})$ $\quad(\nu\leq i\leq m+\nu)$

(3) $\overline{f}_{i,j}=f^{(i-m-1,0)}(x_m,y_{j-1})$ $\quad(m+\nu+1\leq i\leq m+2\nu-1)$

$\quad(1\leq j\leq n+1)$

For instance, the list of $\overline{f}_{i,j}$ is as follows when $\nu=2$, $m=2$, $n=3$.

	j=1 to 4			
	$f_{00}^{(2,2)}$	$f_{01}^{(2,0)}$	$f_{02}^{(2,0)}$	$f_{03}^{(2,0)}$
	f_{00}	f_{01}	f_{02}	f_{03}
i=1 to 5	f_{10}	f_{11}	f_{12}	f_{13}
	f_{20}	f_{21}	f_{22}	f_{23}
	$f_{20}^{(2,0)}$	$f_{21}^{(2,0)}$	$f_{22}^{(2,0)}$	$f_{23}^{(2,0)}$

```
CALL DSCI7D(XI, YJ, F, CAB, X30, NX, NY, M, WORKC, NXP1D)

CALL DSFI7D(XP, YP, IX, IY, LX, LY, FP, NX, NY, M, XI, YJ, CAB, X30, WORKF, NXP1D)
```

Argument	Type and kind	Attribute	Content
F	Double precision real type Two-dimensional array	Input	Function value $f_{i,j}(0\leq i\leq m),(0\leq j\leq n)$ in grid point. Two-dimensional array of size $(m+1)\times(n+1)$. $f_{i,j}$ is put in $F(i+1,j+1)$.
CAB	Double precision real type Two-dimensional array	Input/output	Output for DSCI7D. Input for DSFI7D. Interpolation coefficient $c_{\alpha,\beta}(-2\upsilon+1\leq\alpha\leq m-2\upsilon+1),(2\upsilon+1\leq\beta\leq n-1)$. Two-dimensional array of size $(m+1)\times(n+2\upsilon-1)$. $c_{\alpha,\beta}$ is put in CAB$(\alpha+2\upsilon,\beta+2\upsilon)$.
X3O	Double precision real type One-dimensional array	Input/output	Output for DSCI7D. Input for DSFI7D. Spline knots $x_0,x_\upsilon,x_{\upsilon+1},\cdots,x_{m-\upsilon},x_m$ are put. Array of size $m-2\upsilon+3$
WORKC	Double precision real type One-dimensional array	Input/output	Work area. The size is either one of k_1,k_2, whichever is greater, given by the following expression: $k_1=(m-1)(2\upsilon-1)+4\upsilon+2(m+1)$, $k_2=n(4\upsilon-1)+4\upsilon$
NXP1D	Integer type		Size of adjustable array. Size of the first subscript of F and CAB. NXP1D$\geq m+1$ must be satisfied.
The other arguments are the same as for the (Type-I)×(Type-I) spline.			

(3) Notes

1. If partial derivatives up to degree $\upsilon-1$ can be given at the boundary, it is better to use DSCI1D or DSFI1D. Of the seven types, these subroutines can be expected to show the highest precision.

2. From the viewpoint that interpolation can be done by using only function values on grid points, DSCI3D and DSFI3D are the most effective.

3. DSCI4D and DSFI4D are effective for interpolation of a function which has periodicity in both x and y directions.

4. If the partial derivatives of $f(x,y)$ which should be given at the boundary are all set to 0, they can be obtained by an interpolation formula for which a one-dimensional natural spline has been extended to a two-dimensional spline.

5. DSCI5D and DSFI5D can be used when the function value of two-dimensional function $z=f(r,\theta)(0\leq a\leq r\leq b),(0\leq\theta\leq 2\pi)$ defined by a cylindrical coordinates system is given on the grid point and the partial derivatives up to degree $\nu-1$ in the r direction are given by $r=a,r=b$.

6. DSCI6D and DSFI6D can be used when the function value of two-dimensional function $z=f(r,\theta)(0\leq a\leq r\leq b),(0\leq\theta\leq 2\pi)$ defined by a cylindrical coordinate system is given on the grid point and the partial derivatives of degrees from ν to $2\nu-2$ in the r are given by $r=a,r=b$.

7. DSCI7D and DSFI7D can be used when the function value of two-dimensional function $z=f(r,\theta)(0\leq a\leq r\leq b),(0\leq\theta\leq 2\pi)$ defined by a cylindrical coordinate system is given on the grid point.

(1987.06.15)

HERM31 and HERM51 (Curve Fitting by the Piecewise Hermite Interpolation Formula (3, 5-Degrees)

Curve Fitting by the Piecewise Hermite Interpolation Formula (3, 5-Degrees)

Programmed by	Yasuyo Hatano, June 1976
Format	Subroutine language: FORTRAN; size: 151 and 175 lines respectively

(1) Outline

HERM31 and HERM51 obtain the function value $y=\overline{f}(x)$ at an arbitrary point x and the differential coefficient using the function value $f_i=f(x_i)(i=1,n)$ given at the discrete point x_i . The interpolation is based on the piecewise Hermitian interpolation of degree 3(HERM31) or 5(HERM51).

(2) Directions

CALL HERM31(I, X, Y, M, N, XI, YI, YD, ND, ILL)

CALL HERM51(I, X, Y, M, N, XI, YI, YD, ND, ILL)

Argument	Type and kind	Attribute	Content
I	Integer type	Input	Value of the number i of mesh points in the range of $x_i \leqq x \leqq x_{i+1}$. $1 \leqq I < N$
X	Real type	Input	x coordinates of points where interpolation values are to be obtained. $XI(1) \leqq XI(I) \leqq X \leqq XI(I+1) \leqq XI(N)$
Y	Real type One-dimensional array	Output	Name of one-dimensional array containing (M+1) elements. Interpolation value y of a function at x and differential coefficient in that point. Real type variable name can be also used at M=0.
M	Integer type	Input	The highest order of differential coefficients to be obtained (function value only at 0). $0 \leqq M \leqq 1$ for HERM31, and $0 \leqq M \leqq 2$ for HERM51.
N	Integer type	Input	Total of input data x_i. $2 \leqq N$.
XI	Real type One-dimensional array	Input	Name of one-dimensional array containing N elements. Value of discrete point x_i. $XI(1) < XI(2) < \cdots < XI(N)$

Argument	Type and kind	Attribute	Content
YI	Real type One-dimensional array	Input	Name of one-dimensional array containing N elements. Function value $f_i(i=1,\cdots N)$ at x_i.
YD	Real type One-dimensional array (HERM31) Or Real type Two-dimensional array (HERM51)	Input/output	(1) HERM31: Name of one-dimensional array containing N elements. In an input meaning, the first order differential coefficient at the discrete points XI(I) and XI(I+1) should be input to YD(I) and YD(I+1) respectively. At this time, ILL=0 must be specified. If differential coefficients at discrete points are unknown, the output becomes valid. If ILL≠0 is specified, the approximate value of the first order differential coefficient at all the points (N points) is output. (2) HERM51: Name of two-dimensional array containing ND elements. As an input, the first and second order differential coefficients at discrete points are specified. The first order differential coefficient at XI(I) should be input to YD(I,1), and the second order differential coefficient at XI(I) should be input to YD(I,2). The first order differential coefficient at XI(I+1) should be input to YD(I+1,1), and the second order differential coefficient at XI(I+1) should be input to YD(I+1,2). At this time, ILL=0 must be specified. If the differential coefficient is unknown, the output becomes valid if ILL≠0 is specified, and the approximate value of the first and second order differential coefficients at all the points (N points) is output.
ND	Integer type	Input	Value of the first subscript in the array declaration of YD. N≦ND
ILL	Integer type	Input/output	The input means as follows: If the differential coefficients at each point (the first order coefficient for HERM31, and the first and second order coefficients for HERM51) are already known, those values should be input to YD with ILL=0. If the coefficients are unknown, the approximate values are obtained and output to YD using piecewise Lagrange interpolation in this routine, so the degree of interpolation formulas to be used must be specified. If ILL=1, the linear polynomial is used. If ILL=2, the quadratic polynomial is used. If ILL≧3, the quartic polynomial is used. The output means as follows: If ILL=0, normal termination is assumed. If ILL=30000, no calculation was made at all because limits on the argument were exceeded.

(3) Note

Even if differential coefficients at discrete points are unknown when two or more interpolation values are to be repeatedly obtained, the calculation time is the same as with ILL=0 because ILL=0 is automatically assigned if ILL≠0 is specified for the first time only.

Bibliography

1) A. Ralston; "A First Course in Numerical Analysis", McGraw-Hill, p. 42 and 60 (1965).

(1987. 08. 10)

HERM32 and HERM52 (Surface Fitting by the Piecewise Hermite Interpolation Formula (3, 5-degrees))

Surface Fitting by the Piecewise Hermite Interpolation Formula (3, 5-Degrees)

Programmed by	Yasuyo Hatano, April 1977
Format	Subroutine language: FORTRAN; size: 190 and 242 lines respectively

(1) Outline

HERM32 and HERM52 obtain the function value $z=\bar{f}(x,y)$ and differential coefficients at an arbitrary point (x,y) using the function value $f_{ij}=f(x_i,y_i)$, $(i=1,\cdots,n_x,\ j=1,\cdots,n_y)$ given at the rectangular mesh point (x_i,y_i). The interpolation is based on the piecewise Hermitian interpolation of degree 3(HERM32) or 5(HERM52).

(2) Directions

CALL HERM32/52(IX, X, JY, Y, Z, M, XI, NX, YI, NY, F, N1, N2, ILL)

Argument	Type and kind	Attribute	Content
IX	Integer type	Input	Value of the number i of the $x_i \leqq x \leqq x_{i+1}$ mesh points in reference to the x coordinates of points where interpolation values are to be obtained. $1 \leqq IX < NX$
X	Real type	Input	x coordinates of points where interpolation values are to be obtained. $XI(1) \leqq XI(IX) \leqq X \leqq XI(IX+1) \leqq XI(NX)$
JY	Integer type	Input	Value of the number j of the $y_i \leqq y \leqq y_{j+1}$ mesh points in reference to the y coordinates of points where interpolation values are to be obtained. $1 \leqq JY < NY$

Argument	Type and kind	Attribute	Content
Y	Real type	Input	y coordinates of points where interpolation values are to be obtained. YI(1)≦YI(JY)≦Y≦YI(JY+1)≦YI(NY)
Z	Real type One-dimensional array	Output	Name of one-dimensional array containing $(M+1)^2$ elements. Interpolation value and differential coefficient at the point (X, Y). For instance, if M=1, the values are output in the order that is shown in the expression (1). If M=0, even a real type variable name can be used.
M	Integer type	Input	The maximum order of differential coefficients to be obtained (only function value at 0). 0≦M≦1 at HERM32. 0≦M≦2 at HERM52.
XI	Real type One-dimensional array	Input	Name of one-dimensional array containing NX elements. x coordinates at mesh points. XI(1)<XI(2)<…<XI(NX)
NX	Integer type	Input	Total number of mesh points in x direction. 2≦NX
YI	Real type One-dimensional array	Input	Name of one-dimensional array containing NY elements. y coordinates at mesh points. YI(1)<YI(2)<…<YI(NY)
NY	Integer type	Input	Total number of mesh points in y direction. 2≦NY

Argument	Type and kind	Attribute	Content
F	Real type Three-dimensional array	Input/output	(1) HERM32: Name of three-dimensional array containing N1×N2×4 elements. As an input, the function value f_{ij} at mesh points is specified. The function value $f_{I,J}$ at mesh points (X(I), Y(J)) should be specified for F(I, J, 1) (I=1, ···, NX, J=1, ···, NY). If ILL=0 is specified, data should be input to F(I, J, K) (K=2, 3, 4) individually in the order shown in expression (2). Retained. When the differential coefficients are unknown, if ILL≠0 is specified, F(I, J, K) (I=1, ···, NX, J=1, ···, NY, K=2, 3, 4) has the meaning as an output variable, and the approximate value of the differential coefficient is output. (2) HERM52: Name of three-dimensional array containing N1×N2×9 elements. As an input, the function value f_{ij} at mesh points is specified. If ILL=0 is specified, data should be input to F(I, J, K) (K=2, ···, 9) individually in the order shown in expression (3). Retained. When the differential coefficients are unknown, if ILL≠0 is specified, F(I, J, K) (I=1, ···, NX, J=1, ···, NY, K=2, ···, 9) has the meaning as an output variable, and the approximate value of the differential coefficients is output.
N1, N2	Integer type	Input	The first and second subscripts in the array declaration of F. NX≦N1, NY≦N2

Argument	Type and kind	Attribute	Content
ILL	Integer type	Input/output	If ILL=0, differential coefficients at the corresponding points should be input to F as an input. When these differential coefficients are unknown, the approximate values are obtained using piecewise Lagrange interpolation in this routine, and output to F, so the degree of the interpolation formula to be used must be specified. If ILL=1, the linear polynomial is used. If ILL=2, the quadratic polynomial is used. If ILL≧3, the quartic polynomial is used. ILL≦NX, ILL≦NY The meaning of output is as follows: If ILL=0, normal termination is assumed. If ILL=30000, no calculation was executed because limits on the argument were exceeded.

$$Z(1)=\overline{f},\ Z(2)=\left[\frac{\partial \overline{f}}{\partial x}\right]_{x=X,y=Y},\ Z(3)=\left[\frac{\partial \overline{f}}{\partial y}\right]_{x=X,y=Y},\ Z(4)=\left[\frac{\partial^2 \overline{f}}{\partial x \partial y}\right]_{x=X,y=Y} \tag{1}$$

$$\left[\frac{\partial f}{\partial x},\ \frac{\partial f}{\partial y},\ \frac{\partial^2 f}{\partial x \partial y}\right]_{x=XI(I),y=YI(J)} \tag{2}$$

$$\left[\frac{\partial f}{\partial x},\ \frac{\partial^2 f}{\partial x^2},\ \frac{\partial f}{\partial y},\ \frac{\partial^2 f}{\partial x \partial y},\ \frac{\partial^3 f}{\partial x^2 \partial y},\ \frac{\partial^2 f}{\partial y^2},\ \frac{\partial^3 f}{\partial x \partial^2 y},\ \frac{\partial^4 f}{\partial x^2 \partial y^2}\right]_{x=XI(I),y=YI(J)} \tag{3}$$

(3) Note

Even if differential coefficients at mesh points are unknown, the calculation time is the same as with ILL=0 because ILL=0 is automatically assigned if ILL=0 is specified for the first time only when interpolation values at two or more points are to be repeatedly obtained. A smooth, beautiful figure can be obtained by using this routine as the preprocessing of contour lines and three-dimensional display of bivariable functions.

(1987.05.14)

HERM3S/D, HERP3S/D, HERD3S/D, and HERM3V/W (Data Fitting of Three-variable Function f(x, y, z) by the Piecewise Hermite Interpolation Formula)

Data Fitting of Three-variable Function f(x, y, z) by the Piecewise Hermite Interpolation Formula

Programmed by	Yasuyo Hatano, March 1990
Format	Subroutine Language: FORTRAN; Size: 1006, 1319, 1504, and 1293 lines respectively

(1) Outline

This function obtains the approximate value of the function values and first order partial derivatives at an arbitrary point in the region using the function value $f_{ijk}=f(x_i,y_j,z_k)$, $(i=1,\cdots,n_x,\ j=1,\cdots,n_y,k=1\ldots,n_z)$ given at the rectangular mesh point (x_i,y_j,z_k). The approximation method conforms to the piecewise Hermite interpolation of degree three.

HERM3S should be used when memory can be sufficiently used. HERP3S should be used when memory is insufficient. Generally, HERM3S is faster. If the total number of interpolation points is great, it is adequate to use HERD3S or HERM3V. If memory is sufficient, it is efficient to use HERM3V.

(2) Directions

```
CALL HERM3S/D(NP, CP, NC, CI, MC1, V, M, FD, MX, MY, MZ, IND, ICON)
CALL HERP3S/D(NP, CP, NC, CI, MC1, V, M, F, MFX, MFY, G, IND, ICON)
CALL HERD3S/D(NA, PA, MP1, NC, CI, MC1, FA, MAX, MAY, MAZ, M, F, MFX, MFY, NW, IND, ICON)
CALL HERM3V/W(NA, PA, MP1, NC, CI, MC1, FA, MAX, MAY, MAZ, M, FD, MX, MY, MX, NW, IND, ICON)
```

- HERM3S/D

Argument	Type and kind (*1)	Attribute	Content
NP	Integer type One-dimensional array	Input/output	Name of one-dimensional array containing three elements. As the input, NP(1) contains the value of mesh point number i ($x_i \leqq x \leqq x_{i+1}$) with respect to the x, y, and z coordinates at points where interpolation values are to be obtained, and NP(2) and NP(3) contain the value of j and k that meets the similar conditions with respect to y and z. If NP(1) is not equal to i that meets $x_i \leqq x \leqq x_{i+1}$, the value of i is output to NP(1). The value of j and k that meets the similar conditions with respect to y and z is output to NP(2) and NP(3).
CP	Real type one-dimensional array	Input	The value of coordinates (Xp, Yp, Zp) at points where interpolation values are to be obtained should be put in CP(1), CP(2), and CP(3). CI(1, K)≦CI(NP(K), K)≦CP(K)≦CI(NP(K)+1, K)≦CI(NC(K), K), K=1, 2, 3.
NC	Integer type one-dimensional array	Input	Name of one-dimensional array containing three elements. The total number of mesh points in the x, y, and z directions should be put in NC(1), NC(2), and NC(3) respectively. 2≦NC at IND=1. 3≦NC at 2≦IND.
CI	Real type two-dimensional array	Input	Name of two-dimensional array containing MC1*3 elements. The x coordinates at a mesh point where function values are given should be put in CI(*, 1). The y coordinates should be put in CI(*, 2), and the z coordinates should be put in CI(*, 3). The order should be an ascending order of each coordinate. CI(1, K)≦CI(2, K)≦...≦CI(NC(K), K), K=1, 2, 3.
MC1	Integer type	Input	The first subscript in the array declaration of CI.

Argument	Type and kind (*1)	Attribute	Content
V	Real type one-dimensional array	Output	The number of elements is up to eight. If M=0, the number of elements is one or a single variable. If M=1, the number of elements is 4. The number of elements is eight at M≧2. Functions and partial derivatives at the point (Xp, Yp, Zp) are output. V(1) is a function value. The value of V(2), V(3), and V(4) is output in the order of expression (1), the value of V(5), V(6), and V(7) is output in the order of expression (2), and the value of V(8) is output in the order of expression (3).
M	Integer type	Input	Index with respect to the order of differential coefficients to be obtained. 0≦M≦2.
FD	Real type four-dimensional array	Input/output	Four dimensional array containing (MX*MY*MZ*8) elements. As the input, the function values $f_{i,j,k}$ at mesh points should be put in FD(I, J, K, 1). If IND=0, partial derivatives at each mesh point ($C_iC_jC_k$) should be further put in FD(I, J, K, L) (L=2, ···8) in the order of expression (4). Note that C_i=CI(I, 1), C_j=CI(J, 2), and C_k=CI(K, 3). The output is the approximate value of differential coefficients by the piecewise Lagrange interpolation of the degree that corresponds to the value specified with IND. It becomes valid when IND≠0 is specified.
MX	Integer type	Input	The first subscript in the array declaration of FD.
MY	Integer type	Input	The second subscript in the array declaration of FD.
MZ	Integer type	Input	The third subscript in the array declaration of FD.

Argument	Type and kind (*1)	Attribute	Content
IND	Integer type	Input	Index with respect to input/output to the array FD. If IND=0, function values and partial derivatives at mesh points should be put in FD(I, J, K, L) (L=1, 2, ... 8). If the value of partial derivatives is unknown, IND≠0 is specified. At this time, the approximate value of partial derivatives is calculated using piecewise Lagrange interpolation in this routine before it is output to FD(I, J, K, L) (L=2, 3, ···, 8). If IND=1, the linear polynomial should be used. If IND=2, the quadratic polynomial should be used. If IND≧3, the quartic polynomial should be used.
ICON	Integer type	Output	Termination condition indication code. ICON=0: Normal termination. ICON=30000: Indicates that calculation is not executed at all because limits on the input argument are exceeded. 1≦ICON≦111: Indicates that the value of IP, JP, and KP is changed to meet the conditions of (XP, YP, ZP) before it is output.

*1 For double precision routines (those ending with D or W), all real types should be double precision real types.

- HERP3S/D

Argument	Type and kind (*1)	Attribute	Content
Same as HERM3S with respect to NP, CP, NC, CI, MC1, V, and M.			

Argument	Type and kind (*1)	Attribute	Content
F	Real type three-dimensional array	Input	Three dimensional array containing (MFX*MFY*NZ) or more elements. The function value f_{ijk} at a mesh point should be put in F(I, J, K).
MFX	Integer type	Input	The first subscript in the array declaration of F.
MFY	Integer type	Input	The second subscript in the array declaration of F.
G	Real type two-dimensional array	Input/output	Two-dimensional array containing (8*8) elements. This argument becomes useful as an input when this routine is called for the same NP(1, 2, 3) for many times. At the second or later call, IND=0 is specified, and this routine is called without changing the contents of G after the last call. The output is the approximate value of differential coefficients by piecewise Lagrange interpolation. It becomes valid when IND≠0 is specified.
IND	Integer type	Input	Index with respect to input/output to the array G. If IND=0, interpolation calculation is executed using the value of G at the last call. Refer to the explanation of G. If the value of partial derivatives is unknown, IND=0 is specified. At this time, the approximate value of partial derivatives is calculated using piecewise Lagrange interpolation in this routine before it is output to G. If IND=1, the linear polynomial should be used. If IND=2, the quadratic polynomial should be used. If IND≧3, the quartic polynomial should be used.

Argument	Type and kind (*1)	Attribute	Content
ICON	Integer type	Output	Termination condition indication code. ICON=0: Normal termination. ICON=30000: Indicates that calculation is not executed at all because limits on the input argument are exceeded.

*1 For double precision routines (those ending with D or W), all real types should be double precision real types.

- HERD3S/D

Argument	Type and kind (*1)	Attribute	Content
NA	Integer type One-dimensional array	Input	Name of one-dimensional array containing three elements. The total number of the x, y, and z coordinates at points where interpolation values are to be obtained should be put in NA(1), NA(2), and NA(3) respectively.
PA	Integer type one-dimensional array	Input	Name of two-dimensional array containing MP1*3 elements. The x coordinates at mesh points where interpolation values are to be obtained should be put in PA(*, 1), the y coordinates in PA(*, 2), and the z coordinates in PA(*, 3). The order should be an ascending order with respect to each coordinate. $CI(1, K) \leq PA(1, K) \leq PA(NP(K), K) \leq CI(NC(K), K)$.
MP1	Integer type	Input	The first subscript in the array declaration of PA.
Same as HERM3S and HERP3S with respect to NC, CI, and MC1.			

Argument	Type and kind (*1)	Attribute	Content
FA	Real type four-dimensional array	Output	Four dimensional array containing (MAX*MAY*MAZ*8) or more elements. Function values and partial derivatives at the mesh point (X_i, Y_j, Z_k) are output. Note that X_i=PA(I, 1), Y_j=PA(J, 2), and Z_k=PA(K, 3). Function values in FA(I, J, K, 1). If M≧1, the values are output in the order of expression (5). If M≧2, the values are further output in the order of expression (6).
MAX	Integer type	Input	The first subscript in the array declaration of FA.
MAY	Integer type	Input	The second subscript in the array declaration of FA.
MAZ	Integer type	Input	The third subscript in the array declaration of FA.
Same as HERM3S and HERD3S with respect to M, F, MFX, and MFY.			
NW	Integer type array	Work area	Two-dimensional array containing (MC1*6) elements.
IND	Integer type	Input	Index with respect to the order used when the approximate value of partial derivatives is calculated using piecewise Lagrange interpolation in this routine. If IND=1, the linear polynomial should be used. If IND=2, the quadratic polynomial should be used. If IND≧3, the quartic polynomial should be used.
ICON	Integer type	Output	Termination condition indication code. ICON=0: Normal termination. ICON=30000: Indicates that calculation is not executed at all because limits on the input argument are exceeded. 1≦ICON≦111: Indicates that the value of NP(1), NP(2), and NP(3) is changed to meet the conditions of (XP, YP, ZP) before it is output.

*1 For double precision routines (those ending with D or W), all real types should be double precision real types.

- HERM3V/W

Argument	Type and kind (*1)	Attribute	Content
Same as HERD3 with respect to NA, PA, MP1, NC, CI, MC1, FA, MAX, MAY, MAZ, and M.			
FD	Real type four-dimensional array	Input/output	Four dimensional array containing (MX*MY*MZ*8) elements. As the input, the function value f_{ijk} at mesh points should be put in FD(I, J, K, 1). If IND=0, partial derivatives at each mesh point (C_i^1, C_j^2, C_k^3) should be further put in FD(I, J, K, L) (L=2,...8) in the order of expression (4). Note that C_i^1=CI (I, 1), C_j^2=CI (J, 2), and C_k^3=CI(K, 3). The output is the approximate value of differential coefficients by the piecewise Lagrange interpolation of the degree that corresponds to the value specified with IND. It becomes valid when IND≠0 is specified.
MX	Integer type	Input	The first subscript in the array declaration of FD.
MY	Integer type	Input	The second subscript in the array declaration of FD.
MZ	Integer type	Input	The third subscript in the array declaration of FD.
NW	Integer type array	Work area	Two-dimensional array containing (MC1*6) elements.

Argument	Type and kind (*1)	Attribute	Content
IND	Integer type	Input	Index with respect to input/output to the array FD. If IND=0, function values and partial derivatives at mesh points should be put in FD(I, J, K, L) (L=1, 2, ... 8). If the value of partial derivatives is unknown, IND≠0 should be specified. At this time, the approximate value of partial derivatives is calculated using piecewise Lagrange interpolation in this routine before it is output to FD(I, J, K, L) (L=2, 3, ..., 8). If IND=1, the linear polynomial should be used. If IND=2, the quadratic polynomial should be used. If ND≧3, the quartic polynomial should be used.
ICON	Integer type	Output	Termination condition indication code. ICON=0: Normal termination. ICON=30000: Indicates that calculation is not executed at all because limits on the input argument are exceeded.

*1 For double precision routines (those ending with D or W), all real types should be double precision real types.

$$V(1)=\overline{f},\ V(2)=\left[\frac{\partial \overline{f}}{\partial x}\right]_{x=Xp,y=Yp,z=Zp},$$

$$V(3)=\left[\frac{\partial \overline{f}}{\partial y}\right]_{x=Xp,y=Yp,z=Zp},\quad V(4)=\left[\frac{\partial \overline{f}}{\partial z}\right]_{x=Xp,y=Yp,z=Zp} \tag{1}$$

$$V(5)=\left[\frac{\partial^2 \overline{f}}{\partial x \partial y}\right]_{x=Xp,y=Yp,z=Zp},\quad V(6)=\left[\frac{\partial^2 \overline{f}}{\partial y \partial z}\right]_{x=Xp,y=Yp,z=Zp}, \tag{2}$$

$$V(7)=\left[\frac{\partial^2 \overline{f}}{\partial x \partial z}\right]_{x=Xp,y=Yp,z=Zp} \tag{2}$$

$$V(8)=\left[\frac{\partial^3 \overline{f}}{\partial x \partial y \partial z}\right]_{x=Xp,y=Yp,z=Zp}, \tag{3}$$

$$FD(I,J,K,1)=\overline{f}(C_i^1,C_j^2,C_k^3),$$

$$FD(I,J,K,2)=\left[\frac{\partial \overline{f}}{\partial x}\right]_{x=C_i^1,y=C_j^2,z=C_k^3},$$

$$FD(I,J,K,3)=\left[\frac{\partial \overline{f}}{\partial y}\right]_{x=C_i^1,y=C_j^2,z=C_k^3},$$

$$FD(I,J,K,4)=\left[\frac{\partial \overline{f}}{\partial z}\right]_{x=C_i^1,y=C_j^2,z=C_k^3},$$

$$FD(I,J,K,5)=\left[\frac{\partial^2 \overline{f}}{\partial x \partial y}\right]_{x=C_i^1,y=C_j^2,z=C_k^3},$$

$$FD(I,J,K,6)=\left[\frac{\partial^2 \overline{f}}{\partial y \partial z}\right]_{x=C_i^1,y=C_j^2,z=C_k^3},$$

$$FD(I,J,K,7)=\left[\frac{\partial^2 \overline{f}}{\partial x \partial z}\right]_{x=C_i^1,y=C_j^2,z=C_k^3},$$

$$FD(I,J,K,8)=\left[\frac{\partial^3 \overline{f}}{\partial x \partial y \partial z}\right]_{x=C_i^1,y=C_j^2,z=C_k^3}, \tag{4}$$

$$FA(I,J,K,1)=\overline{f}(X_i,Y_j,Z_k)$$

$$FA(I,J,K,2)=\left[\frac{\partial \overline{f}}{\partial x}\right]_{x=Xi,y=Yj,z=Zk},$$

$$FA(I,J,K,3)=\left[\frac{\partial \overline{f}}{\partial y}\right]_{x=Xi,y=Yj,z=Zk},$$

$$FA(I,J,K,4)=\left[\frac{\partial \overline{f}}{\partial z}\right]_{x=Xi,y=Yj,z=Zk},$$

$$FA(I,J,K,5)=\left[\frac{\partial^2 \overline{f}}{\partial x \partial y}\right]_{x=Xi,y=Yj,z=Zk},$$

$$FA(I,J,K,6)=\left[\frac{\partial^2 \overline{f}}{\partial y \partial z}\right]_{x=Xi,y=Yj,z=Zk},$$

$$FA(I,J,K,7)=\left[\frac{\partial^2 \overline{f}}{\partial x \partial z}\right]_{x=Xi,y=Yj,z=Zk}, \quad (5)$$

$$FA(I,J,K,8)=\left[\frac{\partial^3 \overline{f}}{\partial x \partial y \partial z}\right]_{x=Xi,y=Yj,z=Zk}, \quad (6)$$

(3) Note

When partial derivatives at mesh points are unknown with respect to HERM3S/D, HERP3S/D, and HERM3V/W, if interpolation values at two or more points are to be repeatedly obtained, the calculation time can be shortened IND=0 if IND≠0 is specified for the first time only, and IND=0 is specified thereafter. By using this routine as the pre-processing for displaying the three-dimensional figures of 3-variable functions, a smooth, beautiful figure can be drawn.

HERP2S/D, HERD2S/D, HERM2S/D, and HERM2V/W (Data Fitting of Two-variable Function f (x, y) by the Piecewise Hermite Interpolation Formula)

Data Fitting of Two-variable Function f(x, y) by the Piecewise Hermite Interpolation Formula

Programmed by	Yasuyo Hatano, March 1990
Format	Subroutine Language: FORTRAN; Size: 809, 866, 635, and 287 lines respectively

(1) Outline

This function obtains a function value at a certain point (x,y) in the region and the approximate value of the first order partial derivative using the function values $f_{ij}=f(x_i,y_j)$, $(i=1,\cdots,n_x,\ j=1,\cdots,n_y)$ given at the rectangular mesh point (x_i,y_j). The approximation method conforms to the piecewise Hermite interpolation of degree three.

HERM2S/D should be used when memory can be sufficiently used. HERM2S/D should be used when memory is insufficient.

If the total number of interpolation points is great, it is adequate to use HERD2S/D or HERM2V/W. If memory is sufficient, it is most effective to use HAHERM2V/W.

(2) Directions

```
CALL HERP2S/D(NP, CP, NC, CI, MC1, V, M, F, MFX, G, IND, ICON)
CALL HERD2S/D(NA, PA, MP1, NC, CI, MC1, FA, MAX, MAY, M, F, MFX, NW, IND, ICON)
CALL HERM2S/D(NP, CP, NC, CI, MC1, V, M, FD, MX, MY, IND, ICON)
CALL HERM2V/W(NA, PA, MP1, NC, CI, MC1, FA, MAX, MAY, M, FD, MX, MY, NW, IND, ICON)
```

- HERP2S/D

Argument	Type and kind (*1)	Attribute	Content
NP	Integer type One-dimensional array	Input/output	Name of one-dimensional array containing two elements. As the input, NP(1) contains the value of mesh point number i ($x_i \leqq x \leqq x_{i+1}$) with respect to the x and y coordinates at points where interpolation values are to be obtained, and NP(2) contains the value of J that satisfies the similar conditions with respect to y. As the output, the value of i is output if NP(1) does not equal i that meets $x_i \leqq x \leqq x_{i+1}$. The value of j that meets the similar conditions with respect to y is output to NP(2).
CP	Real type one-dimensional array	Input	The value of coordinates (Xp, Yp) at points where interpolation values are to be obtained should be put to CP(1) and CP(2). CI(1, K) ≦ CI(NP(K), K) ≦ CP(K) ≦ CI(NP(K)+1, K) ≦ CI(NC(K), K), K=1, 2.
NC	Integer type one-dimensional array	Input	Name of one-dimensional array containing two elements. The total number of mesh points in the x and y directions should be put in NC(1) and NC(2). 2≦NC at IND=1. 3≦NC at 2≦IND.
CI	Real type two-dimensional array	Input	Name of two-dimensional array containing MC1*2 elements. The x coordinates at a mesh point where function values are given should be put in CI(*, 1), and the y coordinate should be put in CI(*, 2). The order should be an ascending order with respect to each coordinate. CI(1, K) ≦ CI(2, K) ≦ ... ≦ CI(NC(K), K), K=1, 2.
MC1	Integer type	Input	The first subscript in the array declaration of CI.

Argument	Type and kind (*1)	Attribute	Content
V	Real type one-dimensional array	Output	The number of elements is up to four. If M=0, the number of elements is one or a single variable. If M=1, the number of elements is three. If M≧2, the number of elements is four. The functions and partial derivatives at the point (Xp, Yp) are output. V(1) is a function value. V(2), V(3), and V(4) are output in the order shown in expression (1).
M	Integer type	Input	Index with respect to the order of differential coefficients to be obtained. 0≦M≦2.
F	Real type two-dimensional array	Input	Two-dimensional array containing (MFX*NY) or more elements. The value of functions $f_{i,j}$ at a mesh point should be put in F(I, J).
MFX	Integer type	Input	The first subscript in the array declaration of F.
G	Real type two-dimensional array	Input/output	Two-dimensional array containing (8*8) elements. This argument is useful as an input when this routine is called many times for the same NP(1) and NP(2). If IND=0 is specified, and this routine is called without changing the contents of G after the last call, the calculation is prompt. This argument is useful as an output when IND≠0 is specified, and the approximate value of differential coefficients by the piecewise Lagrange interpolation of the degree that corresponds to the value specified with IND is output.

Argument	Type and kind (*1)	Attribute	Content
IND	Integer type	Input	Index with respect to input/output to the approximation of differential coefficients and array G. If IND=0, interpolation calculation is executed by using the value of G at the last call. Refer to the explanation of G. If the value of differential coefficients is unknown, IND≠0 is specified. At this time, the approximate value of partial derivatives is calculated by using piecewise Lagrange interpolation in this routine before it is output to G. If IND=1, the linear polynomial should be used. If IND=2, the quadratic polynomial should be used. If IND≧3, the quartic polynomial should be used.
ICON	Integer type	Output	Termination condition indication code. ICON=0: Normal termination. ICON=30000: Indicates that calculation is not executed at all because limits on the input argument are exceeded. 1≦ICON≦111: Indicates that the value of NP(1) and NP(2) is changed to meet the conditions of (XP, YP) before it is output.

*1 For double precision subroutines (those ending with D or W), all real types should be double precision real types.

- HBRD2S/D

Argument	Type and kind (*1)	Attribute	Content
NA	Integer type One-dimensional array	Input	Name of one-dimensional array containing two elements. The total number of the x and y coordinates at mesh points where interpolation values are to be obtained should be put to NA(1) and NA(2).
PA	Integer type one-dimensional array	Input	Name of two-dimensional array containing MP1*2 elements. The x and y coordinates at a mesh point where interpolation values are to be obtained should be put in PA(*, 1) and PA(*, 2). The order should be an ascending order for each axis. $CI(1,K) \leqq PA(1,K) \leqq PA(NP(K),K) \leqq CI(NC(K),K), K=1,2.$
MP1	Integer type	Input	The first subscript in the array declaration of PA.
Same as HBRP2S with respect to NC, CI, and MC1.			
FA	Real type three-dimensional array	Output	Three dimensional array containing (MAX*MAY*4) or more elements. Functions and partial derivatives at the mesh point (X_I, Y_J) are output. Function values are output to FA(I, J, 1). If $M \geqq 1$, the values are output in the order of expression (2). If $M \geqq 2$, the values are additionally output in the order of expression (3). Note that X_I=PA(I, 1) and Y_J=PA(J, 2).
MAX	Integer type	Input	The first subscript in the array declaration of FA.
MAY	Integer type	Input	The second subscript in the array declaration of FA.
Same as HBRP2S with respect to M, F, and MFX.			
D	Real type array	Work area	Three-dimensional array containing (6*6*4) elements.

Argument	Type and kind (*1)	Attribute	Content
NW	Positive type array	Work area	Two-dimensional array containing (MC1*4) elements.
IND	Integer type	Input	Index with respect to the order used when the approximate value of partial derivatives is calculated using piecewise Lagrange interpolation in this routine. If IND=1, the linear polynomial should be used. If IND=2, the quadratic polynomial should be used. If IND≧3, the quartic polynomial should be used.
ICON	Integer type	Output	Termination condition indication code. ICON=0: Normal termination. ICON=30000: Indicates that calculation is not executed at all because limits on the input argument are exceeded.

*1 For double precision subroutines (those ending with D or W), all real types should be double precision real types.

- HERM2S/D

Argument	Type and kind (*1)	Attribute	Content
Same as HERP2S with respect to NP, CP, NC, CI, MC1, V, and M.			

84

Argument	Type and kind (*1)	Attribute	Content
FD	Real type three-dimensional array	Input/output	Three dimensional array containing (MX*MY*4) elements. As the input, the function values $f_{i,j}$ at a mesh point should be put in FD(I, J, 1). If IND=0, partial derivatives at each mesh point (C_i, C_j) should be further put in FD(I, J, L) (L=2, 3, 4) in the order of expression (4). Note that C_i=CI(I, 1) and C_j=CI(J, 2). The output is the approximate value of differential coefficients by the piecewise Lagrange interpolation of the degree that corresponds to the value specified with IND. It becomes valid when IND≠0 is specified.
MX	Integer type	Input	The first subscript in the array declaration of FD.
MY	Integer type	Input	The second subscript in the array declaration of FD.
IND	Integer type	Input	Index with respect to input/output to the array FD. If IND=0, function values and partial derivatives at a mesh point should be put in FD(I, J, L) (L=1, 2, ... 4). If the value of partial derivatives are unknown, IND≠0 should be specified. At this time, the approximate value of partial derivatives is calculated using piecewise Lagrange interpolation in this routine before it is output to FD(I, J, L) (L=2, 3, 4). If IND=1, the linear polynomial should be used. If IND=2, the quadratic polynomial should be used. If IND≧3, the quartic polynomial should be used.

Argument	Type and kind (*1)	Attribute	Content
ICON	Integer type	Output	Termination condition indication code. ICON=0: Normal termination. ICON=30000: Indicates that calculation is not executed at all because limits on the input argument are exceeded. 1≦ICON≦11: Indicates that the value of IP and JP is changed to meet the condition of (XP, YP) before it is output.

*1 For double precision routines (those ending with D or W), all real types should be double precision real types.

- HERM2V/W

Argument	Type and kind (*1)	Attribute	Content
Same as HERD2S with respect to NA, PA, MP1, NC, CI, MC1, FA, MAX, MAY, and M.			
FD	Real type three-dimensional array	Input/output	Three-dimensional array containing (MX*MY*4) elements. As the input, the function value $f_{i,j}$ at a mesh point should be put in FD(I, J, 1). If IND=0, partial derivatives at the mesh point (C_i^1, C_j^2) should be put in FD(I, J, L) (L=2, 3, 4) in the order of expression (4). Note that C_i^1=CI(I, 1) and C_j^2=CI(J, 2). The value to be output in the order of expression (4) is the approximate value of differential coefficients by the piecewise Lagrange interpolation of the degree that corresponds to the value specified with IND. The output becomes valid when IND≠0 is specified.
MX	Integer type	Input	The first subscript in the array declaration of FD.
MY	Integer type	Input	The second subscript in the array declaration of FD.

Argument	Type and kind (*1)	Attribute	Content
NW	Integer type array	Work area	Two-dimensional array containing (MC1*4) elements.
IND	Integer type	Input	Index with respect to input/output to the array FD. If IND=0, function values and partial derivatives at mesh points should be put in FD(I, J, L) (L=1, 2, ... 4). If the value of partial derivatives is unknown, IND≠0 should be specified. At this time, the approximate value of partial derivatives is calculated using piecewise Lagrange interpolation in this routine before it is output to FD(I, J, L) (L=2, 3, 4). If IND=1, the linear polynomial should be used. If IND=2, the quadratic polynomial should be used. If IND≧3, the quartic polynomial should be used.
ICON	Integer type	Output	Termination condition indication code. ICON=0: Normal termination. ICON=30000: Indicates that calculation is not executed as all because limits on the input argument are exceeded.

*1 For double precision subroutines (those ending with D or W), all real types should be double precision real types.

$$V(1)=\overline{f},\ V(2)=\left[\frac{\partial \overline{f}}{\partial x}\right]_{x=Xp,y=Yp},\ V(3)=\left[\frac{\partial \overline{f}}{\partial y}\right]_{x=Xp,y=Yp}, \tag{1}$$

$$V(4)=\left[\frac{\partial^2 \overline{f}}{\partial x \partial y}\right]_{x=Xp,y=Yp} \tag{1}$$

$$FA(I,J,1)=\overline{f}(X_i,Y_j)$$

$$FA(I,J,2)=\left[\frac{\partial \overline{f}}{\partial x}\right]_{x=Xi,y=Yj},\ FA(I,J,3)=\left[\frac{\partial \overline{f}}{\partial y}\right]_{x=Xi,y=Yj} \tag{2}$$

$$FA(I,J,4)=\left[\frac{\partial^2 \overline{f}}{\partial x \partial y}\right]_{x=Xi,\,y=Yj} \qquad (3)$$

$$FD(I,J,1)=\overline{f}(C_i^1,C_j^2),$$

$$FD(I,J,2)=\left[\frac{\partial \overline{f}}{\partial x}\right]_{x=C_i^1,C_j^2},$$

$$FD(I,J,3)=\left[\frac{\partial \overline{f}}{\partial y}\right]_{x=C_i^1,\,y=C_j^2},$$

$$FD(I,J,4)=\left[\frac{\partial^2 \overline{f}}{\partial x \partial y}\right]_{x=C_i^1,\,y=C_j^2} \qquad (4)$$

(3) Note

When partial derivatives at mesh points are unknown with respect to HERP2S/D, HERD2S/D, and HERM2V/W, if interpolation values at two or more points are to be repeatedly obtained, the calculation time can be shortened if IND≠0 is specified for the first time only, and IND=0 is specified thereafter. If this routine is used as the pre-processing for displaying figures such as the contour lines of double-variable functions, a smooth, beautiful figure can be drawn.

HERM2S has the function that is equivalent to the library HERM32. It is added to standardize the arguments.

88

LSAICS/D (Least Square Approximation by Orthogonal Polynomials)

Least Square Approximation by Orthogonal Polynomials

Programmed by	Yasuyo Hatano, February 1976
Format	Subroutine language: FORTRAN; size: 51 and 53 lines respectively

(1) Outline

LSAICS/D obtains a least squares solution [1] from the observed values $\overline{f}_1, \overline{f}_2, \cdots, \overline{f}_N$ at the discrete points $x_1, x_2, \cdots, x_N$ using the m-th order orthogonal polynomial of an unknown function $f(x)$. It determines the optimum degree m automatically using Minimum AIC Estimation method [2]. This routine consists of the following four entry names:

LSAICS/D Obtains the optimal order m of approximation polynomials.

LSFUNS/D Obtains the value of the approximation polynomial $y_m(x)$.

LSCOFS/D Obtains $c_j^{(m)}$ in $y_m(x)=\sum_{j=0}^{m} c_j^{(m)} x^j$ to be written.

LSDEGS/D Changes the degree of the approximation polynomial.

(2) Directions

```
CALL LSAICS/D(X, F, W, N, MIN, MAX, MOPT, P, P1, P2, AIC, ILL)
CALL LSFUNS/D(D, Y, K)
CALL LSCOFS/D(COF)
CALL LSDEGS/D(MD)
```

Argument	Type and kind (*1)	Attribute	Content
X	Real type One-dimensional array	Input	Name of array containing N elements. Discrete points $x_1, x_2, \cdots, x_N$. These points need not always be put in the order of magnitude.

Argument	Type and kind (*1)	Attribute	Content
F	Real type One-dimensional array	Input	Name of array containing N elements. The observation value at X(I) should be entered in F(I).
W	Real type One-dimensional array	Input/output	Name of array containing N elements. As an input, the weight should be entered. If ILL=0 is specified, the output becomes valid, and 1 is set in all Ws.
N	Integer type	Input	Total number of discrete points x_i.
MIN	Integer type	Input	Lower limit of degree of approximation polynomial. $0 \leq MIN \leq 20$
MAX	Integer type	Input	Upper limit of degree of approximation polynomial. $MIN \leq MAX \leq \min(N-1, 20)$
MOPT	Integer type	Output	Optimum degree of approximation polynomial.
P, P1, P2	Real type One-dimensional array	Input/output	Work area. Name of one-dimensional array containing N elements.
AIC	Real type One-dimensional array	Output	Name of array containing (MAX+1) elements. The value of AIC in polinomial of degree J is entered in AIC(J+1).
ILL	Integer type	Input/output	For the meaning as an input, see the item of W. The meaning as the output is as follows: If ILL=0, normal termination is assumed. If ILL=30000, no calculation was executed at all because limits on the argument were exceeded.

Argument	Type and kind (*1)	Attribute	Content
D	Real type	Input	Value of x coordinates to be obtained.
Y	Real type One-dimensional array	Output	Name of array containing K+1 elements. Value of approximation polynomial in D. (J-1)-th order differential coefficient is entered in Y(J). If K=0, even the real type variable name can be used.
K	Integer type	Input	Highest order of differential coefficients to be obtained.
COF	Real type One-dimensional array	Output	Name of array containing MOPT+1 elements. If an approximation polynomial is written as $y_m(x)=\sum_{j=0}^{m} c_j^{(m)} x^j$, the values of the coefficient $c_j^{(m)}$ are entered sequentially from the lower order. For instance, $y_2(x)=COF(1)+COF(2)\times x+COF(3)\times x^2$ is entered at MOPT=2. However, it is better to use LSFUNS/D to obtain the value of $y_m(x)$.
MD	Integer type	Input	The degree to be changed should be entered. If LSDEGS/D is called, Y and OF in LSFUNS/D and LSCOFS/D become the value based on the polynomial of degree MD. MD≦MAX

*1 For all double precision subroutines, all real types should be double precision real types.

(3) Note

A numerically steady orthogonal polynomial is taken as the base, and the scale of the observation points x_i and the observation values $\overline{f}_i$ is provided so that the digits are not missed. If the degree of polynomials to be applied is unknown, the function that determines the degree automatically by AIC is useful.

(4) Calculation method

Refer to "Research and Development Division Research Report No. 2," Nagoya University Computer

Center, 1976, p.5.

Bibliography

1) A.Ralston; "A First Course in Numerical Analysis", McGraw-Hill, p.232 (1965).

2) Kunio Tanabe; "Error in Numerical Calculation", Bit special issue, pp.113-125 (1975).

(1987.05.07)

92

LSANLS/D (Curve Fitting by Least Square Approximation with Nonlinear Parameters)

Curve Fitting by Least Square Approximation with Nonlinear Parameters

Programmed by	Yasuyo Hatano, October 1982
Format	Subroutine language: FORTRAN; size: 510 and 511 lines respectively

(1) Outline

LSANLS/D obtains the parameter $c_0, c_1, \cdots c_m$ from the initial estimate C_k^0 using the derivative correction method based on least squares method if n points $x_1, x_2 \cdots x_n$, and the observation value $y_1, y_2 \cdots, y_n$ in these points are given, and the form of the function

$$y=f(x, c_0, c_1, \cdots c_m) \quad (n>m)$$

is already known.

(2) Directions

CALL LSANLS/D(X, Y, N, C, M, EPS, W, KW, FUN, ITER, ILL)

Argument	Type and kind (*1)	Attribute	Content
X	Real type One-dimensional array	Input	Value of independent variable $x_i, i=1,2, \cdots, n$. Size N.
Y	Real type one-dimensional array	Input	Value of dependent variable $y_i, i=1,2, \cdots, n$. Size N. y for x=X(I) should be entered in Y(I).
N	Integer type	Input	Number of variables x_i.

Argument	Type and kind (*1)	Attribute	Content
C	Real type one-dimensional array	Input/output	The initial estimate of the non-linear parameter $C_0, C_1, \cdots C_m$ should be entered as an input. The value estimated by least squares method is entered as an output. Size M.
M	Integer type	Input	Number of non-linear parameters C_k. $1 \leqq M \leqq 10$, $M<N$
EPS	Real type	Input	Convergence criterion. If $C_k \sim 0$, this argument is used in the meaning of the absolute value. If $C_k \neq 0$, it is used in the meaning of the absolute value.
W	Real type two-dimensional array	Work area	Size (KW×N).
KW	Integer type	Input	Adjustable dimensions of W. KW≧N+1
FUN	Real type	Input	Subroutine for calculating the function value and $\partial f / \partial c_k$ for the function $f(x, c_1, c_2, \cdots c_m)$. The subroutine for that subroutine as a real argument must be prepared by the user, and defined by EXTERNAL declaration. FUN(XX, YY, G, C, M) XX (input): Independent variable x. YY (output): Value of function f. G (output): One-dimensional array of size M. G(K) contains the first order derivative of f at x=XX with respect to the parameter C(K). C (input): Parameter $C_k, k=1, 2, \cdots, M$. M (input): Number of parameters.

Argument	Type and kind (*1)	Attribute	Content
ITER	Integer type	Input/output	The upper limit on iteration count should be entered as an input. An actual iteration count is entered as an output. 1≦ITER
ILL	Integer type	Output	ILL=0: Normal termination. 1≦ILL≦M: No solution was obtained because the regular equation was under ill conditions. 20000<ILL≦20000+M: The solution did not settle. ILL=30000: No calculation was made because of limits on the input argument.

*1 For double precision subroutines, all real types should be double precision real types.

(3) Example of use

```
C.... EXAMPLE FOR LSANLS....
      F(X,A,B)=A*EXP(B*X)
      DIMENSION X(5),Y(5),W(6,5),C(2),EXACT(2)
      EXTERNAL FUN
      N=5
      M=2
      DO 1000 I=1,N
 1000 X(I)=FLOAT(I)/10.
      Y(1)=1.228
      Y(2)=1.005
      Y(3)=0.823
      Y(4)=0.674
      Y(5)=0.552
      C(1)=1.4
      C(2)=-1.0
      EXACT(1)=1.5
      EXACT(2)=-2.0
      EPS=1.E-4
      ITER=20
      CALL LSANLS(X,Y,N,C,M,EPS,W,6,FUN,ITER,ILL)
      WRITE(6,6100) N,M,ITER,ILL
 6100 FORMAT(1HO,' N,M,ITERATION, CONDITION =',4I6)
      WRITE(6,6200) (I,C(I),I=1,M)
      WRITE(6,6200) (I,EXACT(I),I=1,M)
 6200 FORMAT(2X,2(I3,F10.5))
      STOP
      END
      SUBROUTINE FUN(XX,YY,G,C,M)
      DIMENSION G(1),C(M)
      YY=C(1)*EXP(C(2)*XX)
```

```
G(1)=EXP(C(2)*XX)
G(2)=C(1)*XX*EXP(C(2)*XX)
RETURN
END
```

Bibliography

1) Written by T. R. Mackerra and translated by Isao Miura and Yoichi Tao; "Outline of Numerical Calculation for Computer," Science Library No 8, Science Company, p. 225 (1972).

(1987. 08. 11)

TETPCK (Three Dimensional C^k Interpolation Scheme for Irregularly Spaced Data ($0 \leqq k \leqq 1$))

Three Dimensional C^k Interpolation Scheme for Irregularly Spaced Data

Programmed by	Yoshio Sato, January 1979
Format	Subroutine language: FORTRAN; size: 970 lines

(1) Outline

TETPCK generates a tetrahedral mesh having the vertexes at each data point (x_i, y_i, z_i) for irregularly distributed three-variable function data x_i, y_i, z_i, $f_i = f(x_i, y_i, z_i)$, $(i = 1, 2, \cdots, N)$, and obtains the value of partial derivatives of up to k class at that data point. Then, it assigns to each of the tetrahedral elements the interpolation function to be the C^k class over the entire domain (convex polyhedral area) to obtain the interpolation value at rectangular hexahedral lattice points in the domain.

(2) Directions

CALL TETPCK(X, Y, Z, F, N, P, M1, M2, MX, MY, MZ, XL, YL, ZL, XU, YU, ZU, K, ICON)

Argument	Type and kind	Attribute	Content
X, Y, Z	Real type One-dimensional array	Input	Name of array containing N elements. x, y, and z coordinate at each data point. However, the number of points at the same coordinate must not be two or more.
F	Real type One-dimensional array	Input	Name of array containing N elements. Function value at each data point.
N	Integer type	Input	Number of data points. The size of N must be 4 to 5000 at K=0, and 10 to 5000 at K=1.
P	Real type Three dimensional array	Output	Name of array containing M1×M2×MZ elements. The interpolation value at rectangular hexahedral lattice points is entered.
M1, M2	Integer type	Input	Value of the first and second subscript in the array declaration of P. M1≧MX, M2≧MY

Argument	Type and kind	Attribute	Content
MX, MY, MZ	Integer type	Input	The number of rectangular hexahedral lattice interpolation partition points in the x, y, and z directions.
XL, YL, ZL	Real type	Input	Lower end position of rectangular hexahedral lattice interpolation points in the x, y, and z directions.
XU, YU, ZU	Real type	Input	Upper end position of rectangular hexahedral lattice interpolation points in the x, y, and z directions.
K	Integer type	Input	Indicates that the interpolation function is the C^K class. (K=0, 1). If K≠0 or 1, the interpolation part is skipped.
ICON	Integer type	Input/output	This argument has the following meaning as an input argument. ICON>0: Generates a tetrahedral mesh (at ICON=1 only), and calculates the value of first class partial derivatives at each data point (at K=1). ICON≦0: The above part is skipped.
ICON	Integer type	Input/output	This argument has the following meaning as an output argument. ICON=0: Normal. ICON<0: |ICON| means the number of interpolation points outside the domain. If the interpolation point is outside the domain, an extrapolation value is obtained by the least squares method using the $(k+1)$-th order polynomial, and entered in P. ICON=30000: The limit on the input argument is broken. ICON=10000: Break Down by work area shortage, etc. (Rare. Error messages are printed.)

(3) Example of use

The principal part of the main program for using TETPCK is as follows:

```
  DIMENSION X(350), Y(350), Z(350), F(350), P(20, 20, 20)
    :·····Calculations of X, Y, Z, and F
  ICON=1
  CALL TETPCK(X, Y, Z, F, 125, P, 20, 20, 10, 10, 2, 1.0, 1.0, 4.0, 10.0, 10.0, 7.0, 1, ICON)
    :
  END
```

The following program gives the same result, too. (See 3 in Note)

```
  DIMENSION X(350), Y(350), Z(350), F(350), P(20, 20, 20)
    :·····Calculations of X, Y, Z, and F
  ICON=1
  XP=1.0
  DO 1 I=1, 10
  YP=1.0
  DO 2 J=1, 10
  ZP=4.0
  DO 3 K=1, 2
  CALL TETPCK(X, Y, Z, F, 125, P(I, J, K), 1, 1, 1, 1, 1, XP, YP, ZP, XP, YP, ZP, 1, ICON)
3 ZP=ZP+3.0
2 YP=YP+1.0
1 XP=XP+1.0
```

```
      :
      END
```

(4) Note

1. This routine should be called with ICON=1 only once at the first time for the same value of X, Y, Z, P, N, and K. The generation of a tetrahedral mesh and the evaluation of partial derivatives are completed at the first call, and ICON≦0 is made at the result. Because the above part should be skipped for the same data at the subsequent steps, this routine should be called with ICON≦0 hereafter.

2. The number of four vertexes of each element of the generated tetrahedral mesh can be referred to with the named COMMON statement as shown below.

```
COMMON/CL0123/L0(40000),L1(40000),L2(40000),L3(40000),L
```

The number of four vertexes of L tetrahedral elements is stored in L0(I), L1(I), L2(I), and L3(I) (I=1,2,・・・,L) so that the three vertexes L1(I), L2(I), and L3(I) are ordered clockwise as viewed from the vertexes L0(I).

3. When the interpolation value at a point (x_p, y_p, z_p) is to be obtained, $M1=M2=MX=MY=MZ=1$, $XL=x_p$, and $YL=y_p$, $ZL=z_p$ should be assumed. In this case, the output argument P can be a real type variable.

(1987.05.14)

TRIPCK (Two Dimensional C^k Interpolation Scheme for Irregularly Spaced Data ($0 \leqq k \leqq 3$))

Two Dimensional C^k Interpolation Scheme for Irregularly Spaced Data

Programmed by	Yoshio Sato, January 1979
Format	Subroutine language: FORTRAN; size: 549 lines

(1) Outline

TRIPCK assigns to each of triangular elements the interpolation function to be the C^k class over the entire domain (convex polygonal area) and obtains the interpolation value at rectangular mesh points in the domain after generating a triangular mesh having the vertexes at each data point (x_i, y_i) and obtaining the value of partial derivatives of up to the k class at each data point for irregularly spaced bivariable function data

x_i, y_i, $f_i = f(x_i, y_i)$, $(i = 1, 2, \cdots, N)$.

(2) Directions

CALL TRIPCK(X, Y, F, N, P, M1, MX, MY, XL, YL, XU, YU, K, ICON)

Argument	Type and kind	Attribute	Content
X, Y	Real type One-dimensional array	Input	Name of one-dimensional array containing N elements. x, y coordinate at each data point. Two or more points of the same coordinate must not exist.
F	Real type One-dimensional array	Input	Name of one-dimensional array containing N elements. Function value at each data point.
N	Integer type	Input	Number of data points. The size of N must be 3 to 5000 for K=0, 6 to 5000 for K=1, 10 to 5000 for K=2, and 15 to 5000 for K=3.
P	Real type Two-dimensional array	Output	Name of two-dimensional array containing M1×MY elements. Interpolation values at rectangular mesh points are entered.
M1	Integer type	Input	Value of the first subscript in the array declaration of P. M1≧MX

Argument	Type and kind	Attribute	Content
MX, MY	Integer type	Input	Number of partition points in the x and y directions at rectangular mesh interpolation points.
XL, YL	Real type	Input	Lower end position in the x and y directions at rectangular mesh interpolation points.
XU, YU	Real type	Input	Upper end position in the x and y directions at rectangular mesh interpolation points.
K	Integer type	Input	Indicates that the interpolation function is the C^K class. K=0, 1, 2, and 3. Interpolation part is skipped if K≠0, 1, 2, and 3.
ICON	Integer type	Input/output	This argument has the following meaning as an input argument. ICON>0: Generates a triangular mesh (only at ICON=1), and calculates the partial derivatives of up to the k class (K=1, 2, and 3) at each data point. ICON≦0: The above part is skipped.
			This argument has the following meaning as an output argument. ICON=0: Normal. ICON<0: |ICON| represents the number of interpolation points outside the domain. If the interpolation point is outside the domain, an extrapolation value is obtained by the least squares method using the $(k+1)$-th order polynomial, and entered in P. ICON=30000: The limit on the input argument is not kept.

(3) Example of use

The principal part of the main program for using TRIPCK is as follows:

```
      DIMENSION X(500),Y(500),F(500),P(20,20)
       :                                          ·····Calculation of X, Y, and F
      ICON=1
      CALL TRIPCK(X,Y,F,400,P,20,10,10,1.0,1.0,10.0,10.0,1,ICON)
       :
      END
```

The following program also gives the same result. (See 3 in Note.)

```
      DIMENSION X(500),Y(500),F(500),P(20,20)
       :                                          ·····Calculation of X, Y, and F
      ICON=1
      XP=1.0
      DO 1 I=1,10
      YP=1.0
      DO 2 J=1,10
      CALL TRIPCK(X,Y,F,400,P(I,J),1,1,1,XP,YP,XP,YP,1,ICON)
    2 YP=YP+1.0
    1 XP=XP+1.0
       :
      END
```

(4) Note

1. This routine should be called with ICON=1 only once at the first time for the same value of X, Y, F, N, and K. The generation of a triangular mesh and the evaluation of some partial derivatives are completed at the first call, and ICON≦0 is made at the result. Because the above part should be skipped for the same data at the subsequent steps, this routine should be called with ICON≦0 hereafter.

2. The number of three vertexes of each element of the generated triangular mesh can be referred to with the named COMMON statement as shown below.

```
COMMON/CL9995/L1(9995),L2(9995),L3(9995),L
```

The number of three vertexes of L triangular elements is stored counterclockwise in $L1(i), L2(i), L3(i)$, $(i=1,2,\cdots,L)$.

3. If the interpolation value at a point (x_p, y_p) is to be obtained, $M1=MX=MY=1$, $XL=x_p$, $YL=y_p$ must be assumed. In this case, the output argument P can be a real type variable.

4. The program TRIMAP for generating a triangular mesh for irregularly distributed bivariable function data (of up to 5000 points) and displaying the contour line is prepared. Refer to p. 110 in "Chart Output Guide".

Bibliography

1) Yoshio Sato; "Display of Contour Lines for Irregularly Distributed Data and C^k Class Interpolation", Nagoya University Computer Center News, Vol. 10, No. 2, p. 161 (1979).

2) Yoshio Sato and Ichizo Ninomiya; "Two-Dimensional C^k Interpolation for Irregularly Spaced Data", Transactions of Information Processing Soc. of Japan, Vol. 22, p. 581. (1981).

(1987. 05. 08)

7. Fourier analysis

BITREV/BITRVD/BITRVC/BITRVB (Rearrangement of Data by Bit Reversal)

Rearrangement of Data by Bit Reversal

Programmed by	Ichizo Ninomiya, April 1981
Format	Subroutine Language: Assembler; Size: 108, 110, and 114 lines respectively

(1) Outline

BITREV/BITRVD/BITRVC/BITRVB is a subroutine for rearrangement by bit reversal required for the fast Fourier transform. The bit reversal is to reverse the order of binary bits. If the reversal of M-digit binary number K is represented with $\overline{K}$, this routine stores A(K) in $A(\overline{K-1}+1)$ for integer K from 1 to 2^M.

(2) Directions

CALL BITREV(A, M, ILL)

CALL BITRVD(A, M, ILL)

CALL BITRVC(A, M, ILL)

CALL BITRVB(A, M, ILL)

Argument	Type and kind (*1)	Attribute	Content
A	Real type One-dimensional array	Input/output	One-dimensional array with 2^M elements. The elements are rearranged by bit reversal in this routine.
M	Integer type	Input	Indicates that the size of array A is 2^M. $M \geq 0$
ILL	Integer type	Output	If M<0, ILL=30000, and calculation is not performed. In all other cases, calculation is performed, and ILL=0.

*1 For BITRVD(BIRVB), all real types should be changed to double precision real types.

(3) Performance

10K

Effective algorithm and careful coding make this routine very fast.

(4) Calculation method

Refer to the bibliography [1] .

Bibliography

1) Ichizo Ninomiya; "Method of Bit Reversal Scrambling," Preprints of the 23th Symposium of Information Processing Soc. of Japan, pp. 899-900 (1981).

(1987. 08. 10) (1987. 08. 21)

DRCH1S/D,DRCH3S/D,IICH1S/D,IICH3S/D

(Derivative of First Kind Chebyshev Series) (Derivative of Shifted Chebyshev Series)

(Indefinite Integral of First Kind Chebyshev series)

(Indefinite Integral of Shifted Chebyshev Series)

Derivative of First Kind Chebyshev Series (DRCH1S/D)

Derivative of Shifted Chebyshev Series (DRCH3S/D)

Indefinite Integral of First Kind Chebyshev (IICH1S/D)

Indefinite Integral of Shifted Chebyshev Series (IICH3S/D)

Programmed by	Tatsuo Torii, December 1978
Format	Subroutine Language: FORTRAN; Size: 24, 24, 24, 24, 26, 26, 26, and 27 lines respectively

(1) Outline

The subroutines represent the termwise differentiation and integration of the first kind Chebyshev series $\sum'_{0\leq k<N} a_k T_k(x)$ from given with Chebyshev series. Similarly, the termwise differentiation and integral of the shifted Chebyshev series $\sum'_{0\leq k<N} a_k T_k^*(x)$ are obtained from series of $\{T_k{}^*(x)\}$.

(2) Directions

CALL DRCH1S/D(A, NA, B, NB, ICON)

CALL IICH1S/D(A, NA, B, NB, ICON)

CALL DRCH3S/D(A, NA, B, NB, ICON)

CALL IICH3S/D(A, NA, B, NB, ICON)

Argument	Type and kind (*1)	Attribute	Content
A	Real type One-dimensional array	Input	DRCHB1S/DRCH1D and IICHB1S/IICH1D: The coefficients of the first kind Chebyshev series are stored in A. Number of terms NA≧1
NA	Integer type		DRCHB3S/DRCH3D and IICHB3S/IICH3D: Series of the shifted Chebyshev polynomial.

Argument	Type and kind (*1)	Attribute	Content
B	Real type one-dimensional array	Output	The coefficients of series to which termwise integration or differentiation is applied are stored in array B. NB≧1 is the number of coefficients of an output.
NB	Integer type		
ICON	Integer type	Output	ICON=0: Normal. ICON=30000: Parameter error.

*1 For double precision subroutines, all real types should be double precision real types.

(3) Calculation method

The Chebyshev series of N terms is integrated termwise to

$$\int_{-1}^{x} {\sum_{0\leq k<N}}' a_k T_k(x)dx = \sum_{0\leq k<N}' b_k T_k(x)$$

where

$$b_k=(a_{k-1}-a_{k+1})/2k, k\geq 1$$

and

$$b_0=2\sum_{k=1}^{N+1}(-1)^{k-1}b_k$$

However, $a_{N+1}=a_N=0$.

In termwise differentiation, inversely the coefficient $\{a_k\}$ is obtained by giving $\{b_k\}$. When the series is to be expanded by shifted Chebyshev polynomials, the relations $b_k = (a_{k-1}-a_{k+1})/4k, k\geq 1$ hold between the coefficients of both series $\int^{x}{\sum}' a_k T_k^{*}(x)dx = {\sum}' b_k T_k^{*}(x)$.

(4) Example

If a trigonometric function is expanded into Chebyshev series, a Bessel function is appeared.

$$\cos ax = J_0(a)+2\sum_{k=1}^{\infty}(-1)^k J_{2k}(a)T_{2k}(x)$$

$$\sin ax = 2\sum_{k=1}^{\infty}(-1)^k J_{2k+1}(a)T_{2k+1}(x)$$

The right hand side is integrated and differentiated termwise. The following program integrates and differentiates $\cos ax$ and $\sin ax$ termwise by expanding them by shifted Chebyshev polynomials. The integration constant is defined so that the sum of the series equals 0 at x=-1 (or x=0 when the shifted Chebyshev polynomial is used).

```
C       TEST FOR SUBROUTINE IICH1S AND DRCH1S.
        DIMENSION A(258),B(258),C(258)
        EXTERNAL F
        COMMON L,T
        DATA EPSA,EPSR,NMIN,NMAX/0.0,0.0,0,257/
        T=10.0
        DO 10 L=1,2
        CALL FCHB1S(F,EPSA,EPSR,NMIN,NMAX,A,N,ERR,ILL1)
        CALL IICH1S(A,N,B,NB,ILL2)
        CALL DRCH1S(A,N,C,NC,ICON)
        ICON=ICON+ILL1+ILL2
        WRITE(6,601) L,T,ICON
  600   FORMAT(////7X,9HPROBLEM (,I2,1H),5X,1HT,F8.3,6X,4HICON,
     *         I8)
        WRITE (6,600) (I,A(I),B(I),C(I),I=1,NB)
  600   FORMAT(1HO/(1H ,I8,3F25.06))
        CALL VCHB1S(B,NB,-1.0,VB,ICON)
        WRITE(6.603) VB
  603   FORMAT(///8X,26HCHECK OF INTEGRAL CONSTANT,E25.5)
   10   CONTINUE
        STOP
        END

        FUNCTION F(P)
        COMMON L,T
        GO TO (1,2),L
    1   F=SIN(T*P)
        RETURN
    2   F=COS(T*P)
        RETURN
        END
```

(1987.05.29) (1987.08.08) (1987.08.10)

FCHB1S/D, FCHB2S/D, FCHB3S/D, FCHBOS/D

(Fourier Expansion of Functions by Chebyshev Polynomials of First Kind) (FCHB1S/D)

(Fourier Expansion of Functions by Chebyshev polynomials of Second Kind) (FCHB2S/D)

(Fourier Expansion of Functions by Shifted Chebyshev Polynomials) (FCHB3S/D)

(Fourier Expansion of Functions on The Open Interval by First Chebyshev Polynomials) (FCHBOS/D)

Fourier Expansion of Functions by Chebyshev Polynomials of First Kind(FCHB1S/D)

Fourier Expansion of Functions by Chebyshev Polynomials of Second Kind(FCHB2S/D)

Fourier Expansion of Functions by Shifted Chebyshev Polynomials(FCHB3S/D)

Fourier Expansion of Functions on The Open Interval by First Chebyshev Polynomials(FCHBOS/D)

Programmed by	Tatsuo Torii, July 1978
Format	Subroutine Language: FORTRAN; Size: 98, 99, 78, 79, 91, 92, 101, and 103 lines respectively

(1) Outline

The function $f(x)$ given in the finite interval (open or closed interval) is expanded in the Chebyshev series according to the required precision ε. The basis of the calculation method is the same as the cosine series expansion of the periodic function (sine series).

FCHB1 expands a smooth function $f(t)$ on a closed interval [-1, 1] by the first kind Chebyshev polynomials.

$$f(t) \simeq \sum_{0\le k\le N}{}'' C_k T_k(t) = \sum_{0\le K\le N}{}'' C_k \cos k\theta$$

Where $t=\cos\theta$, and the order number N=N(ε) takes the value of power of 2.

FCHB2 expands a smooth function $f(t)$ in an open interval (-1, 1) with the Chebyshev polynomials of the second kind.

$$f(t) \simeq \sum_{0\le k\le N-2} C_k U_k(t) = \sum_{0<k<N} C_{k-1}\frac{\sin k\theta}{\sin\theta}$$

The smooth function is expanded over a closed interval [0, 1] with the shifted Chebyshev polynomials.

$$f(t) \simeq \sum_{0\le k\le N}{}'' C_k T_k^*(t) = \sum_{0\le k\le N}{}'' C_k \cos k\theta$$

Where $t=\cos^2\theta/2$.

If the function $f(t)$ that cannot take both ends of a given interval as sampling points is to be expanded with the Chebyshev polynomials of first kind, FCHBO should be used.

$$f(t)=\sum_{0\leq k\leq N-2} C_k T_k(t)$$

(2) Directions

CALL FCHB1S/D(F, EPSA, EPSR, NMIN, NMAX, A, N, ERR, ICON)

CALL FCHB2S/D(F, EPSA, EPSR, NMIN, NMAX, A, N, ERR, ICON)

CALL FCHB3S/D(F, EPSA, EPSR, NMIN, NMAX, A, N, ERR, ICON)

CALL FCHBOS/D(F, EPSA, EPSR, NMIN, NMAX, A, N, ERR, ICON)

Argument	Type and kind (*1)	Attribute	Content
F	Real type Function subprogram	Input	The user should define the function of one variable as a function subprogram. The domain of the function must be [-1, 1] for FCHB1S/D, (-1, 1) for FCHB2S/D, [0, 1] for for FCHB3S/D, (-1, 1) for FCHBOS/D.
EPSA EPSR	Real type	Input	Error bound of Chebyshev series to be obtained. EPSA≧0 is the precision required for absolute error, and EPSR≧0 is the precision required for relative error.
NMIN NMAX	Integer type	Input	Lower and upper limits of the number of samples. FCHB1S/D and FCHB3S/D: 0≦NMIN≦NMAX≦1025 FCHB2S/D and FCHBOS/D: 0≦NMIN≦NMAX≦1023
A	Real type One-dimensional array	Output	Size of array A≧NMAX. An N number of Chebyshev polynomial coefficients are stored in the normal order. N takes a positive integer value of the form of 2^n+1 for both FCHB1S/D and FCHB3S/D, and of the form of 2^n-1 for both FCHB2S/D and FCHBOS/D.
N	Integer type		
ERR	Real type	Output	Upper bound of errors of obtained Chebyshev series (see the note below).

Argument	Type and kind (*1)	Attribute	Content
ICON	Integer type	Output	If ICON=0, the argument is normal in the following sense: Error $\leq \max\{EPSA, EPSR * \|\| f \|\|\}$ ICON=10000: Because the required precision is too severe, the above condition is not satisfied. However, the truncation error is decreased to rounding error level (limit of calculation error). ICON=20000: Abnormal. Even if the number of samples reaches the upper limit NMAX, the truncation error does not decrease to the level of required or rounding error. ICON=30000: Parameter error.

Note: The norm definition in FCHB1S/D, FCHB3S/D, and FCHB0S/D is $\|f\|_\infty = \max_j |f(x_j)|$, where x_j is a sampling point. In FCHB2S/D, $\|f\|_1 = \sum |C_k|$, where C_k is the Chebyshev expansion coefficient of second kind of $f(t)$. Each truncation error based on these norm is estimated.

*1 For double precision subroutines, all real types should be double precision real types.

(3) Performance

If the time required for the sampling of $f(t)$ is excluded, the time is almost same to that of fast cosine (sine) transformation based on the trapezoidal rule.

(4) Calculation method

Fast cosine transformation for the even function $f(\cos\theta), f(\cos^2\theta/2)$ given in a closed interval of [0, π] is simply FCHB1S or FCHB3S. The cosine transformation of $f(\cos\theta)$ that does not use both ends as the sample points corresponds to FCHB03. Fast sine transformation for the odd function $f(\cos\theta)\sin\theta$ is FCHB2S. The error of obtained Chebyshev series is estimated by the sum of absolute values of the coefficients of the last two terms.

Each subroutine has a one-dimensional array for trigonometric function tables (511 words for FCHB0S, FCHB1S, and and FCHB2S, and 1023 words for FCHB3S). This array is shared with cosine (sine) transformation and sampling points. These constant tables are used for calculation only when each subroutine is called for the first time, and retained thereafter.

(5) Example

The functions are expanded by the Chebyshev polynomials of first kind under a required precision. The generating function is used as test function of the Chebyshev polynomials of first kind

$$\frac{1-t^2}{1-2tx+t^2}=2\sum_{k=0}^{\infty}{}' t^k T_k(x),\quad 0<t<1$$

The following is an example of FCHB1S when 1/4, 2/4, and 3/4 are assigned to the parameter t. The required precision for absolute error is 10^{-5}. The lower and upper limits of the number of samples are described in the following programs:

```
C       TEST FOR SUBROUTINE FCHB1S.
        DIMENSION A(257)
        EXTERNAL F
        COMMON T
        AEPS=1.0E-05
        REPS=0.0
        NMIN=0
        NMAX=257
        T=0.25
        H=0.25
      1 CONTINUE
        CALL FCHB1S (F,AEPS,REPS,NMIN,NMAX,A,N,ERR,ICON)
        WRITE(6,600) N,EER,ICON,T,(A(I),I=1,N)
        T=T+H
        IF(T.LT.1.0) GO TO 1
    600 FORMAT(1HO,4X,2HN=,I3,5X,4HERR=,E10.3,5X,5HICON=,
       *       I5/1HO,4X,*7HARRAY A,5X,2HT=,F5.2/(1H ,4F15.06))
        STOP
        END

        FUNCTION F(P)
C GENERATING FUNCTION OF CHEBYSHEV POLYNOMIALS OF FIRST KIND.
        COMMON T
        F=(1.0-T*T)/(1.0-2.0*T*P+T*T)
        RETURN
        END
```

Expansion of generating functions of the Chebyshev polynomials of first kind

k	t=1/4	t=1/2	t=3/4
0	2.000000	2.000000	2.000000
1	0.500000	1.000000	1.500000
⋮	⋮	⋮	⋮
8	0.000031	0.007813	0.200226
9	0.000008	0.003906	0.150169
⋮	⋮	⋮	⋮
16	0.000000	0.000031	0.020045

k	t=1/4	t=1/2	t=3/4
17		0.000015	0.015034
⋮		⋮	⋮
32		0.000000	0.000201
33		0.000000	0.000151
⋮			⋮
64			0.000000
65			0.000000
Number of terms	17	33	65
Estimated value of error	0.397E-06	0.715E-06	0.167E-05

Note: The number k shows the order of the output data.

The subroutine FCHB2S is tested by using the generating function of the Chebyshev polynomials of second kind

$$\frac{1}{1-2tx+t^2}=\sum_{k=0}^{\infty}t^k U_k(x)$$

```
C       TEST FOR SUBROUTINE FCHB2S
        DIMENSION A(255)
        EXTERNAL F
        COMMON T
        AEPS=1.0E-05
        REPS=0.0
        NMAX=255
        NMIN=0
        T=0.25
        H=0.25
      1 CONTINUE
        CALL FCHB2S(F,AEPS,REPS,NMIN,NMAX,A,N,ERR,ICON)
        WRITE(6,600) N,EER,ICON,T,(A(I),I=1,N)
        T=T+H
        IF(T.LT.1.0) GO TO 1
    600 FORMAT(1HO,4X,2HN=,I3,5X,4HERR=,E10.3,5X,5HICON=,
       *       I5/1HO,4X,7HARRAY A,5X,2HT=,F5.2/(1H ,4F15.06))
        STOP
        END

        FUNCTION F(P)
C GENERATING FUNCTION OF CHEBYSHEV POLYNOMIALS OF SECOND KIND.
```

```
COMMON T
F=1.0/(1.0-2.0*T*P+T*T)
RETURN
END
```

Expansion of generating functions of Chebyshev polynomial of the second kind

k	t=1/4	t=1/2	t=3/4
0	1.000000	1.000000	1.000000
1	0.250000	0.500000	0.750000
2	0.062500	0.250000	0.562500
⋮	⋮	⋮	⋮
15		0.000031	0.013363
16		0.000015	0.010023
17		0.000008	0.007517
⋮		⋮	⋮
31			0.000134
32			0.000100
33			0.000075
⋮			⋮
62			0.000000
Number of terms	15	31	63
Estimated value of error	0.159E-06	0.238E-06	0.477E-06

Note: The number k shows the order of the output data.

The following two functions are expanded by using the shifted Chebyshev polynomials.

$$f(x)=\frac{1-t^2}{1-2t(2x-1)+t^2},\ 0\leq x\leq 1$$

$$=-2\sum_{k=0}^{\infty}{}' t^{-k}T_k^*(x),\ t>1$$

$$g(x)=\frac{1}{1+x^2},\ -t\le x\le t$$

$$=\frac{2t^{-1}}{\sqrt{1+t^{-2}}}\sum_{k=0}^{\infty}{}'(-1)^k(t^{-1}+\sqrt{1+t^{-2}})^{-2k}T_k^*((\frac{x}{t})^2)$$

Because the domain of the function $g(x)$ is [-t, t], and the function is an even function, the variable transformation $y=(x/t)^2$ is adopted. 2, 4, and 8 are assigned to the parameter t. Required precision for absolute error is 10^{-5}.

```
C       TEST PROBLEMS OF SUBROUTINE FCHB3S
        DIMENSION A(257)
        EXTERNAL F
        COMMON T,L
        AEPS=1.0E-05
        REPS=0.0
        NMIN=0
        NMAX=257
        DO 10 L=1,2
        T=2.0
      1 CONTINUE
        CALL FCHB3S(F,AEPS,REPS,NMIN,NMAX,A,N,ERR,ICON)
        WRITE(6,601) L
    601 FORMAT(1HO,4X,9HPROBLEM (,I1,1H),)
        WRITE(6,600) N,ERR,ICON,T,(A(I),I=1,N)
    600 FORMAT(1HO,4X,2HN=,I3,5X,4HERR=,E10.3,5X,5HICON=,
       *       I5/1HO,4X,7HARRAY A,5X,2HT=,F5.2/(1H ,4F15.06))
        T=T+T
        IF(T.LE.8.0)GO TO 1
     10 CONTINUE
        STOP
        END

        FUNCTION F(P)
        COMMON T,L
        GO TO (10,20),L
C       PROBLEM (1)
     10 CONTINUE
C       GENERATING FUNCTION OF SHIFTED CHEBYSHEV POLYNOMIALS.
        Q=P+P-1.0
        F=(1.0-T*T)/(1.0-2.0*T*Q+T*T)
        RETURN
C       PROBLEM (2)
     20 CONTINUE
C       APPLY THE VARIABLE TRANSFORMATION.
        Q=T*SQRT(P)
        F=1.0/(1.0+Q*Q)
        RETURN
        END
```

The following lists show the results of the function $1/(1+x^2)$ expanded with $\{T_k^*((x/t)^2)\}$.

k	t=2	t=4	t=8
0	0.894427	0.485071	0.248069
1	-0.341641	-0.295705	-0.193322
2	0.130495	0.180265	0.150656
⋮	⋮	⋮	⋮
15	-0.000001	-0.000289	-0.005891
16	0.000000	0.000176	0.004591
17		0.000108	-0.003578
⋮		⋮	⋮
31		0.000000	-0.000109
32		0.000000	0.000085
33		0.000000	-0.000066
⋮			⋮
63			0.000000
64			0.000000
Number of terms	17	33	65
Estimated value of error	0.905E-06	0.272E-06	0.238E-06

Note: The number k means the order of the output data.

When the function defined in an open interval is to be expanded into the Chebyshev series of first kind, FCHBOS should be used. Because FCHBOS does not use the end points as sampling points, its precision is generally inferior to the one that uses them as sampling points. For comparison with FCHBIS, the generating function

$$f(x)=\frac{1-t^2}{1-2xt+t^2}\,2{\sum}'t^kT_k(x)$$

is expanded over an interval [-1, 1]. The function

$$g(x)=\frac{(1-t^2)(1+x)}{2\{(1-t^2)+(1+t)^2x\}}=\sum_{k=0}^{\infty}{}'t^kT_k\left(\frac{1-x}{1+x}\right)$$

defined by (0, ∞) is transformed to [-1, 1] by the variable transformation $y=(1-x)/(1+x)$.

and expanded in the Chebyshev series.

The following is an example of calculation when 1/4, 2/4, and 3/4 are allocated to the parameter t.

```
C       TEST PROBLEMS OF SUBROUTINE FCHBOS
        DIMENSION A(255)
        COMMON T,L
        EXTERNAL F
        AEPS=1.0E-05
        REPS=0.0
        NMIN=0
        NMAX=255
        DO 10 L=1,2
        T=0.25
        H=0.25
     1  CONTINUE
        CALL FCHBOS(F,EPSA,EPSR,NMIN,NMAX,A,N,ERR,ICON)
        WRITE(6,601) L
        WRITE(6,600) N,EER,ICON,T,(A(I),I=1,N)
   601  FORMAT(1HO,4X,9HPROBLEM (,I1,1H),)
   600  FORMAT(1HO,4X,2HN=,I3,5X,4HERR=,E10.3,5X,5HICON=,I5/1HO,
     *         4X,7HARRAY A,5X,2HT=,F5.2/(1H ,4F15.06))
        T=T+H
        IF(T.LT.1.0) GO TO 1
    10  CONTINUE
        STOP
        END

        FUNCTION F(P)
        COMMON T,L
        GO TO (10,20),L
    10  CONTINUE
C       PROBLEM (1)
C  GENERATING FUNCTION OF CHEBYSHEV POLYNOMIALS OF FIRST KIND.
        F=(1.0-T*T)/(1.0-2.0*T*P+T*T)
        RETURN
C       PROBLEM (2)
    20  CONTINUE
C       APPLY THE VARIABLE TRANSFORMATION
        Q=(1.0-P)/(1.0+P)
        F=(1.0-T*T)*(1.0+Q)*0.5/((1.0-T)**2+(1.0+T)**2*Q)
        RETURN
        END
```

This is an example of the rational function $g(x)$ expanded with $\{T_k((1-x)/(1+x))\}$.

k	t=1/4	t=1/2	t=3/4
0	1.000000	1.000000	1.000000
1	0.250000	0.500000	0.750000
2	0.062500	0.250000	0.562500
⋮	⋮	⋮	⋮
15		0.000031	0.013363
16		0.000015	0.010023
17		0.000008	0.007517
⋮		⋮	⋮
31			0.000134
32			0.000100
33			0.000075
⋮			⋮
62			0.000000
Number of terms	15	31	63
Estimated value of error	0.195E-06	0.351E-06	0.811E-06

Note: The number k means the order of the output data.

(1987.06.03) (1987.08.07) (1987.08.08)

FCOSCS/D, FCOSOS/D, FSINOS/D

(Cosine series expansion of an even function given in a closed interval $(0, \pi)$) (FCOSCS/D)

(Cosine series expansion of an even function given in an open interval $(0, \pi)$) (FCOSOS/D)

(Sine series expansion of an odd function given in an open interval $(0, \pi)$) (FSINOS/D)

Fourier Cosine Series of Even Function Defined on The Closed Interval $(0, \pi)$ (FCOSCS/D)

Fourier Cosine Series of Even Function Defined on The Open Interval $(0, \pi)$ (FCOSOS/D)

Fourier Sine Series of Function Defined on The Open Interval $(0, \pi)$ (FSINOS/D)

Programmed by	Tatsuo Torii, December 1978
Format	Subroutine language: FORTRAN; size: 112, 114, 115, 117, 91, and 93 lines respectively

(1) Outline

If a function $f(t)$ of period 2π is even or odd, $f(t)$ is to be given only in a half period $[0, \pi]$. If the function $f(t)$ is to be expanded to cosine series, the end point of the interval may or may not be used as the sample point. The former method is FCOSC/D, and the latter one is FCOSO/D. For the expansion of sine series, the end point is not used as the sample point.

If the function $f(t)$ is input, the number of terms to be expanded is automatically decided by a required precision, and Fourier coefficients are output. This calculation method is efficient because it is based on the high-speed cosine (sine) transformation using the mid-point formula.

(2) Directions

CALL FCOSCS/D(F, EPSA, EPSR, NMIN, NMAX, A, N, ERR, ICON)

CALL FCOSOS/D(F, EPSA, EPSR, NMIN, NMAX, A, N, ERR, ICON)

CALL FSINOS/D(F, EPSA, EPSR, NMIN, NMAX, A, N, ERR, ICON)

Argument	Type and kind (*1)	Attribute	Content
F	Real type Function subprogram	Input	The user should define the periodic function of one variable (even or odd function) as a function subprogram. The domain of this function can be a closed interval [0, π] for FCOSC/D, and and an open interval (0, π) for FCOSO/D and FSINO/D.
EPSA EPSR	Real type	Input	Error bound of Fourier series to be found. EPSA≧0 is the required precision for an absolute error, and EPSR≧0 is the required precision for a relative error.
NMIN NMAX	Integer type	Input	Lower and upper bounds on the number of terms to be expanded. 0≦NMIN≦NMAX≦1025 for FCOSC/D. 0≦NMIN≦NMAX≦1023 for FCOSO/D and FSINO/D.
A	Real type One-dimensional array	Output	Size of array A ≧ NMAX. N Fourier coefficients are stored on A in the order of number. The number of samples used is also N. N is as follows:
N	Integer type		2^n+1 for FCOSC/D 2^n-1 for FCOSO/D and FSINO/D. For the restriction on the number of samples, the priority of NMAX is higher than that of NMIN.
ERR	Real type	Output	Estimated absolute error of obtained Fourier series.

Argument	Type and kind (*1)	Attribute	Content
ICON	Integer type	Output	If ICON=0, the error is normal in the following sense: If the Fourier series of degree N for the input function $f(t)$ is $P_N(t)$, $\lvert f(t)-P_N(t) \rvert \leq \max\{EPSA, EPSR*\lVert f \rVert\}$ Where $\lVert f \rVert = \max_{0\leq j\leq N} \lvert f(\pi/N\ j) \rvert$ ICON=10000: $P_N(t)$ does not satisfy the above conditions because the required precision is too severe. However, it is within the limit of a calculation error. The error can be assumed to be normal. ICON=20000: Abnormal. The required precision cannot be obtained at N≦NMAX. ICON=30000: Parameter error.

*1 For double precision subroutines, all real types should be double precision real types.

(3) Performance

If the sampling time required for the input function $f(t)$ is omitted, the computation time is the same as with fast cosine (sine) transformation.

(4) Calculation method

This is the fast cosine (sine) transformation based on the trapezoidal rules by the successive approximation. However, FCOSO is corrected so that it does not use the end point of an interval $(0, \pi)$ as a sampling point.

The error of the obtained Fourier series is estimated by the sum of absolute coefficient values of the last two terms. The bound of propagation error of round off error is evaluated with $16u \lVert f \rVert$ by assuming the minimum unit of mechanical computer precision as u. In the program, the FUNCTION subprogram AMACH is referred to as u.

This subroutine contains an integer type one-dimensional array of size 256 for the bit reverse of the samples of $f(t)$ and a real type one-dimensional array of size 511 for the trigonometric function table. If this routine is called, these constant tables are calculated for the first

time only. If the size of the constant table is doubled, the upper bound of number of samples can be increased twice.

(5) Example of use

1. Example of cosine series expansion of an even function on the closed interval of $[0, \pi]$

Check by generating function of cosine function

$$f(\theta)=\frac{1-t^2}{1-2t\cos\theta+t^2}$$

$$=1+2\sum_{n=1}^{\infty}t^n\cos n\theta$$

The following shows the program when $t=0.5$ is specified.

```
C       EXAMPLE FOR SUBROUTINE FCOSCS
        DIMENSION A(257)
        EXTERNAL F
        COMMON T
        T=0.5
        EPSA=1.0E-5
        EPSR=EPSA
        NMAX=257
        NMIN=0
        CALL FCOSCS(F,EPSA,EPSR,NMIN,NMAX,A,N,ERR,ICON)
        WRITE(6,600) N,ERR,ICON,(A(I),I=1,N)
  600   FORMAT(1HO,4X,2HN=,I3,5X,4HERR=,E10.3,5X,
     *         5HICON=,I5/1HO,4X,7HARRAY A/(1H ,4F15.06))
        STOP
        END

        FUNCTION F(P)
        COMMON T
        F=(1.0-T*T)/(1.0-2.0*T*COS(P)+T*T)
        RETURN
        END
```

2. Example of sine series expansion of an odd function given in an open interval of $(0, \pi)$

122

$$\frac{\sin\theta}{1-2t\cos\theta+t^2}=\sum_{k=1}^{\infty}t^{k-1}\sin k\theta$$

```
C       EXAMPLE FOR SUBROUTINE FSINOS
        DIMENSION A(257)
        EXTERNAL F
        COMMON T
        T=0.5
        EPSA=1.0E-5
        EPSR=EPSA
        NMAX=257
        NMIN=0
        CALL FSINOS(F,EPSA,EPSR,NMIN,NMAX,A,N,ERR,ICON)
        WRITE(6,600) N,ERR,ICON,(A(I),I=1,N)
  600   FORMAT(1HO,4X,2HN=,I3,5X,4HERR=,E10.3,5X,
     *         5HICON=,I5/1HO,4X,7HARRAY A/(1H ,4F15.06))
        STOP
        END

        FUNCTION F(P)
        COMMON T
        F=SIN(P)/(1.0-2.0*T*COS(P)+T*T)
        RETURN
        END
```

3. Example of cosine series expansion of an even function given in an open interval $(0, \pi)$

If the end point of the interval cannot be used as the sample point, this routine can be used. If the even function

$$f(\theta)=\frac{\alpha}{2}\cdot\frac{1+\tan^2\frac{\theta}{2}}{\alpha^2+\tan^2\frac{\theta}{2}}$$

is extended to cosine series,

$$=\sum_{k=0}^{\infty}{}'\left(\frac{1-\alpha}{1+\alpha}\right)^k\cos k\theta$$

is obtained. The following shows the program for the expansion of $f(\theta)$ on $(0, \pi)$ setting α=1/3.

```
C       EXAMPLE FOR SUBROUTINE FCOSOS
        DIMENSION A(257)
        EXTERNAL F
        COMMON ALPHA
        ALPHA=1.0/3.0
        EPSA=1.0E-5
        EPSR=EPSA
```

```
      NMAX=257
      NMIN=0
      CALL FCOSOS(F,EPSA,EPSR,NMIN,NMAX,A,N,ERR,ICON)
      WRITE(6,600) N,ERR,ICON,(A(I),I=1,N)
  600 FORMAT(1HO,4X,2HN=,I3,5X,4HERR=,E10.3,5X,
     *       5HICON=,I5/1HO,4X,7HARRAY A/(1H ,4F15.06))
      STOP
      END

      FUNCTION F(P)
      COMMON ALPHA
      Q=TAN(P*0.5)**2
      F=ALPHA*0.5*(1.0+Q)/(ALPHA**2+Q)
      RETURN
      END
```

The results are shown as below.

k	Problem 5.1	Problem 5.2	Problem 5.3	k	Problem 5.1	Problem 5.2	Problem 5.3
0	2.000000	——	1.000000	⋮	⋮	⋮	⋮
1	1.000000	1.000000	0.500000	30	0.000000	0.000000	0.000000
2	0.500000	0.500000	0.250000	31	0.000000	0.000000	——
⋮	⋮	⋮	⋮	32	0.000000	——	——
14	0.000121	0.000122	0.000061	Estimated Error	0.715 E-06	0.317 E-06	0.351 E-06
15	0.000061	0.000061	0.000031				
16	0.000031	0.000031	0.000015				

Note: The number k represents the order of output data.

(1987.05.20) (1987.08.08)

FCOSMS/D, FSINMS/D

(Fast Fourier cosine transform based on the midpoint rule) (FCOSMS/D)

(Fast Fourier sine transform based on the midpoint rule) (FSINMS/D)

Fast Fourier Cosine Transform Based on The Midpoint Rule (FCOSMS/D)

Fast Fourier Sine Transform Based on The Midpoint Rule (FSINMS/D)

Programmed by	Tatsuo Torii; December 1978
Format	Subroutine language; FORTRAN Size; 165, 166, 165, and 166 respectively

(1) Outline

A half period of function X(t) with the period 2π is equally divided into N parts as below:

$$X_{j+\frac{1}{2}}=X\left[\frac{\pi}{N}\left(j+\frac{1}{2}\right)\right],\ 0\leq j<N,\ N=2^n$$

When X(t) is an even function:

$$B_k=\sum_{0\leq j<N} X_{j+\frac{1}{2}}\cos\frac{\pi}{N}k\left(j+\frac{1}{2}\right),\ 0\leq k<N$$

is calculated. When X(t) function is an odd function:

$$B_k=\sum_{0\leq j<N} X_{j+\frac{1}{2}}\sin\frac{\pi}{N}k\left(j+\frac{1}{2}\right),\ 0<k\leq N$$

is calculated.

(2) Directions

Before these subroutines are called, it is needed to perform calculation of the trigonometric function table by TRIGQP or TRIGQD and to arrange input data in binary reverse order by BTREV or BTRVD. More concrete, call such subroutines as in the table below:

For single precision	For double precision
CALL TRIGQP(W, MW, ICON) CALL BITREV(X, MX, ICON)	CALL TRIGQD(W, MW, ICON) CALL BITRVD(X, MX, ICON)

Then, call a target subroutine:

```
CALL FCOSMS/D(X, MX, LX, W, MW, ICON)

CALL FSINMS/D(X, MX, LX, W, MW, ICON)
```

Argument	Type and kind (*1)	Attribute	Content
X	Real type One-dimensional array	Input/output	Size of array X $\geqq 2^{MX}$. Number of input samples $=2^{MX}$. LX specifies the beginning address of input data on array X. That is, $X(LX+1), X(LX+2), \cdots, X(LX+2^{MX})$ is input. And output is written over this. FCOSMS/D: $B_{k-1}=X(LX+k)$ FSINMS/D: $B_{2^{MX}-k+1}=X(LX+k)$ MX≧1, LX≧0
MX	Integer type		
LX	Integer type		
W	Real type One-dimensional array	Input	Size of array W $\geqq 2^{MW}-1$. MW≧MXMW
MW	Integer type		
ICON	Integer type	Output	ICON = 0: Normal. ICON = 30000: Parameter error

*1 For double precision subroutines, real types are all assumed as double precision real types.

(3) Performance

The number of real multiplications needed for N-term cosine (sine) transform is $N\log_2 N (N=2^n)$. Output data is written over the input data. The algorithm is stable.

(4) Calculation method

The algorithm [3] of fast Fourier cosine (sine) transform based on the midpoint rule has been arranged so that output data is written over input data. Because it uses not only reality but also symmetricity of input data, the number of operations and the work area reduced to the half of those fast Fourier transform of real data.

(5) Example

When the number of input data items is 2^M, a trigonometric function table (TRIGQP) of the size 2^{M-1}, at least, must have been calculated. If input data queues up in order of number, rearrange it in binary reverse order, and then call this subroutine.

The beginning address of input data can be chosen by the parameter LX. The reason of this form is to use this subroutine for Chebyshev series expansion and fast Fourier cosine transform based on the trapezoidal rule by taking appropriate values LX. Only for cosine transform for the data $X(1), X(2), \cdots, X(2^M)$, it is enough to make LX=0.

For a cosine transform test, we use the following two problems whose analytical solutions are known:

Problem (1)

$$\sum_{0\leq k\leq N}{}' \cos k\theta = \sin\left(N+\frac{1}{2}\right)\theta \Big/ \left(2\sin\frac{\theta}{2}\right)$$

Problem (2)

$$\sum_{0\leq k\leq N}{}' (k+1)\cos k\theta = \left\{2(N+1)\sin\frac{\theta}{2}\sin\left(N+\frac{1}{2}\right)\theta + \cos N\theta - 1\right\} \Big/ \left(4\sin^2\frac{\theta}{2}\right)$$

If a right hand side function is sampled at sample point $\theta_j = \pi/N\,(j+1/2)$ and input them, each Fourier coefficient (N/2 times) is generated. That is, when the samples

$$X_{j+\frac{1}{2}} = \frac{1}{2}(-1)^j \cot\frac{\pi}{2N}\left(j+\frac{1}{2}\right),\quad 0\leq j<N$$

are input, Fourier coefficients are obtained as follows:

$$\frac{2}{N}B_k = 1,\quad 0\leq k<N$$

Similarly, when the samples

$$X_{j+\frac{1}{2}} = \{(-1)^j(N+1)\sin\theta_j - 1\} \Big/ \left(4\sin^2\frac{\theta_j}{2}\right)$$

are input;

$$\frac{2}{N}B_k = k+1$$

128

. 0 LESS-EQUAL k<N are obtained. The program which verifies the above operation is shown as follows:

```
C       TEST PROBLEMS OF SUBROUTINE FCOSMS
        DIMENSION X(128),C(127)
        EXTERNAL F
        COMMON L,N,J
        M=7
        N=2**M
        CALL TRIGQS(C,M,ICON)
        DO 30 L=1,2
        DO 10 J=1,N
        P=(FLOAT(J)-0.5)/FLOAT(N)
        X(J)=F(P)
   10   CONTINUE
        CALL BITREV(X,M,ICON)
        LX=0
        CALL FCOSMS(X,M,LX,C,M,ICON)
        CT=2.0/FLOAT(N)
        DO 20 I=1,N
        X(I)=X(I)*CT
   20   CONTINUE
        WRITE(6,600) L,N,(X(I),I=1,N)
   30   CONTINUE
  600   FORMAT(////8X,9HPROBLEM (,I1,1H),4X,2HN=,I3/1X/
     *         (1H ,4F15.06))
        STOP
        END

        FUNCTION F(P)
        COMMON L,N,J
        SGN=1.0
        IF(MOD(J,2).EQ.0) SGN=-SGN
        GO TO (10,20),L
C       PROBLEM (1)
   10   F=0.5*SGN*COTHP(P)
        RETURN
C       PROBLEM (2)
   20   F=0.25*(SGN*FLOAT(N+1)*SINHP(P+P)-1.0)/SINHP(P)**2
        RETURN
        END
```

The calculation results of cosine transform based on the midpoint rule for the two problems described above are shown below.

k	problem (1)	problem (2)
0	1.000000	0.999998
1	1.000000	1.999999
2	1.000000	2.999998
3	1.000000	3.999999
⋮	⋮	⋮
125	1.000000	125.999998
126	1.000000	126.999999
127	1.000000	127.999999

Note : k represents the output order of the data.

Next, for a sine transform test, the following two problems are used:

Problem (1)

$$\sum_{1\leq k\leq N} \sin k\theta = \left\{ \cos\frac{\theta}{2} - \cos\left(N+\frac{1}{2}\right)\theta \right\} / \left(2\sin\frac{\theta}{2}\right)$$

Problem (2)

$$\sum_{1\leq k\leq N} k\sin k\theta = \left\{ \sin N\theta - 2N\sin\frac{\theta}{2}\cos\left(N+\frac{1}{2}\theta\right)\right\} / \left(4\sin^2\frac{\theta}{2}\right)$$

The right hand side function is sampled at the points

$$\theta_j = \frac{\pi}{N}\left(j+\frac{1}{2}\right)$$

, and samples

$$X_{j+\frac{1}{2}} = \frac{1}{2}\left\{ \cot\frac{\theta_j}{2} + (-1)^j \right\}, 0\leq j<N$$

are input. Then, Fourier coefficients

$$\frac{2}{N}B_k = 1, 1\leq k<N$$

$$\frac{2}{N}B_N = 2$$

are obtained. When

$$X_{j+\frac{1}{2}} = (-1)^j\left\{ 1+2N\sin^2\frac{\theta_j}{2}\right\} / \left(4\sin^2\frac{\theta_j}{2}\right)$$

are input, then

13.

$$\frac{2}{N}B_k=k, 1\leq k<N$$

$$\frac{2}{N}B_N=2N$$

are obtained.

```
C       TEST PROBLEMS OF SUBROUTINE FSINMS
        DIMENSION X(128),C(127)
        EXTERNAL F
        COMMON L,N,J
        M=7
        N=2**M
        CALL TRIGQS(C,M,ICON)
        DO 30 L=1,2
        DO 10 J=1,N
        P=(FLOAT(J)-0.5)/FLOAT(N)
        X(J)=F(P)
   10   CONTINUE
        CALL BITREV(X,M,ICON)
        LX=0
        CALL FSINMS(X,M,LX,C,M,ICON)
        CT=2.0/FLOAT(N)
        DO 20 I=1,N
        X(I)=X(I)*CT
   20   CONTINUE
        WRITE(6,600) L,N,(X(I),I=1,N)
   30   CONTINUE
  600   FORMAT(////8X,9HPROBLEM (,I1,1H),4X,2HN=,I3/1X/
       *       (1H,4F15.06))
        STOP
        END

        FUNCTION F(P)
        COMMON L,N,J
        SGN=1.0
        IF(MOD(J,2).EQ.0) SGN=-SGN
        GO TO (10,20),L
C       PROBLEM (1)
   10   F=0.5*(COTHP(P)+SGN)
        RETURN
C       PROBLEM (2)
   20   F=0.25*(1.0/SINHP(P)**2+FLOAT(N+N))*SGN
        RETURN
        END
```

The calculation results of fast Fourier sine transform based on the midpoint rule are as follows:

k	problem (1)	problem (2)
1	2.000000	255.999999
2	1.000000	126.999999
3	1.000000	125.999999
4	1.000000	124.999999
⋮	⋮	⋮
126	1.000000	3.000000
127	1.000000	2.000000
128	1.000000	1.000000

Note : k represents the output order of the data.

Bibliography

1) J. W. Cooley & J. W. Tukey; "An Algorithm for Machine Calculation of Complex Fourier Series", Mathematics of Computation, Vol.19, pp.297-301 (1965)
2) Ichizo Ninomiya; "Real Fast Fourier Analysis (Synthesis), Reverse Binary Transformation etc.", Library Program User's Guide, pp.95-106, Nagoya University Computer Center (1978)
3) Tatsuo Torii; "Fast Fourier Sine and Cosine Transform and Its Application to Numerical Integration," Information Processing, Vol.15, pp.670-679 (1974)
4) Saburo Makinouchi and Tatsuo Torii; "Numerical Analysis," pp.293-303, Ohm-sha (1975)
5) C. W. Clenshaw & A. R. Curtis; "A Method for Numerical Integration on an Automatic Computer", Numerishe Mathematik, Vol.2, pp.197-205 (1960)
6) Ichizo Ninomiya; "Trigonometric Function for Argument $\pi/2\,x$," Library Program User's Guide, p.12, Nagoya University Computer Center (1978)
7) FACOM FORTRAN SSL II User's Guide, p.247, Fujitsu Limited (1978)
8) Tatsuo Torii; "Fourier Transform Subroutine Package," Nagoya University Computer Center News, Vol.10, No.1, p.24 (1979), Vol.10, No.2, and p.138 (1979)

(1987.05.19) (1987.08.08)

132

FCOSTS/D, FSINTS/D

(Fast Fourier Cosine Transform Based on The Trapezoidal Rule (FCOSTS/D))

(Fast Fourier Sine Transform Based on The Trapezoidal Rule (FSINTS/D))

Fast Fourier Cosine Transform Based on The Trapezoidal Rule(FCOSTS/D)

Fast Fourier Sine Transform Based on The Trapezoidal Rule(FSINTS/D)

Programmed by	Tatsuo Torii, July 1978
Format	Subroutine Language: FORTRAN; Size: 64, 65, 33, and 34 lines respectively

(1) Outline

Assume that N+1 samples that can be obtained by dividing a half period of the function X(t) with the period 2π into N parts are represented by

$$X_j=X(\frac{\pi}{N}j),0\leq j\leq N,N=2^n$$

If X(t) is an even function, discrete cosine coefficients are given by

$$C_k=\sum_{0\leq j\leq N}{}''X_j\cos\frac{\pi}{N}kj,0\leq k\leq N$$

Where Σ'' means the summation multiplying 1/2 to the first and last terms.

If X(t) is an odd function, discrete sine coefficients

$$C_k=\sum_{0<j<N}X_j\sin\frac{\pi}{N}kj,0<k<N$$

are obtained using N-1 samples. Even the inverse transformation can be executed with the same program.

(2) Directions

```
CALL FCOSTS/D(X, MX, W, MW, ICON)

CALL FSINTS/D(X, MX, W, MW, ICON)
```

Argument	Type and kind (*1)	Attribute	Content
X	Real type One-dimensional array	Input/output	FCOST/D: If $X(j+1)=X_j, 0\leq j\leq 2^{MX}$ are input, $X(j+1)=C_j$ are output. FSINT/D: If $X(1)=X_0=0, X(j+1)=X_j, 1\leq j<2^{MX}$ are input, $X(1)=C_0=0, X(j+1)=C_j$ are output.
MX	Integer type	Input	MX≧1
W	Real type One-dimensional array	Input	A trigonometric function table should be provided on W in advance by using TRIGQP/TRIGQD(W, MW, ICON). The number of data items $2^{MW}-1$ of the trigonometric function table should be specified by MW. Size of array W should be $\geq 2^{MW}-1$. MW≧MX-1
MW	Integer type		
ICON	Integer type	Output	ICON=0: Normal. ICON=30000: Parameter error.

*1 For double precision subroutines, all real types should be double precision real types.

(3) Performance

Performance is almost same to the fast cosine(sine) transformation based on the middle point formula.

(4) Example

If the even function $\sin(N+1/2)\theta/2\sin\theta/2$ is sampled at the point $\theta_j=\pi j/N$ as

$$X_j=\begin{cases}N-1/2, & j=0\\ (-1)^j/2, & 1\leq j\leq N\end{cases}$$

, the solution is given by

$$\frac{2}{N}C_k=\begin{cases}1, & 0\leq k<N\\ 2, & k=N\end{cases}$$

If the even function

$$\{2(N+1)\sin\frac{\theta}{2}\sin(N+\frac{1}{2})\theta+\cos N\theta-1\}/4\sin^2\frac{\theta}{2}$$

is sampled as

$$X_j = \begin{cases} (N^2+3N+1)/2, & j=0 \\ (N+1)/2, & j \text{ is an even number, and } 0<j\leq N. \\ -(N+1+\mathrm{cosec}^2\theta_j/2)/2, & j \text{ is an odd number, and } 1\leq j<N. \end{cases}$$

We get

$$\frac{2}{N}C_k=\begin{cases} k+1, & 0\leq k<N \\ 2(N+1), & k=N \end{cases}$$

We can confirm this by the cosine transformation based on the trapezoidal rule.

```
C       TEST PROBLEMS OF SUBROUTINE FCOSTS
        DIMENSION X(129),W(63)
        EXTERNAL F
        COMMON L,AN,J
        M=7
        MW=6
        N=2**M+1
        AN=FLOAT(N-1)
        CALL TRIGQS(W,MW,ICON)
        DO 30 L=1,2
        DO 10 J=1,N
        X(J)=F(FLOAT(J-1)/AN)
   10 CONTINUE
        CALL FCOSTS(X,M,W,MW,ICON)
        CT=2.0/AN
        DO 20 I=1,N
        X(I)=X(I)*CT
   20 CONTINUE
        WRITE(6,600) L,N,(X(I),I=1,N)
   30 CONTINUE
  600 FORMAT(////8X,9HPROBLEM (,I1,1H),4X,2HN=,I3/1X/
     *       (1H,4F15.06))
        STOP
        END

        FUNCTION F(P)
        COMMON L,AN,J
        SGN=1.0
```

```
      IF(MOD(J,2).EQ.0) SGN=-SGN
      GO TO (10,20),L
C     PROBLEM (1)
   10 F=AN+0.5
      IF(J.EQ.1) RETURN
      F=SGN*0.5
      RETURN
C     PROBLEM (2)
   20 IF(J.EQ.1) GO TO 21
      IF(SGN.LT.0.0) GO TO 22
      F=(AN+1.0)*0.5
      RETURN
   21 F=(AN*AN+3.0*AN+1.0)*0.5
      RETURN
   22 F=-(AN+1.0)*0.5-0.5/SINHP(P)**2
      RETURN
      END
```

Example of fast cosine transform values based on trapezoidal rule

k	Problem (1)	Problem (2)
0	1.000000	1.000000
1	1.000000	2.000000
2	1.000000	3.000000
3	1.000000	4.000000
⋮	⋮	⋮
125	1.000000	125.000000
126	1.000000	125.999999
127	1.000000	127.999999
128	2.000000	257.999999

Note: The number k shows the order of the output data.

The next example is sine transformation. If the odd function

$\{\cos\theta/2-\cos(N+1/2)\theta\}/2\sin\theta/2$ is sampled at the point $\theta_j=\pi/Nj, 1\leq j<N$, and N-1 data items.

$X_j = 0$, for even number j .

$X_j = \cot\theta_j/2$, j is an odd number.

are input, all of these Fourier coefficients are 1. That is,

$$\frac{2}{N}C_k=1, 1\leq k\leq N-1$$

. If N-1 samples for $\{(N+1)\sin N\theta-N\sin(N+1)\theta\}/4\sin^2\theta/2$

$$X_j=\frac{1}{2}(-1)^{j-1}N\cot\frac{\theta_j}{2},\ 1\le j\le N-1$$

are input,

$$\frac{2}{N}C_k=k,1\le k\le N-1$$

are obtained.

```
C       TEST PROBLEMS OF SUBROUTINE FSINTS
        DIMENSION X(128),W(63)
        EXTERNAL F
        COMMON L,AN,J
        M=7
        MW=6
        N=2**M-1
        AN=FLOAT(N+1)
        CALL TRIGQS(W,MW,ICON)
        DO 30 L=1,2
        X(1)=0.0
        DO 10 J=1,N
        X(J+1)=F(FLOAT(J)/AN)
   10   CONTINUE
        CALL FSINTS(X,M,W,MW,ICON)
        CT=2.0/AN
        DO 20 I=1,N
        X(I+1)=X(I+1)*CT
   20   CONTINUE
        WRITE(6,600) L,N,(X(I+1),I=1,N)
   30   CONTINUE
  600   FORMAT(////8X,9HPROBLEM (,I1,1H),4X,2HN=,I3/1X/
     *       (1H,4F15.06))
        STOP
        END

        FUNCTION F(P)
        COMMON L,AN,J
        SGN=1.0
        IF(MOD(J,2).NE.0) SGN=-SGN
        GO TO (10,20),L
C       PROBLEM (1)
   10   F=0.0
        IF(SGN.GT.0.0) RETURN
        F=COTHP(P)
        RETURN
C       PROBLEM (2)
   20   F=0.5*SGN*COTHP(P)*(-AN)
        RETURN
        END
```

Example of fast sine transform based on trapezoidal rule

k	Problem (1)	Problem (2)
1	1.000000	1.000000
2	1.000000	2.000000
3	1.000000	3.000000
4	1.000000	4.000000
⋮	⋮	⋮
125	1.000000	124.999996
126	1.000000	125.999998
127	1.000000	126.999998

Note: The number k shows the order of the output data.

(1987.05.28) (1987.08.08)

FFT2DC/B and FFT3DC/B (2- and 3-Dimensional Complex Fast Fourier Transform)

2- and 3-Dimensional Complex Fast Fourier Transform

Programmed by	Ichizo Ninomiya, May 1982
Format	Subroutine Language: FORTRAN77; Size: 21, 22, 30, and 31 lines respectively

(1) Outline

FFT2DC/B is a subroutine for 2-dimensional complex fast Fourier transform. FFT3DC/B is a subroutine for 3-dimensional complex fast Fourier transform.

The outline of the algorithm is given only for two dimensions. If function values X_{rs}; $r=0,1,\cdots, N_1-1$; $s=0,1,\cdots, N_2-1$ at the $N_1(=2^{M_1})\times N_2(=2^{M_2})$ equipartition mesh points of the fundamental rectangle of period of the two-dimensional periodic complex value function (X_{00} is a value at the origin) are given, the Fourier transform is given by

$$C_{kl}=\frac{1}{N_1N_2}\sum_{r=0}^{N_1-1}\sum_{s=0}^{N_2-1}X_{rs}\,e^{-\frac{2\pi ikr}{N_1}}\,e^{-\frac{2\pi ils}{N_2}},\quad k=0,1,\cdots,N_1-1;\ l=0,1,\cdots,N_2-1$$

This expression is called the forward transformation. Conversely, obtaining a function value

$$X_{rs}=\sum_{k=0}^{N_1-1}\sum_{l=0}^{N_2-1}C_{kl}\,e^{\frac{2\pi ikr}{N_1}}\,e^{\frac{2\pi ils}{N_2}},\quad r=0,1,\cdots,N_1-1;\ s=0,1,\cdots,N_2-1$$

at the power of 2 equipartition mesh points of the fundamental rectangle of period of a periodic function having C_{kl} as periodic components is called the inverse transformation.

(2) Directions

```
CALL FFT2DC/B(A, KA, M, INV, W, ILL)

CALL FFT3DC/B(A, KA, LA, M, INV, W, ILL)
```

Argument	Type and kind (*1)	Attribute	Content
A	Complex type Two-dimensional array (Three-dimensional array)	Input/output	Forward transformation: If X is input, $N_1N_2C(N_1N_2N_3C)$ is output. Inverse transformation: If C is input, X is output. Size $2^{M(1)}\times 2^{M(2)}$ ($2^{M(1)}\times 2^{M(2)}\times 2^{M(3)}$)
KA	Integer type	Input	Value of the first subscript in the array declaration of A. KA≧$2^{M(1)}$
LA	Integer type	Input	Value of the second subscript in the array declaration of A. LA≧$2^{M(2)}$
M	Integer type One-dimensional array	Input	$2^{M(1)}$, $2^{M(2)}$, $2^{M(3)}$ represent the number of equipartitions in each direction of axis. M(1)>1, M(2)>1, and M(3)>1
INV	Integer type	Input	Forward transformation is executed at INV=0. Inverse transformation is executed at INV=1.
W	Complex type One-dimensional array	Work area	Size $2^{M(2)}$ for two dimensions. Size $max(2^{M(2)},2^{M(3)})$ for three dimensions.
ILL	Integer type	Output	ILL=0: Normal termination. ILL=30000: Argument error.

*1 For FFT2DB and FFT3DB, all complex types should be changed to double precision complex types.

(3) Example

Function values at 128×128 equipartition mesh points of fundamental square of period $[0,1]^2$ of complex periodic function

$$f(x,y)=(1+2ie^{2\pi ix}+3e^{4\pi ix})(-1-2ie^{2\pi iy})$$

are obtained by the inverse transformation, and the forward transformation is applied to them.

```
COMPLEX*8 A,B,C,S
DIMENSION A(128,128),B(128,128),C(128),S(2),M(2)
```

```
      N=128
      C(1)=1.0
      C(2)=(0.,2.)
      C(3)=3.
      S(1)=-1.
      S(2)=(0.,-2.)
      DO 10 J=1,N
      DO 10 I=1,N
      A(I,J)=0.
   10 B(I,J)=0.
      DO 20 J=1,2
      DO 20 I=1,3
      A(I,J)=C(I)*S(J)
   20 B(I,J)=A(I,J)
      KA=N
      M(1)=7
      M(2)=7
      INV=1
      CALL FFT2DC(A,KA,M,INV,C,ILL)
      INV=0
      CALL CLOCKM(I0)
      CALL FFT2DC(A,KA,M,INV,C,ILL)
      CALL CLOCKM(I1)
      IT=I1-I0
      D=1./FLOAT(N)**2
      DO 30 J=1,N
      DO 30 I=1,N
   30 EM=AMAX1(CABS(A(I,J)*D-B(I,J)),EM)
      WRITE(6,600) IT,ILL,EM
  600 FORMAT(10X,'TIME =',I7,'MS',2X,'ILL=',I6,2X,'EM=',E11.3)
      STOP
      END
```

(4) Summary

An output given by the forward transformation is not the Fourier transform itself but $N_1N_2(N_1N_2N_3)$ times of it. Refer to the explanation and the example of use of the argument A and the explanation of the subroutine FFTC.

(1987. 05. 11) (1987. 08. 08)

FFT2DR/D and FFT3DR/D (2- and 3-Dimensional Real Fast Fourier Analysis and Synthesis)

2- and 3-Dimensional Real Fast Fourier Analysis and Synthesis

Programmed by	Ichizo Ninomiya, May 1982
Format	Subroutine Language: FORTRAN77; Size: 28, 29, 40, and 41 lines respectively

(1) Outline

FFT2DR/D is a subroutine for 2-dimensional real fast Fourier analysis and synthesis. FFT3DR/D is a subroutine for 3-dimensional real fast Fourier analysis and synthesis.

The outline of the algorithm is explained only for the case of two dimensions. If function values $F_{rs}; r=0, 1, \cdots, N_1-1; s=0, 1, \cdots, N_2-1$ (F_{00} is a value at the origin) at $N_1(=2^{M1}) \times N_2(=2^{M2})$ equipartition mesh points of fundamental rectangle of period of two-dimensional real periodic function are given, the sine (C) and cosine (S) elements are given by

$$\begin{Bmatrix} C \\ S \end{Bmatrix}_{kl} = \frac{\varepsilon_1 \varepsilon_2}{N_1 N_2} \sum_{r=0}^{N_1-1} \sum_{s=0}^{N_2-1} F_{rs} \begin{Bmatrix} \cos \frac{2\pi kr}{N} \\ \sin \frac{2\pi kr}{N_1} \end{Bmatrix} \begin{Bmatrix} \cos \frac{2\pi ls}{N_2} \\ \sin \frac{2\pi ls}{N_2} \end{Bmatrix}, \quad k=0,1,\cdots,N_1/2;$$

$$l=0,1,\cdots,N_2/2$$

Where,

$$\varepsilon_1 = \begin{cases} 2, & 0<k<N_1/2 \\ 1, & k=0, N_1/2 \end{cases}$$

$$\varepsilon_2 = \begin{cases} 2, & 0<l<N_2/2 \\ 1, & l=0, N_2/2 \end{cases}$$

and

$$\begin{cases} CS_{kl}=0, & l=0,N_2/2 \\ SC_{kl}=0, & k=0,N_1/2 \\ SS_{kl}=0, & k=0,N_1/2 \text{ or } l=0,N_2/2 \end{cases}$$

The calculation described above is called Fourier analysis. Conversely, obtaining a function value

$$F_{rs}=\sum_{k=0}^{N_1/2}\left\{\cos\frac{2\pi kr}{N_1}\sum_{l=0}^{N_1/2}\left(CC_{kl}\cos\frac{2\pi ls}{N_2}+CS_{kl}\sin\frac{2\pi ls}{N_2}\right)\right.$$

$$\left.+\sin\frac{2\pi kr}{N_1}\sum_{l=0}^{N_1/2}\left(SC_{kl}\cos\frac{2\pi ls}{N_2}+SS_{kl}\sin\frac{2\pi ls}{N_2}\right)\right\},r=0,1,\cdots,N_1-1;s=0,1,\cdots,N_2-1$$

at the equipartition mesh points of fundamental rectangle of period from cosine and sine elements is called Fourier synthesis.

(2) Directions

CALL FFT2DR/D(A, KA, M, INV, W, ILL)

CALL FFT3DR/D(A, KA, LA, M, INV, W, ILL)

Argument	Type and kind (*1)	Attribute	Content
A	Real type Two-dimensional array (Three-dimensional array)	Input/output	Fourier analysis: If F is input, cosine and sine elements are output. The order of storing the outputs is the direct product of the case of one dimension. For instance, CS_{IJ} is stored in A(I+1, 2**(M(2)-1)+1+J) in the case of two dimensions. Fourier synthesis: If cosine and sine elements are stored in the above order, a function value F is output in natural order. Size $2^{M(1)}\times 2^{M(2)}$ ($2^{M(1)}\times 2^{M(2)}\times 2^{M(3)}$).

Argument	Type and kind (*1)	Attribute	Content
KA	Integer type	Input	Value of the first subscript in the array declaration of A. $KA \geq 2^{M(1)}$
LA	Integer type	Input	Value of the second subscript in the array declaration of A. $LA \geq 2^{M(2)}$
M	Integer type One-dimensional array	Input	$2^{M(1)}, 2^{M(2)}, 2^{M(3)}$ represents the number of equipartitions in each direction of axis. M(1)>1, M(2)>1, and M(3)>1.
INV	Integer type	Input	Fourier analysis is done at INV=0. Fourier synthesis is done at INV=1.
W	Real type One-dimensional array	Work area	Size $2^{M(2)}$ in the case of two dimensions. Size $max(2^{M(2)}, 2^{M(3)})$ in the case of three dimensions.
ILL	Integer type	Output	ILL=0: Normal termination. ILL=30000: Argument error.

*1 For FFT2DD and FFTDD, all real types should be changed to double precision real types.

(3) Example

A function value at 128×128 equipartition mesh points of fundamental square of period $[0, 1]^2$ of periodic function

$$f(x,y)=(1+\cos 2\pi x+2\cos 4\pi x)(-\sin 2\pi x-2\sin 4\pi x)$$

is obtained by Fourier synthesis and applied to Fourier analysis.

```
DIMENSION A(128,128),B(128,128),C(128),S(2),M(2)
N=128
C(1)=1.0
```

```
      C(2)=2.
      C(3)=3.
      S(1)=-1.
      S(2)=-2.
      DO 10 J=1,N
      DO 10 I=1,N
      A(I,J)=0.
   10 B(I,J)=0.
      NH1=N/2+1
      DO 20 J=1,2
      DO 20 I=1,3
      A(I,J+NH1)=C(I)*S(J)
   20 B(I,J+NH1)=A(I,J+NH1)
      KA=N
      M(1)=7
      M(2)=7
      INV=1
      CALL FFT2DR(A,KA,M,INV,C,ILL)
      INV=0
      CALL CLOCKM(I0)
      CALL FFT2DR(A,KA,M,INV,C,ILL)
      CALL CLOCKM(I1)
      IT=I1-I0
      DO 30 J=1,N
      DO 30 I=1,N
   30 EM=AMAX1(ABS(A(I,J)-B(I,J)),EM)
      WRITE(6,600) IT,ILL,EM
  600 FORMAT(10X,'TIME=',I7,'MS',2X,'ILL=',I6,2X,'EM =',E11.3)
      STOP
      END
```

(4) Summary

The order of storing cosine and sine elements is the direct product of the case of one-dimensional real Fourier analysis. Refer to the explanation and the example of use of argument A and the explanation of subroutine FFTR.

(1987.05.19) (1987.08.08)

FFTC/B (Complex fast Fourier analysis)

Complex Fast Fourier Analysis

Programmed by	Ichizo Ninomiya; April 1981
Format	Subroutine language: Assembler, Size: 267 lines each

(1) Outline

When sample value $X_j, j=0,1,\cdots,N-1$ (X_0 is a value in the origin) in N equipartition point of a period of a periodic function is given, the periodic component $C_j, j=0,1,\cdots,N-1$ is given by the following Fourier variable

$$C_j=\sum_{k=0}^{N-1} X_k W^{jk}, j=0,1,\cdots,N-1$$

:

where

$$W=exp\left(\frac{-2\pi i}{N}\right)$$

On the contrary, when periodic component C_j is given, X_j is given by the following inverse transformation:

$$X_j=\sum_{k=0}^{N-1} C_k W^{-jk}, j=0,1,\cdots,N-1$$

This routine is used to perform the above calculation using the complex fast Fourier conversion technique when N is of the form $N=2^M$ is given.

(2) Directions

CALL FFTC/B(A, M, INV, ILL)

Argument	Type and kind (*1)	Attribute	Content
A	Complex type One-dimensional array	Input/output	For forward transformation, X_k is input and C_j is output. For inverse transformation, C_k is input and X_j is output. $C_{j-1}(X_{j-1})$ is output in $A(j)$.
M	Integer type	Input	Used to indicate that the size of array A is 2^M. $M \geq 2$
INV	Integer type	Input	INV = 0 indicates forward transformation and INV = 1 indicates inverse transformation.
ILL	Integer type	Output	ILL = 0: Normal end. ILL = 30000: $M \leq 1$

*1 For FFTB, the complex type should be changed to a double precision complex type.

(3) Performance

Because this routine uses the technique of the radix 4 complex fast Fourier transform and is written in assembly language, it is farst and accurate.

(4) Note

FFTS or FFTD is available for the same purpose as FFTC or FFTB. Note, however, that FFTS and FFTD are a little different from FFTC and FFTB in the meaning of arguments and Fourier transform definitions. FFTS/D requires an work area B as large as input vector A, but FFTC/B does not. Moreover, the latter is faster. So, it is more advantageous to use FFTC/B.

(1987.08.10)

FFTR/FFTRD (Real Fast Fourier Analysis)

Real Fast Fourier Analysis

Programmed by	Ichizo Ninomiya, April 1981
Format	Subroutine language: Assembler; size: 214 lines

(1) Outline

If the values $X_j, j=0,1,\cdots,N-1$ at $N=2^M$ equipartition points of a period of a real periodic function starting from the origin as the left end are input, FFTR/FFTRD calculates the cosine components $C_j, j=0,1,\cdots,N/2$ and sine components $S_j, j=1,2,\cdots,N/2-1$ using the technique of real fast Fourier analysis. Where,

$$C_j=\frac{\varepsilon_j}{N}\sum_{k=0}^{N-1} X_k\cos\frac{2jk\pi}{N},\quad j=0,1,\cdots,N/2$$

$$\varepsilon_0=\varepsilon_{N/2}=1;\varepsilon_j=2,\quad j=2,\cdots,N/2-1$$

$$S_j=\frac{2}{N}\sum_{k=0}^{N-1} X_k\sin\frac{2jk\pi}{N},\quad j=1,2,\cdots,N/2-1$$

(2) Directions

```
CALL FFTR(A, M, ILL)

CALL FFTRD(A, M, ILL)
```

Argument	Type and kind (*1)	Attribute	Content
A	Real type One-dimensional array	Input/output	One-dimensional array containing 2^M elements. If the values at 2^M equipartition points of a period of the periodic function are input sequentially starting from the one at the origin, the sine and cosine components are entered in this order, where each components are entered in natural order. That is, the K-th order cosine components are entered in A(K+1), and the J-th order sine components are entered in A(N/2+J+1).
M	Integer type	Input	Indicates that one period is equally divided into 2^M. $M \geq 0$
ILL	Integer type	Output	If M<0, ILL=30000 is output, and calculation is not executed. Otherwise, calculation is executed, and ILL=0 is output.

*1 For FFTRD, real types should be changed to double precision real types.

(3) Performance

Because this routine is written in the assembly language, and an effort is made to reduce the number of calculations of trigonometric functions, its speed and precision are high.

(4) Calculation method

Unlike Bergland's [1] algorithm, bit reversal rearrangement is executed (calling the subroutine BITREV) in the beginning.

(5) Note

There are many methods for Fourier analysis. Without special conditions, however, real fast Fourier analysis should be used with the number of divisions put in the form of 2^M.

Bibliography

1) G. D. Bergland; "A Fast Fourier Transform Algorithm for Real-Valued Series", Comm. ACM, Vol. 11, pp. 703-710 (1968).

(1987. 08. 10)

FFTRI/FFTRID (Real Fast Fourier Synthesis)

Real Fast Fourier Synthesis

Programmed by	Ichizo Ninomiya, April 1981
Format	Subroutine language; Assembler; size: 196 lines

(1) Outline

If the cosine components C_j, $j=0,1,\cdots,N/2$ and the sine components S_j, $j=1,2,\cdots,N/2-1$ of a real periodic function are input, FFTRI/FFTRID calculates the values X_j, $j=0,1,\cdots,N-1$ at N equipartition points of a period of that function, starting from the origin as the left end, using the technique of real fast Fourier analysis.

Where,

$$X_k=\sum_{j=0}^{N/2}C_j\cos\frac{2\pi jk}{N}+\sum_{j=1}^{N/2-1}S_j\sin\frac{2\pi jk}{N},\quad k=0,1,\cdots,N-1$$

, and N is an integer in the form of $N=2^M$.

(2) Directions

CALL FFTRI(A, M, ILL)

CALL FFTRID(A, M, ILL)

Argument	Type and kind (*1)	Attribute	Content
A	Real type One-dimensional array	Input/output	One-dimensional array containing 2^M elements. If the K-th order cosine components are entered in A(K+1), and the J-th order sine components are entered in A(N/2+J+1), the values at 2^M equipartition points of a period are entered sequentially starting from the one at the origin.
M	Integer type	Input	Indicates that a period is divided into 2^M equal parts. M≧0

Argument	Type and kind (*1)	Attribute	Content
ILL	Integer type	Output	If M<0, ILL=30000 is output, and calculation is not executed. Otherwise, calculation is executed, and ILL=0 is output.

*1 For FFTRID, all real types should be changed to double precision real types.

(3) Performance

Because this routine is written in the assembly language, and an effort is made to reduce the number of calculations of trigonometric functions, its speed and precision are high.

(4) Calculation method

The calculation is executed by reversing the algorithm of real fast Fourier analysis (FFTR, FFTRD). Refer to the bibliography [1] of FFTR.

(1987. 08. 10)

FFTS/D (Complex Fast Fourier Transform)

Complex Fast Fourier Transform

Programmed by	Ichizo Ninomiya, April 1977
Format	Subroutine language: FORTRAN; size: 124 and 129 lines respectively

(1) Outline

If sample values X_j, $j=0,1,\cdots,N-1$ (X_0 is the value at the origin) at the N equipartition points of a period of a periodic function is given, each of the periodic components C_j, $j=0,1,\cdots,N-1$ is given as the Fourier transform

$$C_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X_k W^{jk}, \quad j=0,1,\cdots,N-1$$

Where, $W=exp(-2\pi i/N)$. Conversely, if the periodic components C_j are given, X_j is given by the inverse transform

$$X_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} C_k W^{-jk}, \quad j=0,1,\cdots,N-1$$

This routine is used to perform the above calculation by fast Fourier transform when N is in the form of 2^M.

(2) Directions

CALL FFTS/D(A, B, N, INV, ILL)

Argument	Type and kind (*1)	Attribute	Content
A	Complex type One-dimensional array	Input/output	For forward transformation, X_k are input to output C_j. For inverse transformation, C_k are input to output X_j. $C_{j-1}(X_{j-1})$ is entered in A(j).
B	Complex type One-dimensional array	Work area	Work area used in the subroutine.
N	Integer type	Input	Represents the size of arrays A and B. It should be of the form of 2^M. $N \geq 2$
INV	Integer type	Input	INV=0 means forward transformation, and INV=1 means inverse transformation.
ILL	Integer type	Output	ILL=0: Normal termination. ILL=30000: When N is not in the form of $2^M (M>0)$.

*1 For FFTD, all complex types should be changed to double precision complex types.

(3) Performance

Because this routine uses the techniques of Fourier transform and has the following characteristics, its speed and precision are high.

1. The value of sine and cosine is calculated only when the absolute value of arguments is within $\pi/8$. Once the value is obtained, it is used eight times with a small correction added.

2. The low-order approximation polynomials prepared in the routine are used instead of calling the elementary external sine and cosine functions.

(4) Note

1. Usually, Fourier transformation is defined as

$$C_j = \frac{1}{N} \sum_{k=0}^{N-1} X_k \, W^{jk}, \quad j=0,1,\cdots,N-1$$

Inverse transformation is also defined as

$$X_j = \sum_{k=0}^{N-1} C_k \, W^{-jk}, \quad j=0,1,\cdots,N-1$$

However, it should be noted that this routine uses different definitions.

2. The special-purpose routines FFTR and FFTRI should be used for real number input data.

3. FFTC/B is available as the routine with the same function as this routine. Select and use them properly.

(1987. 05. 08) (1987. 08. 10)

FT235C/B and FT235R/D (Complex and Real Fast Fourier Transform for the Case of Sample Number of the Form of $2^K3^L5^M$)

Complex and Real Fast Fourier Transform for the Case of Sample Number of the Form of $2^K3^L5^M$

Programmed by	Ichizo Ninomiya, April 1977
Format	Subroutine language: FORTRAN; size: 178 and 42 lines respectively

(1) Outline

FT235C/B and FT235R/D are the subroutines for making complex fast Fourier analysis (FT235C/B) and real fast Fourier transform (FT235R/D) when the number of divisions of a period is of the form of $N=2^K3^L5^M$.

Because various definitions in FT235C/B are the same as in FFTC/B, and those in FT235R/D are the same as in FFTR/D, refer to each explanation.

(2) Directions

CALL FT235C/B(A, B, N, INV, ILL)

CALL FT235R/D(A, B, N, ILL)

Argument	Type and kind (*1)		Attribute	Content
	FT235C	FT235R		
A	Complex type (*1) One-dimensional array		Input/output	One-dimensional array containing N elements. In forward transformation, X_k are input to output C_j. In inverse transformation, C_k are input to output X_j. $C_{j-1}(X_{j-1})$ is entered in $A(j)$.

Argument	Type and kind (*1)		Attribute	Content
		Real type (*1) One-dimensional array	Input/output	One-dimensional array containing N elements. If the values at N equipartition points of a period of the periodic function is sequentially entered, the cosine and sine components are output in this order. Each components are output in natural order. Precisely, the K-th order cosine components are output to A(K+1), and the J-th order sine components are output to A(N/2+J+1).
B	Complex type (*1) One-dimensional array	Real type (*1) One-dimensional array	Work area	Work area. It must be of the same type and size as the argument A.
N	Integer type	Integer type	Input	N must be the number of divisions in a period, and be in the form of $N=2^K3^L5^M$. N>2. K≧1 should hold for FT235R/D.
INV	Integer type.		Input	If INV=0, forward transformation is executed. If INV=1, inverse transformation is executed.
ILL	Integer type	Integer type	Output	ILL=30000: When limits on the input are exceeded. Otherwise, 0 is output.

*1 For FT235B, all complex types should be changed to double precision complex types.

For FT235D, all real types should be changed to double precision real types.

(3) Performance

Because this routine is not the $N=2^M$ type, its speed is slow as compared with other routines. Therefore, it is reasonable for the 2^M type to use the special-purpose routine for that type.

(4) Example of use

If "CALL FT235R/D(A, B, N, ILL)" is executed, "CALL FT235C/B(A, B, N/2, 0, ILL)" is executed in FT235R.

Therefore, N should be an even number $(N=2^K3^L5^M, K\geq 1)$. Because A and B are handled as a complex type one-dimensional array (array of size N/2: $(A(1)+iA(2), A(3)+iA(4)\cdots)$) in this call, they should be prepared for such handling. One example is to use the EQUIVALENCE statement described in the example below. (It is necessary and sufficient that the top elements of A and B are allocated to the even number address.)

```
C       MAIN PROGRAM
        DIMENSION A(720),B(720)
        COMPLEX CA(360),CB(360)
        EQUIVALENCE(A,CA),(B,CB)
        READ(5,500)(A(I),I=1,720)
  500   FORMAT(6F12.0)
        CALL FT235R(A,B,720,ILL)
           :
        STOP
        END
```

(5) Note

When FT235R/D is to be used, the number N of divisions must be in the form of $N=2^K3^L5^M$ and be an even number. Because the arrays A and B are real type one-dimensional arrays of size N, and handled as a complex type one-dimensional array of size N/2, they should be prepared for such handling. See the example.

(1987. 05. 08) (1987. 08. 10)

TRIGQP/TRIGQD (Table of Trigonometric Function Arranged in Bit Reverse Order)

Table of Trigonometric Function Arranged in Bit Reverse Order

Programmed by	Tatsuo Torii, December 1978
Format	Subroutine Language: FORTRAN; Size: 51 and 52 lines respectively

(1) Outline

TRIGQP/TRIGQD generates a trigonometric function table that is required for fast sine and cosine transforms and the Chebyshev series expansion of functions.

It defines the n-bit decimal fraction $j^*=j_1 2^{-1}+j_2 2^{-2}+\cdots+j_n 2^{-n}$ less than 1 for the n-bit integer $j=j_1 2^0+j_2 2^1+\cdots+j_n 2^{n-1}$, $j_i \in \{0,1\}$, and calculates the complex trigonometric function $e^{\pi/4 ij^*}$, $j=0,1,2,\cdots$.

(2) Directions

CALL TRIGQP(C, M, ICON)

CALL TRIGQD(C, M, ICON)

Argument	Type and kind (*1)	Attribute	Content
C	Real type One-dimensional array	Output	$C(1)=\cos(\pi/4)$ $C(2j)=\cos(\pi/4)j^*$ $1 \leq j < 2^M-1$ $C(2j+1)=\sin(\pi/4)j^*$ $1 \leq j < 2^M-1$
M	Integer type	Input	Size of array C $\geq 2^M-1$ $M \geq 1$
ICON	Integer type	Output	ICON=0: Normal. ICON=30000: Parameter error.

(3) Performance

If the number of data items in the trigonometric function table is 2^M-1, the required

arithmetic operations are M square roots, and 2^M multiplications.

(4) Calculation method

Putting $W_j=e^{\pi/4ij}$* for simplicity, they obey the following recurrence formulas.

$W_0=e^{\pi/4i}, W_1=e^{\pi/8i}$ Initial value

$W_{2^l}=(W_{2^{l-1}})^{1/2}$, The imaginary part of square roots is positivie.

$W_{2^l+j}=W_{2^{l-1}+j}\cdot\overline{W}_{2^l}$, $1\leq j<2^{l-1}$

$W_{2^l+2^{l-1}+j}=W_{2^{l-1}+j}\cdot W_{2^l}$, $0\leq j<2^{l-1}$

$l=1,2,\cdots$

(1987. 05. 12) (1987. 08. 10)

160

VCHB1S/D, DCHB1S/D, ICHB1S/D, VCHB3S/D, DCHB3S/D, ICHB3S/D

(Evaluation of Chebyshev Series) (VCHB1S/D)

(Differential Coefficient of Chebyshev series) (DCHB1S/D)

(Evaluation of Indefinite Integral) (ICHB1S/D)

(Evaluation of Shifted Chebyshev Series) (VCHB3S/D)

(Differential Coefficient) (DCHB3S/D)

(Evaluation of Indefinite Integral) (ICHB3S/D)

Evaluation of Chebyshev Series (VCHB1S/D)

Differential Coefficient (DCHB1S/D)

Evaluation of Indefinite Integral (ICHB1S/D)

Evaluation of Shifted Chebyshev Series (VCHB3S/D)

Differential Coefficient (DCHB3S/D)

Evaluation of Indefinite Integral (ICHB3S/D)

Programmed by	Tatsuo Torii, December 1978
Format	Subroutine Language: FORTRAN Size: 75, 76, 75, 76, 75, 76, 80, 81, 80, 81, 80, and 81 lines respectively

(1) Outline

The subroutines perform the following calculations for the series $\sum_{0\le k<N}{}' a_k T_k(x)$ of the Chebyshev polynomials of first kind.

1. Obtains the value of series (1) at arbitrary points $x \in [-1, 1]$.
2. Calculates differential coefficients at the point x.
3. Obtains the integral $\int_{-1}^{x}(\sum_{0\le k<N}{}' a_k T(t))dt$ with upper limit $x \in [-1, 1]$.

VCHB3S, DCHB3S, and ICHB3S obtain the value of the series, differential coefficient, and integral $\int_0^x$ with respect to the series $\sum_{0\le k<N}{}' a_k T_k^*(x)$ of shifted Chebyshev polynomials.

(2) Directions

CALL VCHB1S/D(A, N, X, F, ICON)

CALL DCHB1S/D(A, N, X, F, ICON)

CALL ICHB1S/D(A, N, X, F, ICON)

CALL VCHB3S/D(A, N, X, F, ICON)

CALL DCHB3S/D(A, N, X, F, ICON)

CALL ICHB3S/D(A, N, X, F, ICON)

Argument	Type and kind (*1)	Attribute	Content
A	Real type One-dimensional array	Input	Size of array A≧N. Fourier coefficients $a_0, a_1, \cdots, a_{N-1}$ are stored in A(1), A(2), ..., and A(N). N≧1
N	Integer type		
X	Real type	Input	-1≦X≦1.
F	Real type	Output	Calculation value of each subroutine.
ICON	Integer type	Output	ICON=0: Normal. ICON=30000: Parameter error.

*1 For double precision subroutines, all real types should be double precision real types.

(3) Calculation method

The value at the point $x \in [-1, 1]$ of the Chebyshev series

$$S_N(x) = \sum_{0 \leq k < N} a_k T_k(x)$$

can be obtained with the recurrence formula, named Clenshaw's algorithm

$$b_N = 0$$

$$b_{N-1} = a_{N-1}$$

$$b_k = 2x b_{k+1} - b_{k+2} + a_k$$

$$k = N-2, N-1, \cdots, 1, 0$$

$$S_N(x) = \frac{1}{2}(b_0 - b_2)$$

or

$$= x b_1 - b_2 + \frac{a_0}{2}$$

The sum of (N-1)-th order Chebyshev series is obtained by N times of multiplication.

Arrays are not used for the sequence $\{b_k\}$ that is the intermediate result. A differential coefficient

$$\sum_{k=1}^{N} a_k \frac{d}{dx} T_k(x) \Big|_{x=x}$$

at the point x of the N-th order Chebyshev series is obtained with the recurrence formula

$$b_{N+1}=0$$

$$b_N=Na_N$$

$$b_k=2xb_k-b_{k+1}+ka_k$$

$$k=N-1, N-2, \cdots, 1$$

Differential coefficient $=b_1$.

The indefinite integral of Chebyshev series is

$$\int_{-1}^{x} \sum_{0\le k<N} a_k T_k(x)dx = \sum_{0\le k<N} a_k \int_{-1}^{x} T_k(x)dx = \sum_{K=1}^{N} \frac{a_{k-1}-a_{k+1}}{2k}(T_k(x)-(-1)^k)$$

Where $a_{N+1}=a_N=0$.

Thus, the integral value can be obtained by

$$b_{N+1}=0, c_N=a_{N-1}/2N$$

$$b_N=c_N, S_N=c_N$$

$$c_k=(a_{k-1}-a_{k+1})/2k$$

$$b_k=2xb_{k+1}-b_{k+2}+c_k, S_{k-1}=c_k-S_k$$

$$k=N, N-1, \cdots, 1$$

Integral value $=xb_1-b_2+S_1$.

The value can be obtained with a similar method for shifted Chebyshev polynomials.

(4) Example

For simplicity, the numerical differentiation and integration are tested by an exponential function. The value of

$$f(x), f'(x), \int_{-1}^{x} f(x)dx$$

$$x=\frac{i}{4}, i=-4, -3, \cdots, 3, 4$$

is obtained by expanding the funciton on the interval [-1, 1]

$$f(x)=e^x$$

into Chebyshev series under a required precision using VCHB1S, DCHB1S and ICHB1S. Also, the example includes the calculation of

$$f(x), f'(x), \int_0^x f(x)dx, x=0,1,\cdots,8$$

where the same exponential function

$$f(x)$$

is expanded over an interval [0, 8] with shifted Chebyshev series. This requires variable transformation for changing an interval [0, 8] to [0, 1].

```
C       TEST FOR SUBROUTINE VCHB1S,DCHB1S AND ICHB1S
        DIMENSION A(257)
        EXTERNAL F
        EPSA=1.0E-05
        EPSR=0.0
        NMIN=0
        NMAX=257
        CALL FCHB1S(F,EPSA,EPSR,NMIN,NMAX,A,N,ERR,ILL1)
        A(N)=A(N)*0.5
        H=0.25
        X=-1.0
   10   CONTINUE
        CALL VCHB1S(A,N,X,V,ILL2)
        CALL DCHB1S(A,N,X,D,ILL3)
        CALL ICHB1S(A,N,X,VI,ICON)
        ICON=ICON+ILL1+ILL2+ILL3
        TRUEV=EXP(X)
        ERV=TRUEV-V
        ERD=TRUEV-D
        ERI=TRUEV-EXP(-1.0)-VI
        WRITE(6,600) X,V,ERV,D,ERD,VI,ERI,N,ICON
  600   FORMAT(1HO,4X,F8.3,3(F15.06,E15.03),2I8)
        X=X+H
        IF(X.LE.1.0) GO TO 10
        STOP
        END

        FUNCTION F(P)
        F=EXP(P)
        RETURN
        END

C       TEST FOR SUBROUTINE VCHB3S,DCHB3S AND ICHB3S
        DIMENSION A(257)
        EXTERNAL F
        EPSA=1.0E-05
        EPSR=0.0
        NMIN=0
        NMAX=257
```

```
      CALL FCHB3S(F,EPSA,EPSR,NMIN,NMAX,A,N,ERR,ILL1)
      A(N)=A(N)*0.5
      Y=0.0
      H=1.0
   10 CONTINUE
C     APPLY THE VARIABLE TRANSFORMATION
      X=Y/8.0
      CALL VCHB3S(A,N,X,V,ILL2)
      CALL DCHB3S(A,N,X,D,ILL3)
      CALL ICHB3S(A,N,X,VI,ICON)
      ICON=ICON+ILL1+ILL2+ILL3
      D=D/8.0
      VI=VI*8.0
      TRUEV=EXP(Y)
      ERV=TRUEV-V
      ERD=TRUEV-D
      ERI=TRUEV-1.0-VI
      WRITE(6,600) Y,V,ERV,D,ERD,VI,ERI,N,ICON
  600 FORMAT(1HO,4X,F8.3,3(F15.06,E15.03),2I8)
      Y=Y+H
      IF(Y.LE.8.0) GO TO 10
      STOP
      END

      FUNCTION F(P)
C     APPLY THE VARIABLE TRANSFORMATION
      Q=8.0*P
      F=EXP(Q)
      RETURN
      END
```

Expansion of e^x by $\{ T_k(x) \}$, sum of series, differential coefficient, and indefinite integral

x	Sum of series	Error	Differential coefficient	Error	Integral	Error
-1.00	0.367879	-0.745E-08	0.367879	0.291E-06	0.000000	0.373E-08
-0.75	0.472367	-0.745E-08	0.472366	0.112E-06	0.104487	0.745E-08
-0.50	0.606531	0.149E-07	0.606531	-0.104E-06	0.238651	0.745E-08
-0.25	0.778801	-0.149E-07	0.778801	-0.596E-07	0.410921	0.0
0.00	1.000000	0.0	1.000000	0.134E-06	0.632121	-0.745E-08
0.25	1.284025	0.0	1.284026	-0.119E-06	0.916146	0.745E-08
0.50	1.648721	-0.298E-07	1.648721	-0.596E-07	1.280842	-0.224E-07
0.75	2.117000	0.0	2.117000	0.596E-07	1.749121	0.745E-08
1.00	2.718282	-0.596E-07	2.718281	0.119E-05	2.350402	-0.522E-07

Note: Precision required for development: $\varepsilon=10^{-5}$; number of samples: N=9.

Expansion of e^x by $\{ T_k^*(x/8) \}$, sum of series, differential coefficient and indefinite integral.

x	Sum of series	Error	Differential coefficient	Error	Integral	Error
0.0	0.999996	0.381E-05	0.999893	0.107E-03	0.000008	-0.763E-05
1.0	2.718304	-0.219E-04	2.718166	0.115E-03	1.718302	-0.200E-04
2.0	7.389103	-0.468E-04	7.389107	-0.507E-04	6.389076	-0.201E-04
3.0	20.085506	0.305E-04	20.085672	-0.135E-03	19.085560	-0.229E-04
4.0	54.598145	0.572E-05	54.597870	0.280E-03	53.598206	-0.553E-04
5.0	148.413208	-0.496E-04	148.413406	-0.248E-03	147.413169	-0.114E-04
6.0	403.428611	0.183E-03	403.428878	-0.839E-04	402.428771	0.229E-04
7.0	1096.63305	0.916E-04	1096.63201	0.113E-02	1095.63314	0.0
8.0	2980.958	0.0	2980.9678	-0.983E-02	2979.95788	0.122E-03

Note: Precision required for expansion: $\varepsilon=10^{-5}$; number of samples: N=17.

(1987.06.05) (1987.08.10)

166

VCHB2S/D, ICHB2S/D

(Evaluation of Second Kind Chebyshev Series) (VCHB2S/D)

(Evaluation of Indefinite Integral) (ICHB2S/D)

Evaluation of Second Kind Chebyshev Series(VCHB2S/D)

Evaluation of Indefinite Integral(ICHB2S/D)

Programmed by	Tatsuo Torii, December 1978
Format	Subroutine Language: FORTRAN; Size: 47, 48, 47, and 48 lines respectively

(1) Outline

VCHB2S/D and ICHB2S/D obtain the value at the point x of the second kind Chebyshev series

$$\sum_{0\leq k<N} a_k U_k(x) \qquad (1)$$

and integral

$$\int_{-1}^{x} \sum_{k} a_k U_k(x)\,dx \qquad (2)$$

(2) Directions

CALL VCHB2S/D(A, N, X, F, ICON)

CALL ICHB2S/D(A, N, X, F, ICON)

Argument	Type and kind (*1)	Attribute	Content
A	Real type One-dimensional array	Input	Size of array A≧N. Fourier coefficients $a_0, a_1, \cdots, a_{N-1}$ are stored in A(1), A(2), ..., and A(N). N≧1
N	Integer type		
X	Real type	Input	-1≦X≦1.
F	Real type	Output	Calculated value of each subroutine.
ICON	Integer type	Output	ICON=0: Normal. ICON=30000: Parameter error.

*1 For double precision subroutines, all real types should be double precision real types.

(3) Calculation method

Because the recurrence formula of the Chebyshev polynomials of second kind is the same as that of first kind except for the initial conditions, the value of series (1) is obtained with

$b_N = 0$, , $b_{N-1} = a_{N-1}$, , $b_k = 2_\alpha b_{k+1} - b_{k+2} + a_k$, , $k = N-2, N-3, \cdots, 1, 0$, and value of series (1) = b_1.

Also, indefinite integral (2) is obtained by

$$\sum_{k=1}^{N} \frac{a_{k-1}}{k} T_k(x) - \sum_{k=1}^{N} (-1)^k \frac{a_{k-1}}{k}$$

. Therefore, the calculation conforms to the indefinite integral (ICHB1S) of the first kind Chebyshev series.

(4) Example

By expanding the exponential function e^x over an interval of [-1, 1] using the second kind Chebyshev series (and FCHB2S), the value and integral of this series are found. The point x is a sample point on the interval [-1, 1] divided into eight equally parts.

```
C       TEST FOR SUBROUTINE VCHB2S AND ICHB2S
        DIMENSION A(255)
        EXTERNAL F
        EPSA=1.0E-05
        EPSR=0.0
```

```
      NMIN=0
      NMAX=255
      CALL FCHB2S(F,EPSA,EPSR,NMIN,NMAX,A,N,ERR,ILL1)
      H=0.25
      X=-1.0
   10 CONTINUE
      CALL VCHB2S(A,N,X,VA,ILL2)
      CALL ICHB2S(A,N,X,VI,ICON)
      ICON=ICON+ILL1+ILL2
      TRUEV=EXP(X)
      ERV=TRUEV-VA
      ERI=TRUEV-EXP(-1.0)-VI
      WRITE(6,600) X,VA,ERV,VI,ERI,N,ICON
  600 FORMAT(1HO,4X,F8.3,2(F15.06,E15.03),2I8)
      X=X+H
      IF(X.LE.1.0) GO TO 10
      STOP
      END

      FUNCTION F(P)
      F=EXP(P)
      RETURN
      END
```

Expansion of e^x by $\{U_k(x)\}$, sum of series, and indefinite integral

x	Sum of series	Error	Integral	Error
-1.00	0.367879	0.0	0.000000	-0.373E-08
-0.75	0.472367	0.745E-08	0.104487	0.745E-08
-0.50	0.606531	-0.0	0.238651	0.745E-08
-0.25	0.778801	-0.298E-07	0.410921	-0.745E-08
0.00	1.000000	0.0	0.632121	-0.745E-08
0.25	1.284025	0.0	0.916146	-0.745E-08
0.50	1.648721	-0.298E-07	1.280842	-0.224E-08
0.75	2.117000	0.0	1.749121	0.745E-08
1.00	2.718282	-0.596E-07	2.350402	-0.522E-07

Note: Required precision $\varepsilon=10^{-5}$ for expansion

Number of samples N=15

(1987.05.25) (1987.8.10)

VCOSS/D,VSINS/D

(Evaluation of cosine series) (VCOSS/D) (Evaluation of sine series) (VSINS/D)

Evaluation of Cosine Series(VCOSS/D)

Evaluation of Sine Series(VSINS/D)

Programmed by	Tatsuo Torii, December 1978
Format	Subroutine Language: FORTRAN; Size: 38, 39, 38, and 39 lines respectively

*1 For double precision subroutines, all real types should be double precision real types.

(1) Outline

VCOSS/D and VSINS/D obtains the values of the cosine series $\sum'_{0<=k<N} a_k \cos k\theta$ and sine series $\sum_{k=1}^{N} a_k \sin k\theta$.

(2) Directions

CALL VCOSS/D(A, N, T, F, ICON)

CALL VSINS/D(A, N, T, F, ICON)

Argument	Type and kind (*1)	Attribute	Content
A	Real type One-dimensional array	Input	Size of array A≧N. For VCOSS/D, $a_0, a_1, \cdots, a_{N-1}$ are stored on A. For VSINS/D, $a_1, \cdots, a_N$ are stored on A.
N	Integer type		N≧1
T	Real type	Input	Arbitrary real number. Retained.
F	Real type	Output	Evaluation of cosine (VCOSS) and sine (VSINS) series at θ=t.
ICON	Integer type	Output	ICON=0: Normal. ICON=30000: Parameter error.

*1 For double precision subroutines, all real types should be double precision real types.

(3) Calculation method

The sum of the cosine series is obtained by Clenshaw's method as well as the sum of the Chebyshev series of first kind.

The sum of the sine series can be obtained by multiplying the sum of the Chebyshev series of second kind by $\sin\theta$.

(4) Example

1. Example of cosine series calculation

If a periodic function can be expanded in Fourier series, then it is easily integrated term by term. Now, the integration of the sine series is obtained below as an example of using the subroutine VCOSS.

The termwise integration of a generating function of the sine function is written by

$$\int_0^{\varphi} \frac{\sin\theta}{1-2t\cos\theta+t^2}d\theta = \sum_{k=0}^{\infty}{}' a_k \cos k\varphi$$

where

$$a_k = -t^{k-1}/k, \quad k \geqq 1$$

, $a_0 = -2\sum_{k=1}^{\infty} a_k$. Thus, the integrand is expanded and integrated termwise by using the subroutine FSINOS.

The value of the cosine series is obtained by using the subroutine VCOSS varying the upper limit φ of the integration with $1/12\pi$, $2/12\pi$, ..., and $6/12\pi$. The analytic solution of this integration is expressed as

$$\frac{1}{2t}\log\left\{\frac{1-2t\cos\varphi+t^2}{(1-t)^2}\right\}$$

```
C       TEST FOR SUBROUTINE VCOSS
        DIMENSION A(256)
        EXTERNAL F
        COMMON T
        TRUE(P,T)=ALOG((1.0-2.0*T*COS(P)+T*T)/(1.0-T)**2)*0.5/T
        T=0.5
        NX=6
        HPI=2.0*ATAN(1.0)
        H=HPI/FLOAT(NX)
        EPSA=1.0E-05
        EPSR=0.0
```

```
      NMIN=0
      NMAX=255
      CALL FSINOS(F,EPSA,EPSR,NMIN,NMAX,A,N,ERR,ILL)
      NP1=N+1
      S=0.0
      DO 10 I=1,N
      K=NP1-I
      A(K+1)=-A(K)/FLOAT(K)
      S=A(K+1)+S
   10 CONTINUE
      A(1)=-S-S
      THETA=H
      DO 20 I=1,NX
      CALL VCOSS(A,NP1,THETA,VA,ICON)
      ICON=ICON+ILL
      ERV=TRUE(THETA,T)-VA
      WRITE(6,600) I,VA,ERV,T,N,ICON
  600 FORMAT(1HO,4X,I4,F15.06,E15.03,F8.3,2I8)
      THETA=THETA+H
   20 CONTINUE
      STOP
      END

      FUNCTION F(P)
      COMMON T
      F=SIN(P)/(1.0-2.0*T*COS(P)+T*T)
      RETURN
      END
```

Development and calculus of sine generating functions

φ	Calculus	Error
$\pi/12$	0.127774	-0.119E-07
$2\pi/12$	0.429115	-0.217E-07
$3\pi/12$	0.775452	-0.300E-07
$4\pi/12$	1.098612	-0.274E-07
$5\pi/12$	1.377436	-0.194E-07
$6\pi/12$	1.609438	-0.190E-07

Note: Required precision 10^{-5} (input) for sine series expansion.

Parameter t=1/2 (input)

Number of samples N=31 (output)

2. Example of sine series calculation

Elliptic Integral

The calculation example of

$$F(\varphi,\alpha)=\int_0^{\varphi}\frac{d\theta}{\sqrt{1-\sin^2\alpha\sin^2\theta}}$$

is given below. For simplicity, suppose $\alpha=\pi/4$. If the integrand developed into cosine series is integrated termwise, it becomes a sine series except the constant terms. Thus, the sum is obtained for various φ. The constant terms can be separately calculated and added. In the following example, the variable φ is assigned as $\varphi=1/12\pi$, $2/12\pi$, ..., and $6/12\pi$.

```
C       TEST FOR SUBROUTINE VSINS.
        DIMENSION A(257)
        EXTERNAL F
        NX=6
        HPI=2.0*ATAN(1.0)
        H=HPI/FLOAT(NX)
        EPSA=1.0E-05
        EPSR=0.0
        NMIN=0
        NMAX=257
        CALL FCOSCS(F,EPSA,EPSR,NMIN,NMAX,A,N,ERR,ILL)
        A(N)=A(N)*0.5
        M=N-1
        CONST=A(1)*0.5
        DO 10 I=1,M
        A(I)=A(I+1)/FLOAT(I)
   10   CONTINUE
        T=H
        DO 20 I=1,NX
        CALL VSINS(A,M,T,V,ICON)
        V=V+CONST*T
```

```
      ICON=ICON+ILL
      WRITE(6,600) I,V,N,ICON
  600 FORMAT(1HO,4X,I4,F15.06,2I8)
      T=T+H
   20 CONTINUE
      STOP
      END

      FUNCTION F(P)
      F=1.0/SQRT(1.0-0.5*SIN(P)**2)
      RETURN
      END
```

Elliptic calculus

φ	Calculus	Error
$\pi/12$	0.263297	Number of samples = 17
$2\pi/12$	0.535623	
$3\pi/12$	0.826018	True value
$4\pi/12$	1.142429	1.14242906
$5\pi/12$	1.487885	
$6\pi/12$	1.854075	1.85407468

(1987.05.28) (1987.08.11)

8. Numerical quadrature

[Method of choosing numerical integration routines]

To meet various cases, NUMPAC includes a large number of excellent quadrature routines such as one-dimensional and multidimensional integrations, finite and infinite interval integrations, and fixed rule and automatic integrations. If they are carefully selected based on the following guides, significant effects can be achieved in both precision and speed. For simplicity, the name of recommended routines is represented with the one for single precision.

(A) One-dimensional definite interval

1. Well-behaved analytic function
   (1) Fixed rule quadrature — GASNS
   (2) Automatic quadrature — QDAPBS, DEFINS, and AQNN9S
2. Analytic function of oscillatory type — QDAPBS
3. Function of peak type — AQNN9S
4. Analytic function with singularity at end points — DEFINS
5. Function with singularity and discontinuity — AQNN9S
6. Function of uncertain behavior — AQNN9S
7. Integral over a whole period of periodic function — TRAPZS

(B) When $f(x)$ is a well behaved function in the integral $\int_0^\infty e^{-x} f(x)dx$ in a one-dimensional semi-infinite interval

1. Fixed rule quadrature — GSLNS
2. Automatic quadrature — HINFAS and HINFES

(C) One-dimensional infinite interval

1. When $f(x)$ is a well-behaved function in the form of $\int_{-\infty}^\infty e^{-x^2} f(x)dx$
   (1) Fixed rule quadrature — GSHNS
   (2) Automatic quadrature — INFINS
2. When $f(x)$ decreased rapidly in the form of $\int_{-\infty}^\infty f(x)dx$ — TRAPZS

(D) Multidimensional, fixed rule quadrature

1. Function input — MQPRRS
2. Data input — MQNCDS
3. Higher dimension — MQFSRS

(E) Multidimensional automatic quadrature AQMDS and AQNDS

To help raise the precision of results, pre-processing should be executed. For example, divide integration intervals if necessary, or turn the upper and lower limits to numbers represented without error such as 0 or 1 by variable transformation.

AQCHYS/D (Automatic quadrature of Cauchy principal value integrals)

Automatic Quadrature of Cauchy Principal Value Integrals

Programmed by	Takemitsu Hasegawa; February 1984
Format	Subroutine language; FORTRAN Size; 319 and 322 lines respectively

(1) Outline

When integrand function f(x), lower limit a, upper limit b, and pole c are given, AQCHYS or AQCHYD automatically calculates the approximate value

$$I_N(c)$$

of the Cauchy principal value integral

$$I(c)=p\int_a^b \frac{f(x)}{x-c}dx \quad , \quad a<c<b$$

It calculates the solution with precision that satisfies

$$|I(c)-I_N(c)| \leq max(\varepsilon_a, \varepsilon_r |I(c)|)$$

where ε_a is the requested absolute precision and ε_r is the requested relative precision.

AQCHYS is a routine for single precision and AQCHYD is one for double precision.

(2) Directions

CALL AQCHYS/D(A, B, C, FUN, EPSA, EPSR, NMIN, NMAX, JUMP, S, N, ERR, ICON)

Argument	Type and kind (*1)	Attribute	Content
A	Real type	Input	Lower limit of integral domain.
B	Real type	Input	Upper limit in integral domain. A<B
C	Real type	Input	Pole C of principal value integral.

Argument	Type and kind (*1)	Attribute	Content
FUN	Real type function subprogram	Input	Given function f(x). The user should prepare a function subprogram f(x) having a variable x.
EPSA EPSR	Real type	Input	Requested absolute error εa (EPSA) and relative error εr (EPSR) for approximate value S of an integral. EPSA≧0, EPSR≧0.
NMIN NMAX	Integer type	Input	Lower limit (NMIN) and upper limit (NMAX) of the number of function FUN evaluations. NMIN is usually set to 9. NMAX is usually set to 200 to 900. When NMAX≧514, NMAX is assumed to be 514 (single precision). When NMAX≧2050, NMAX is assumed to be 2050 (double precision). 0<NMIN<NMAX.
JUMP	Integer type	Input	JUMP is usually set 0. If you want to calculate for the same function f(x) with the same value for εa but with different values for pole C, set JUMP to 1 when calling this routine second time and after. Then, the values of FUN camputed and stored in the first call are reused.
S	Real type	Output	Approximate value of integral.
N	Integer type	Output	Total number of function FUN evaluations.
ERR	Real type	Output	Estimation of absolute error of S.
ICON	Integer type	Output	ICON=0:Normal termination. ICON=10000: The accuracy of the approximate value of the integral has reached the level of rounding error. ICON=20000: Convergence does not occur even after the function has been evaluated NMAX times. ICON=30000: Parameter error.

*1 For double precision subroutines, all real types should be changed to double precision real types.

(3) Calculation method

To make explanation simple, the integration interval is assumed to be [-1, 1]. An integral is transformed as follows:

$$P\int_{-1}^{1}\frac{f(x)}{x-c}dx=\int_{-1}^{1}\frac{f(x)-f(c)}{x-c}dx+f(c)\,ln(\frac{1-c}{1+c})$$

In the first integrand at the right-hand side, c is no longer a pole. F(x) is expanded in Chebyshev polynomial. The order of expansion is increased more gradually than doubly until the requested accuracy is satisfied. This is to save the number of function evaluations. The expansion coefficients are calculated efficiently by using FFT.

(4) Example

The integral

$$\int_{-1}^{1}\frac{1}{x-c}\,\frac{1}{x^2+\alpha^2}dx$$

that has ten poles C=0.1i-0.01 (i = 1, 2, ..., 10), when the values of parameter α are 1, 1/2, and 1/4, is calculated.

$\varepsilon a=10^{-4}$ and $\varepsilon r=0$.

As shown in the above example, when the integrand function contains a parameter (α in this example), the parameter is put in the common area to communicate with the main program.

(5) Notes

1. This method should be used only when pole C is in the integration interval and |c-a| and $|c-b|>10^{-7}$ (AQCHYS) or 10^{-16} (AQCHYD) is satisfied.

2. When this routine is called repeatedly for the same function f(x) with the same value for εa but with different values for pole C, set JUMP to 1 when calling this routine second time and after. Then, the value of FUN used for the first time is reused repeatedly. This enables efficient calculation and greatly saves calculation time. (For this operation, εr should be set to 0.0.)

Bibliography

1) Tatsuo Torii and Takemitsu Hasegawa: "FFT of real function gradually increasing sample points", Information Processing Soc. of Japan, Vol. 24, No. 3, pp. 343-350 (1983).

2) Takemitsu Hasegawa and Tatsuo Torii: "Automatic quadrature of Cauchy principal value integrals", Kagakukenkyuhi Sogokenkyu (A) Reports of Applied Mathematic Symposium (Representative: Nakashima and Yoneda)", pp. 163-174 (1983).

(1987.08.11)

AQCOSS/D and AQSINS/D (Automatic Quadrature of Semi-Infinite Integral of Oscillatory Function)

Automatic Quadrature of Semi-infinite Integral of Oscillatory Function

Programmed by	Toshio Yoshida, September 1982
Format	Subroutine Language: FORTRAN; Size: 164, 168, 164, and 168 lines respectively

(1) Outline

AQCOSS/D and AQSINS/D calculate the semi-infinite integral $\int_a^\infty f(x)\cos qx\,dx$ (same as for $\int_a^\infty f(x)\sin qx\,dx$) within the prescribed absolute precision ε for the function f(x) that attenuates with the increase of x.

(2) Directions

CALL AQCOSS/D(A, Q, F, S, EPS, LF, LA, NF, NS, W, ILL)

CALL AQSINS/D(A, Q, F, S, EPS, LF, LA, NF, NS, W, ILL)

Argument	Type and kind (*1)	Attribute	Content
A	Real type	Input	Lower limit a of definite integral.
Q	Real type	Input	q of integrand function. q>0.
F	Real number type function subprogram	Input	Name of f(x) of integrand function. The function as an actual argument for this name should be prepared as a function subprogram with only one integral variable. The name of f(x) should be defined as the EXTERNAL declaration in the program that calls this subroutine.
S	Real type	Output	The values of an definite integral is output.

Argument	Type and kind (*1)	Attribute	Content
EPS	Real type	Input	Positive number that represents a prescribed absolute accuracy. Single precision: $10^{-3}\sim10^{-5}$ Double precision: $10^{-5}\sim10^{-14}$ These are standard values. (See 3 in "Note")
LF	Integer type	Input	Upper limit of total number of calculations of function f(x). LF>12. The adequate value is several thousands.
LA	Integer type	Input	Upper limit of the number of operations that f(x) requires to obtain a function g(x). LA>10. The adequate value is several hundreds.
NF	Integer type	Output	Total number of calculations of a function f(x). If NF>LF, control escapes from the routine, stopping the calculation.
NS	Integer type	Output	Number of sampling times of an integrand function in $\int_a^{a+\pi/q} g(x)\cos qx$.
W	One-dimensional array of real number type.	Work area	Size LA.
			This argument represents a calculation state in the routine. It is set to 0 in the routine. Each time the next state is activated, a certain value is added.

Argument	Type and kind (*1)	Attribute	Content
ILL	Integer type	Output	(1) Integration of $\int_a^{a+\pi/q} g(x)\cos qx\, dx$ (a) If the length of a small subinterval becomes extremely small, 1 is assumed. (b) If a discontinuity is detected, 10 is assumed. (c) If a logarithmic singular point is detected, 100 is assumed. (d) If an algebraic singular point is detected, 1000 is assumed. (e) If the order of an algebraic point is up to -1, 20000 is assumed. (2) When the value of a function g(x) is to be obtained by Euler transformation, if the number of calculations of f(x) becomes greater than LA, 15000 is assumed. (3) If NF>LF, 10000 is assumed. (4) If limits on the input are exceeded, 30000 is assumed. If 10000 or more is assumed, control escapes from the routine, stopping the calculation.

*1 For double precision subroutines, all real types should be double precision real types.

(3) Calculation method

An integral value is obtained by changing the semi-infinite integral $\int_0^\infty f(x)\cos qx\, dx$ to the finite interval $\int_0^{a+\pi/q} g(x)\cos qx\, dx$, and applying to it the adaptive automatic numerical integration method[1] by Ninomiya based on the Newton-Cotes 9-point rules.

However, suppose

$$g(x)=\sum_{k=0}^{\infty}(-1)^k f_k=\sum_{k=0}^{\infty}(-1)^k f(x+k\pi/q)$$

The value of the function g(x) at a sampling point must be obtained by calculation. Then, if the series f_k decreases very slowly with the increase of k, the calculation of the series $\sum_{k=0}^{\infty}(-1)^k f_k$ does not converge easily.

However, if the series is converted into a fast convergence series by Euler transformation, the value of g(x) can be obtained with a very few number of terms. Actually, the first several terms of the series should be added as they are, and Euler transformation should be applied to the subsequent terms.

In this routine, the series is transformed to

$$\sum_{k=0}^{\infty}(-1)^k f_k=\sum_{k=0}^{5}(-1)^k f_k+\sum_{K=6}^{\infty}(-1)^k\frac{\Delta^k f_6}{2^{k+1}}$$

, and the terms are summed up until $|\Delta^k f_k/2^{k+1}|$ equals $\varepsilon q/\pi$ or less (ε: required absolute precision).

If other than this method is used, an enormous number of function calculations may be required for this kind of integration.

(4) Example

This program obtains the value of

$$\int_1^{\infty}\frac{\cos x}{x}dx$$

using AQCOSS.

```
C       MAIN PROGRAM
        DIMENSION W(100)
        EXTERNAL FUN
        CALL AQCOSS(1.0,1.0,FUN,S,1.0E-4,5000,100,NF,NS,W,ILL)
```

```
      WRITE(6,1000) S,NF,NS,ILL
 1000 FORMAT(1H ,'S=',E15.6,3X,'NF=',I8,3X,'NS=',I8,3X,'ILL=',
     1       I5)
      STOP
      END

C     FUNCTION SUBPROGRAM FOR F(X)
      REAL FUNCTION FUN(X)
      FUN=1.0/X
      RETURN
      END
```

In this example, the result of calculation is

```
S=  -0.337394E+00   NF=      208   NS=       21      ILL=      0
```

If the value of

$$\int_0^{\infty} \frac{\sin x}{x} dx$$

is obtained by using AQSINS (required absolute precision 10^{-4}, the result of calculation is

```
S=   0.157078E+01   NF=      195   NS=       21      ILL=     10
```

The result of these examples is obtained within the required precision.

(5) Note

1. If the lower limit a of an integral is a singular point of the integrand function, and f(x) becomes ∞ at that point, an integral value can be obtained by replacing it with an adequate finite value (0 for example). However, it is more effective to use the adaptive automatic numerical integration method AQNN9S/D in the interval [a, π/q] containing a singular point, and this routine for the remaining interval excluding the singular point.

2. $\int_a^{a+\pi/q} g(x)\cos qx\,dx$ is calculated in the same manner as AQNN9S/D. For details, see the explanation of AQNN9S/D.

3. If the prescribed absolute precision EPS is taken very small as compared with the integral value, the calculation does not converge. EPS should be selected to be the estimated integral value.

4. This routine should be used only when an integral value exists, or f(x) attenuates with the

increase of x. This is because the result of the integration is output in the meaning of summation of divergent series even when the integral value diverges or oscillates (f(x) = constant, for example).

Bibliography

1) Ichizo Ninomiya; Adaptive Automatic Numerical Integration Based on Newton-Cotes 5 (7, 9) Point Rule, Usage Guidance of Library Program pp. 200-202, Nagoya University Computer Center (1982).

2) Moriguchi, Udagawa, and Hitotsumatsu; Mathematical Formula II, p. 34, Iwanami Bookstore (1957).

(1987. 08. 11)

AQCPACK(AQNN5C/B,QDAPBC/B,AQNDC/B,AQNN7C/B,HINFAC/B,AQNN9C/B,INFINC/B,DEFINC/B,AQMDC/B) (Automatic Quadrature for Complex Valued Functions)

Automatic Quadrature for Complex Valued Functions

Programmed by	Ichizo Ninomiya, Takemitsu Hasegawa, and Yasuyo Hatano, August 1982
Format	Subroutine Language; FORTRAN77

(1) Outline

AQCPACK calculates the definite integrals of one, two, and three dimensions of a real variable complex valued function using automatic Quadrature methods. The routines whose name ends with C/B are for 4- and 8-byte complex valued functions.

(2) Directions

AQNN5C/B

CALL AQNN7C/B (A, B, F, S, EPS, LF, NF, ILL)

AQNN9C/B

CALL DEFINC/B(A, B, F, S, EPS, N, ILL)

CALL QDAPBC/B(A, B, F, S, ERR, N, ILL)

CALL HINFAC/B(F, S, EPS, N, ILL)

CALL INFINC/B(F, S, EPS, N, ILL)

CALL AQMDC/B(M, LSUB, F, EPSA, EPSR, NMIN, NMAX, S, ERR, N, ILL)

CALL AQNDC/B(ME, M, AFUN, BFUN, F, EPSA, EPSR, NMIN, NMAX, S, ERR, N, ILL)

The contents of an argument are the same as those of corresponding argument each subroutine whose end character C/B is replaced with S/D. Where, C is an 8-byte complex number type, and B is a 16-byte complex number type.

(3) Calculation method

Each subroutine uses the same calculation method as the corresponding subroutine of a real version. The absolute values (ABS and DABS) are used in the convergence test of an real version. However, the sum of absolute values

$\| x+iy \|_1 = |x| + |y|$

(CABS1 and CDABS1) is used in that of a complex version.

The reason why the sum of absolute values is used instead of the absolute value of usual complex numbers

$\| x+iy \|_2 = \sqrt{x^2+y^2}$

is that the former is much faster and inexpensive.

(4) Example

The following are the program for calculating the definite integral

$$\int_0^x e^{ix} dx$$

by AQNN9B, and its output.

```
      COMPLEX*16 S,FUN
      REAL*8 PI
      EXTERNAL FUN
      PI=3.14159265358979324D0
      CALL AQNN9B(0.D0,PI,FUN,S,1.D-10,2000,NF,ILL)
      WRITE(6,600) S,NF,ILL
600   FORMAT(10X,2D20.10,2I6)
      STOP
      END

      FUNCTION FUN(X)
      COMPLEX*16 FUN
      REAL*8 X
      FUN=DCMPLX(DCOS(X),DSIN(X))
      RETURN
      END
```

< Output result >

```
      0.1387778781D-15      0.2000000000D+01      41      0
```

(5) Note

1. A complex valued function can be calculated with its real and imaginary parts handled separately by using a subroutine for real valued functions, but it is more natural and faster to use present subroutines.

2. It is essential to declare the integrand function and integral value as complex numbers.

(1987. 07. 21) (1987. 08. 21) (1987. 08. 27)

AQDCCS/D, AQDCOS/D

(Automatic quadrature of closed type by Clenshaw-Curtis method) (AQDCCS/D)

(Automatic quadrature of open type by Clenshaw-Curtis method) (AQDCOS/D)

Automatic Quadrature of Closed Type by Clenshaw-Curtis Method (AQDCCS/D)

Automatic Quadrature of Open Type by Clenshaw-Curtis Method (AQDCOS/D)

Programmed by	Tatsuo Torii; July 1978
Format	Subroutine language; FORTRAN Size; 106, 107, 104, and 105 lines respectively

(1) Outline

AQDCCS/D and AQDCOS/D each automatically obtain the approximate value of integral $\int_a^b f(x)dx$ of bounded function $f(x)$ which is smooth in a finite interval (a, b) in the specified precision. The base of this method depends on the expansion of $f(x)$ to a Chebyshev series on the interval $[a,b]$ and on the termwise integration. Therefore, the faster the convergence of this series, the less number of samples this integral method requires to attain the required precision. The smoother the function, the faster the convergence of the Chebyshev series.

If an integrand function is defined on the closed interval $[a,b]$, it is preferable to use the closed-type quadrature of which samples include both end points. If it is given in an open interval, the open-type quadrature must be used.

(2) Directions

CALL AQDCCS/D(A, B, F, EPSA, EPSR, NMIN, NMAX, S, ERR, N, ICON)

CALL AQDCOS/D(A, B, F, EPSA, EPSR, NMIN, NMAX, S, ERR, N, ICON)

Argument	Type and kind	Attribute	Content
A, B	Real type	Input	A and B are lower and upper limits of the integration interval.
F	Real type Function subprogram	Input	The user defines an integrand function as the function subprogram of one variable.

Argument	Type and kind	Attribute	Content
EPSA EPSR	Real type	Input	Required precision. EPSA and EPSR are the limits of absolute and relative errors, respectively (≧0).
NMIN NMAX	Integer type	Input	Lower and upper limits of the number of samples. NMAX≧NMIN≧0 AQDCCS/D: NMAX≦1025 AQDCOS/D: NMAX≦1023
S ERR	Real type	Output	S is an approximate value of the integral to be determined. ERR is an estimated value of the absolute error.
N	Integer type	Output	Number of samples used to compute S.
ICON	Integer type	Output	ICON = 0: Normal. ICON = 10000: Required precision is too severe. The operation result can be regarded as normal because the maximum precision available with the computer used has been already obtained. ICON = 20000: Abnormal. The required precision cannot be obtained even though the number of samples is increased to the limit NMAX. ICON = 30000: Parameter error.

(3) Performance

Fast Fourier cosine transform based on the midpoint rule is used for Chebyshev series expansion of an integrand function. When the number of samples is N, therefore, the number of real multiplications is about $N/2\log_2 N$.

(4) Calculation method

Interval [a, b] is transformed to [-1, 1] by linear transformation, and integrand function $f(t)$ is expanded to a Chebyshev series which is termwise integrated.

$$\int_{-1}^{1} f(t)dt = \frac{1}{2}\int_{-1}^{1}(f(t)+f(-t))dt$$

$$= \int_{-1}^{1} {\sum_{k=0}^{\infty}}' a_{2k} T_{2k}(t)dt$$

$$= a_0 - 2\left(\frac{a_2}{1\cdot 3}+\frac{a_4}{3\cdot 5}+\frac{a_6}{5\cdot 7}+\cdots\right)$$

Errors of closed- and open-type quadratures are evaluated by $(|a_{N-2}|+|a_N|)/N$ with N+1 sample points and $4(|a_{N-4}|+|a_{N-2}|)/N$ with N-1 points, respectively.

Where, coefficient 4 is an expedient.

A "relative error" in each quadrature is the one obtained by dividing each evaluated absolute error by the norm of the integrand function $\| f(t)+f(-t) \|_\infty$, where the function norm is $\| f \|_\infty = max | f(x_j) |$, and x_j is a sample point.

The level of rounding errors (computation or propagation errors) are evaluated by

$$16u \| f(t)+f(-t) \|$$

where u is the minimum unit of machine precision.

Safety coefficient 16 is determined from experience. This completes preparation. The convergence criteria are explained below.

Given required precision values ε_a (absolute error), and ε_r (relative error), at least one of the following conditions is satisfied, it is judged that convergence has attained:

Evaluation value of absolute error $\leqq$ max { ε_a, computation error}

Evaluation value of relative error $\leqq \varepsilon_r$

If neither conditions are satisfied, the number of samples is doubled each time like 17, 33, 65, and so on, in case of closed-type quadrature, or 15, 31, 63, and so on, in case of open-type quadrature.

If $\varepsilon_a=\varepsilon_r=0$ is given, the result with the highest precision (rounding errors are predominant over truncation errors) can be obtained.

The validity of the above convergence criteria depends on the smoothness of integrand function. If integrand function $f(t)$ is sufficiently smooth, error evaluation is successful with less number of samples required (about 2^5). If $f(t)$ cannot be differentiable, however, more number of samples are needed and the error evaluation value tends to be too lower than the actual one. Even if $f(t)$ is analytic on the interval [-1, 1] in the real axis, the similar situation occurs as the singular point approaches [-1, 1].

(5) Example

As the test, we use the following three kinds of problems whose analytic solutions are known:

(1) $$\int_{-1}^{1} \frac{1-t^2}{1-2tx+t^2} dx = \left(\frac{1}{t}-t\right) \log \frac{1+t}{1-t} \quad t=1/2,3/4,15/16$$

(2) $\int_{-1}^{1}\frac{a}{a^2+x^2}dx=2\tan^{-1}\frac{1}{a}$ $a=1,1/4,1/16$

(3) $\int_{-1}^{1}\cos ax\,dx=\frac{2}{a}\sin a$ $a=4,16,64$

The following program performs the above integral calculations changing required precision ε to $10^{-2},10^{-4},10^{-6},\cdots$ and prints the index (ICON) which indicates whether operations have been done normally for calculated values, errors, error evaluation, number of samples, and calculation.

```
C     TEST PROBLEMS FOR SUBROUTINE AQDCCS AND AQDCOS.
C     1978.11.15
      DIMENSION PARAM(3,3)
      DATA PARAM/0.5,0.75,0.9375,1.0,0.25,0.0625,4.0,16.0,64./
      COMMON T,J
      EXTERNAL F
      ZERO=AMACH(ZERO)
      EPSA=1.0E-02
      EPSR=0.0
      NMIN=0
      NMAX=1025
      A=-1.0
      B=1.0
   10 WRITE(6,600) EPSA
  600 FORMAT(1HO/4X,32HPERMISSIBLE ABSOLUTE ERROR BOUND,E15.3/)
      DO 20 J=1,3
      DO 20 I=1,3
      T=PARAM(I,J)
      CALL AQDCCS(A,B,F,EPSA,EPSR,NMIN,NMAX,S,ERR,N,ICON)
      TS=TRUE(T,J)
      ERROR=TS-S
      WRITE(6,601)J,I,T,TS,S,ERROR,ERR,N,ICON
  601 FORMAT(1H,2I4,F8.4,2F15.06,2E13.03,2I8,5X,6HAQDCCS)
      CALL AQDCOS(A,B,F,EPSA,EPSR,NMIN,NMAX,S,ERR,N,ICON)
      ERROR=TS-S
      WRITE(6,602) J,I,T,TS,S,ERROR,ERR,N,ICON
  602 FORMAT(1H ,2I4,F8.4,2F15.06,2E13.03,2I8,5X,6HAQDCOS/)
   20 CONTINUE
      EPSA=EPSA*1.0E-02
      IF(EPSA.GT.ZERO)GO TO 10
      STOP
      END

      FUNCTION F(P)
      COMMON T,J
      GO TO (1,2,3),J
    1 F=(1.0-T*T)/(1.0-2.0*T*P+T*T)
      RETURN
```

```
2 F=T/(T*T+P*P)
  RETURN
3 F=COS(T*P)
  RETURN
  END

  FUNCTION TRUE(P,J)
  GO TO (1,2,3),J
1 TRUE=(1.0/P-P)*ALOG((1.0+P)/(1.0-P))
  RETURN
2 TRUE=2.0*ATAN(1.0/P)
  RETURN
3 TRUE=2.0*SIN(P)/P
  RETURN
  END
```

Calculation result under required precision 10^{-5}

Problem	Parameter	Kind	Calculated integral value	Error	Error evaluation	Number of samples	Normal or Abnormal
(1)	t=1/4	AQDCCS	1.647918	0.0	0.781E-06	33	0
		AQDCOS	1.647918	0.0	0.781E-06	31	0
	t=3/4	AQDCCS	1.135114	0.596E-07	0.166E-05	65	10000
		AQDCOS	1.135114	0.298E-07	0.166E-05	63	10000
	t=15/16	AQDCCS	0.443557	0.104E-06	0.714E-05	257	10000
		AQDCOS	0.443548	0.858E-05	0.647E-05	127	10000
(2)	a=1	AQDCCS	1.570796	-0.298E-07	0 531E-06	17	0
		AQDCOS	1.570796	-0.298E-07	0.472E-06	31	0
	a=1/4	AQDCCS	2.651635	0.596E-07	0.184E-05	65	10000
		AQDCOS	2.651635	0.119E-06	0.184E-05	63	10000
	a=1/16	AQDCCS	3.016755	0.179E-06	0.735E-05	257	10000
		AQDCOS	3.016755	0.238E-06	0.735E-05	255	10000
(3)	a=4	AQDCCS	-0.378401	0.0	0.469E-06	17	0
		AQDCOS	-0.378401	0.0	0.476E-06	31	0
	a=16	AQDCCS	-0.035988	-0.484E-07	0.477E-06	33	0
		AQDCOS	-0.035988	-0.829E-07	0.477E-06	63	0
	a=64	AQDCCS	0.028751	-0.424E-07	0.477E-06	129	0
		AQDCOS	0.028751	0.405E-07	0.477E-07	127	0

(1987.05.21) (1987.08.08)

AQIOSC/B (Automatic quadrature of oscillatory infinite integral of complex-valued function)

Automatic Quadrature of Oscillatory Infinite Integral of Complex-Valued Function

Programmed	Takemitsu Hasegawa; February 1986
Format	Subroutine language; FORTRAN Size; 698 and 704 lines respectively

(1) Outline

When a complex-valued function f(x) is given, AQIOSC or AQIOSB calculates the approximate value of the oscillatory infinite integral

$$I=\int_a^\infty f(x)e^{i\omega x}dx, \qquad a\geqq 0$$

with requested absolute error ε_a.

AQIOSC is a routine for single precision and AQIOSB is one for double precision.

(2) Directions

CALL AQIOSC/B(A, OMEGA, FUN, EPSA, NMIN, NMAX, SC, NFUN, ERR, ICON)

Argument	Type and kind (*1)	Attribute	Content
A	Real type	Input	Lower limit of integral domain. $A\geqq 0$
OMEGA	Real type	Input	Frequency ω. $2\pi\|\omega\|>1.E-7$ (single precision) and $2\pi\|\omega\|>1.E-15$ (double precision). $\omega>0$

FUN	Complex type function subprogram	Input	Given function f(x). For the function as an actual argument of this function, the user should prepare a function subprogram having a variable x.
EPSA	Real type	Input	Requested absolute error ε_a for approximate value SC or SS of an integral. EPSA>0
NMIN NMAX	Integer type	Input	Lower limit (NMIN) and upper limit (NMAX) of the number of function FUN evaluations. NMIN is usually set to 9. NMAX is usually set to 200 to 900. When NMAX≧513, NMAX is assumed to be 513 (single precision). When NMAX≧2049, NMAX is assumed to be 2049 (double precision). 0<NMIN<NMAX.
SC	Real type	Output	Approximate value of integral I.
NFUN	Integer type	Output	Total number of function FUN evaluations.
ERR	Real type	Output	Estimation of absolute error of SC
ICON	Integer type	Output	ICON=0; Normal termination. ICON=1000, 10000, or 11000; The accuracy of the approximate value of an integral has reached the level of rounding error. ICON=2000, 12000, 21000, or 22000; No approximate value satisfied the requested accuracy (EPSA) even after NMAX function evaluations are used. ICON=30000; Parameter error.

*1 For double precision subroutines, all real types should be changed to double precision real types. All complex types should be changed to double precision complex types.

(3) Calculation method

Integral

$$I=\int_a^\infty f(x)e^{i\omega x}dx=\sum_{n=0}^{\infty}Sn$$

is represented as

$$S_n=\int_{x_{n-1}}^{x_n} f(x)e^{i\omega x}dx$$

, where $a<x_0<x_1<x_2<\cdots$ is the root of $\sin\omega x=0$.

(a) Set {Sn} of the approximate value of each Sn is efficiently calculated by using Chebyshev polynomial expansion of f(x).

(b) Alternating series with slow convergence,

$$\sum_{n=0}^{\infty}S_n$$

, is subjected to the Sidi acceleration method, a generalized Richardson extrapolation, to improve the convergence.

By combining these two methods (a) and (b), the approximate value of an integral can be obtained efficiently.

(4) Example

When ω=1, 11, 21, ···, 91 for the values of the parameter α being 1, 4, 7,

$$I=\int_0^\infty e^{-\alpha x}(10+i)e^{i\omega x}dx$$

is calculated. $\varepsilon_a=10^{-4}$.

```
C       EXAMPLE FOR AQIOSC
C       FEBRUARY 15,1986
        IMPLICIT COMPLEX*8(C)
        COMMON ALPHA
        EXTERNAL CFUN
```

```
      A=0.E0
      EPSA=1.E-4
      NMIN=9
      NMAX=400
      WRITE(6,1000)
 1000 FORMAT(1H0,'TEST '///1H ,'ALPHA OMEGA', 8X,'REAL',
     * 9X,'IMAGINARY',6X,'N',5X,'ERR      ICON')
      DO 20 IALPHA=1,7,3
      ALPHA=FLOAT(IALPHA)
      WRITE(6,1010)
 1010 FORMAT(1H )
      DO 10 IOMEGA=1,100,10
      OMEGA=FLOAT(IOMEGA)
      CALL AQIOSC(A,OMEGA,CFUN,EPSA,NMIN,NMAX,CSS,NFUN,
     *ERR,ICON)
      WRITE(6,1020) ALPHA,OMEGA,CSS,NFUN,ERR,ICON
 1020 FORMAT(1H ,F5.2,F6.2,2E16.7,I5,E10.2,I7)
   10 CONTINUE
   20 CONTINUE
      STOP
      END
C
      FUNCTION  CFUN(X)
      IMPLICIT COMPLEX*8(C)
      COMMON ALPHA
      CFUN=EXP(-ALPHA*X)*(10.E0,1.E0)
      RETURN
      END
```

As shown in the above example, when the integrand function contains a parameter (α in this example), the parameter is put in the common area to communicate with the main program.

(5) Note

1. When there is a point x=p (or a sharp peak point) that the function f(x) is singular or near singular in the integration interval [a, ∞), this method should be used for integrals in the interval [p+δ, ∞) (δ>0) beyond this point. For integrals in the [a, a+δ] interval, however, another method should be used.

Bibliography

1) Takemitsu Hasegawa and Tatsuo Torii; "Oscillatory semi-infinite integral based on Chebyshev series expansion", Preprints of Working Group for Numerical Analysis, IPSJ 10-3 (1984).

(1987.08.05)

AQIOSS/D (Automatic quadrature of oscillatory infinite integral)

Automatic Quadrature of Oscillatory Infinite Integral

Programmed by	Takemitsu Hasegawa; February 1985
Format	Subroutine language; FORTRAN Size; 669 and 677 lines respectively

(1) Outline

When a constant-sign function f(x) is given, AQIOSS or AQIOSD calculates the approximate value of the oscillatory infinite integral

$$I^c=\int_a^\infty f(x)\cos\omega x dx, \qquad I^s=\int_a^\infty f(x)\sin\omega x dx, \qquad a\geqq 0$$

with requested absolute error ε_a.

AQIOSS is a routine for single precision and AQIOSD is one for double precision.

(2) Directions

CALL AQIOSS/D(A, OMEGA, FUN, KEY, EPSA, NMIN, NMAX, SC, SS, NFUN, ERR, ICON)

Argument	Type and kind (*1)	Attribute	Content
A	Real type	Input	Lower limit of integral domain. A≧0
OMEGA	Real type	Input	Frequency ω. $2\pi\|\omega\|>1.E-7$ (single precision) and $2\pi\|\omega\|>1.E-15$ (double precision)
FUN	Real type function subprogram	Input	Given function f(x). For the function as an actual argument of this function, the user should prepare a function subprogram having a variable x.

KEY	Integer type	Input	KEY should be set to 0, 1, or 2 for obtaining the cosine integral Ic, the sine integral Is, or both, respectively. $0 \leqq KEY \leqq 2$
EPSA	Real type	Input	Requested absolute error ε_a for approximate integral SC or SS. EPSA>0
NMIN NMAX	Integer type	Input	Lower limit (NMIN) and upper limit (NMAX) of the number of function FUN evaluations. NMIN is usually set to 9. NMAX is usually set to 200 to 900. When $NMAX \geqq 513$, NMAX is assumed to be 513 (single precision). When $NMAX \geqq 2049$, NMAX is assumed to be 2049 (double precision). $0<NMIN<NMAX$.
SC SS	Real type	Output	Approximate value (SC) of cosine integral Ic and approximate value (SS) of sine integral Is.
NFUN	Integer type	Output	Total number of function FUN evaluations.
ERR	Real type	Output	Estimated value of absolute error for SC and SS.
ICON	Integer type	Output	ICON=0; Normal termination. ICON=1000, 10000, or 11000; The accuracy of the approximate value of an integral has reached the level of rounding error. ICON=2000, 12000, 21000, or 22000; No approximate value satisfied requested precision (EPSA) even after the number of function evaluations reached NMAX. ICON=30000; Parameter error.

*1 For double precision subroutines, all real types should be changed to double precision real types.

(3) Calculation method

$$I^c=\int_a^\infty f(x)\cos\omega x dx=\sum_{n=0}^{\infty}Sn$$

cosine integral is represented as follows:

$$S_n=\int_{x_{n-1}}^{x_n} f(x)\cos\omega x dx, \qquad S_0=\int_a^{x_0} f(x)\cos\omega x dx,$$

where $a<x_0<x_1<x_2<\cdots$ is the root of $\cos\omega x=0$.

(a) Set {Sn} of the approximate value of each Sn is efficiently calculated by using Chebyshev polynomial expansion of f(x).

(b) Alternating series with slow converence,

$$\sum_{n=0}^{\infty}S_n$$

, is subjected to the Sidi acceleration method, a generalized Richardson extrapolation, to improve the convergence.

By combining these two methods (a) and (b), the approximate value of an integral can be obtained efficiently.

Sine integral Is is calculated in the same way.

(4) Example

When ω=1, 3, 5, 7, 9 for the value of parameter α being 1, 2, 3,

$$I^c=\int_0^\infty e^{-\alpha x}\cos\omega x dx, \qquad I^s=\int_0^\infty e^{-\alpha x}\sin\omega x dx$$

is calculated. $\varepsilon a=10^{-4}$.

```
C       EXAMPLE FOR AQIOSS
        COMMON ALPHA
        EXTERNAL FUN
        A=0.DO
        EPSA=1.E-4
        KEY=2
```

```
      NMIN=9
      NMAX=200
      WRITE(6,1)
    1 FORMAT(1HO,'TEST FOR AQIOSS'///1H ,'ALPHA OMEGA',7X,
     *'COSINE',9X,'SINE',8X,'N',7X,'ERR  ICON')
      DO 10 IALPHA=1,3
      ALPHA=FLOAT(IALPHA)
      WRITE(6,2)
    2 FORMAT(1H )
      DO 20 IOMEGA=1,9,2
      OMEGA=FLOAT(IOMEGA)
      CALL AQIOSS(A,OMEGA,FUN,KEY,EPSA,NMIN,NMAX,
     *SC,SS,N,FUN,ERR,ICON)
      WRITE(6,3) ALPHA,OMEGA,SC,SS,N,FUN,ERR,ICON
    3 FORMAT(1H ,F5.2,F6.2,2E15.7,I5,E10.2,I7)
   20 CONTINUE
   10 CONTINUE
      STOP
      END
C
      FUNCTION FUN(X)
      COMMON ALPHA
      FUN=EXP(-ALPHA*X)
      RETURN
      END
```

As shown in the above example, when the integrand function contains a parameter (α in this example), the parameter is put in the common area to communicate with the main program.

(5) Notes

1. When there is a point x=p (or a sharp peak point), at which the function f(x) is singular or near singular in the integration interval [a, ∞), this method should be used for integrals in the interval [p+δ, ∞) (δ>0) beyond this point. For integrals in the [a, a+δ] interval, however, another method should be used.

2. When both cosine integral Ic and sine integral Is are needed for the same function f(x), calculation of the function f(x) can be used commonly. This method thus has an advantage that both approximate values can be obtained by the time needed for function calculation of either Ic or Is.

Bibliography

1) Takemitsu Hasegawa and Tatsuo Torii; "Oscillatory semi-infinite integral based on Chebyshev series expansion", Preprints of Working Group for Numerical Analysis, IPSJ 10-3 (1984).

(1987.08.05)

AQMDS/D (Automatic multiple integration based on the interpolatory type quadrature increasing the sample points with arithmetical progression)

Automatic Multiple Integration Based on the Interpolatory Type Quadrature Increasing the Sample Points with Arithmetical Progression

Programmed by	Takemitsu Hasegawa: April 1980
Format	Subroutine language; FORTRAN Size; 562 and 563 respectively

(1) Outline

AQMDS and AQMDD are automatic integration routines that calculate multiple integration

$$I=\int_{\psi_1}^{\phi_1} dx_1 \int_{\varphi_2}^{\phi_2} dx_2 \cdots\cdots \int_{\varphi_m}^{\phi_m} dx_m f(x_1, x_2, \cdots, x_m)$$

in a curved boundary region to obtain approximate value S with precision satisfying

$$|S-I| \leq \max(\varepsilon_a, \varepsilon_r |I|)$$

, where ε_a is an absolute error and ε_r is a relative error.

It uses a product formula that repeatedly applies an interpolatory type quadrature increasing sample points with arithmetical progression in each coordinate axial direction. (For the interpolatory type quadrature, this routine uses an open formula which does not use both ends of an integration interval as sample points. QDAPBS/D uses a closed formula. AQMDS/D uses an open formula to handle functions which are near singular at the ends of the integration interval, as well as smooth functions. The product formula is effective for smooth or oscillatory-type functions, in paticular.

(2) Directions

CALL AQMDS/D(M, LSUB, FUN, EPSA, EPSR, NMIN, NMAX, S, ERR, N, ICON)

Argument	Type and kind (*1)	Attribute	Content
M	Integer type	Input	Multiplicity of integral calculus. $1 \leq M \leq 3$

Argument	Type and kind (*1)	Attribute	Content
LSUB	Subroutine subprogram	Input	Name of the subroutine subprogram that calculates upper and lower limits of integration. Number k in the direction of coordinate axis x_k, on which integration is being done, is put into the first argument (K). The second argument (X) is the name of an one-dimensional array having M elements. Values of x_1 and x_2 enters X(1) and X(2). The lower limit of integration is put into the third argument (A), and the upper limit is put into the fourth argument (B). This subprogram must be declared in the EXTERNAL statement in the main program.
FUN	Real type Function subprogram	Input	Name of an integrand. This function needs to have only one one-dimensional array having M elements as an actual argument (X). Value of x_i is put into X(i). ($1 \leqq i \leqq M$). This function subprogram must be declared in the EXTERNAL statement in the main program.
EPSA EPSR	Real type	Input	Requested absolute error ε_a (EPSA) and relative error ε_r (EPSR) for approximate value S of an integral. EPSA≧0, EPSR≧0.
NMIN NMAX	Integer type	Input	Lower limit (NMIN) and upper limit (NMAX) of the number of times the integrand function FUN is to be evaluated for an integral in the direction of each coordinate axis. NMIN=7 and NMAX=100 (in case of AQMDS) or NMAX=511 (in case of AQMDD) are suitable. When NMAX≧511 is specified, NMAX=511 is assumed.
S	Real type	Output	Approximate value of an integral.
ERR	Real type	Output	Estimation of the absolute error of S.
N	Integer type	Output	Total number of evaluations of the integrand function FUN.
ICON	Integer type	Output	ICON=0: Normal termination. ICON=30000: Parameter error. If integration in the direction of each coordinate axis does not converge even if NMAX function evaluations are used, ICON is set as follows: ICON=200 when the coordinate axis is x_3, ICON=2000 when the coordinate axis is x_2, ICON=20000 when the coordinates axis is x_1. If requested accuracy is too high and, as the result of integration in the direction of a certain coordinate axis, the accuracy of the approximate value of the integral has reached the level of the rounding error of the computer, ICON is set as follows: ICON=100 when the coordinate axis is x_3, ICON=1000 when the coordinate axis is x_2, ICON=10000 when the coordinate axis is x_1. If two or more such events occur simultaneously, ICON is set to the sum of the respective values.

*1 For double precision subroutines, all real types should be changed to double precision real types.

(3) Example

The program below calculates the following triple integral with the value of parameter p allowed varied:

$$\int_{-1}^{1} dx_1 \int_{-2}^{2} dx_2 \int_{-3}^{3} dx_3 \frac{1}{48(4-\cos px_1-\cos px_2-\cos px_3)}$$

```
C       EXAMPLE ...AQMDS...
        EXTERNAL FUN,LSUB
        COMMON P
        EPSA=1.0E-4
        EPSR=0.0
        NMIN=7
        NMAX=100
        M=3
        DO 10 IP=1,10
        P=FLOAT(IP)*0.5
        CALL AQMDS(M,LSUB,FUN,EPSA,EPSR,NMIN,NMAX,S,ERR,N,ICON)
   10   WRITE(6,100) P,S,ERR,N,ICON
  100   FORMAT(1H ,'P=',F4.1,5X,'S=',E15.7,5X,'ERR=',E10.2,5X,
       *'N=',I7,5X,'ICON=',I5)
        STOP
        END
C
        FUNCTION FUN(X)
        DIMENSION X(3)
        COMMON P
        FUN=1.0/(4.0-COS(P*X(1))-COS(P*X(2))-COS(P*X(3)))/48.0
        RETURN
        END
C
        SUBROUTINE LSUB(K,X,A,B)
        DIMENSION X(3)
        GO TO (1,2,3),K
      1 A=-1.0
        B=1.0
        RETURN
      2 A=-2.0
        B=2.0
        RETURN
      3 A=-3.0
        B=3.0
        RETURN
        END
```

As shown in this example, if the integrand function contains a parameter (p in this example), it is put in a common region to communicate with the main program.

(4) Performance

With EPSA=1.0D-7, we tested the following three triple integrals using double precision subroutine AQMDD. The results are as follows.

$$A \quad \int\int_D\int \prod_{i=1}^{3} \frac{p}{x_i^2+p^2} dx_1 dx_2 dx_3, \qquad p=1, \frac{1}{2}, \frac{1}{4}$$

$$B \quad \int\int_D\int \prod_{i=1}^{3} \frac{1-p^2}{1-2px_i+p^2} dx_1 dx_2 dx_3, \qquad p=\frac{1}{4}, \frac{1}{2}, \frac{3}{4}$$

$$C \quad \int\int_D\int \prod_{i=1} \cos px_i dx_1 dx_2 dx_3, \qquad p=8, 16, 32$$

where region D is $[-1,1]^3$.

Problem	A			B			C		
p	1	1/2	1/4	1/4	1/2	3/4	8	16	32
Number of samples	12, 167	59, 487	350, 847	11, 215	29, 791	223, 543	29, 663	65, 151	272, 199

The three values for parameter p in each problem correspond, from left to right, to (integration is) "easy," "rather difficult," and "difficult".

(5) Notes

1. When this routine is called several times, it calculates weight of a one-dimensional integral and sample points only when it is called for the first time. So, it can save time a little in calculation when it is called second and subsequent time.

2. If ICON is other than 0 and 30000, the integration results do not satisfy requested precision, but the absolute error can be estimated from argument ERR.

3. If ICON is 200, 2000, or 20000, requested accuracy may be satisfied by increasing the NMAX value.

4. Multiple integration generally uses many sample points, resulting in remarkable accumulation of rounding errors. Generally speaking, therefore, AQMDD can be used when EPSA and EPSR are less than 0.5E-4. If satisfactory precision cannot be obtained with AQMDS when NMAX≧60, use of AQMDD may improve precision.

Bibliography

1) Tatsuo Torii, Takemitsu Hasegawa, and Ichizo Ninomiya; "Interpolatory automatic integration increasing sample points with arithmetical progression" Information processing, Vol. 19, No. 3, and pp. 248-255 (1978).

2) Takemitsu Hasegawa, Tatsuo Torii, and Ichizo Ninomiya; "Automatic multiple integration with less number of sample points", Preprints of the 21th Symposium of Information Processing Soc. of Japan, pp. 951 (1980).

3) Ichizo Ninomiya; "Newly registered numerical analysis software", Nagoya University Computer Center News, Vol. 10, No. 3, pp. 278-308 (1979).

4) Takemitsu Hasegawa; "Automatic multiple integration based on interpolatory type quadrature increasing sample points with arithmetical progression", Nagoya University Computer Center News, Vol. 11, No. 4, p. 413 (1980).

(87. 05. 26)

AQNDS/D, AQ3DS/D, AQ2DS/D, and AQ1DS/D (Automatic multiple quadrature)

Automatic Multiple Quadrature

Programmed by	Ichizo Ninomiya, Takemitsu Hasegawa, and Yasuyo Hatano: March 1979
Format	Subroutine language; FORTRAN Size; 622 and 623 lines respectively

(1) Outline

Suppose we calculate multiple integrals:

$$I=\int_{l_1}^{u_1} dx_1 \cdots \int_{l_n}^{u_n} dx_n \cdot f(x_1 \cdots x_n) \qquad (1)$$

, where

$$u_1=a, u_2=u_2(x_1), u_3=u_3(x_1,x_2)\cdots \qquad (2)$$

$$l_1=b, l_2=l_2(x_1), l_3=l_3(x_1,x_2)\cdots$$

Now, $\varepsilon_a, \varepsilon_r$, which are the upper limits of absolute and relative errors for approximate value S of integral I, are given to calculate S satisfying

$$|S-I| \leq max(\varepsilon_a, \varepsilon_r |I|) \qquad (3)$$

. We use a product formula by which various automatic integration methods for one variable are repeatedly used in each dimensional direction. The following six types of automatic formulas are available for one variable. They can be used in arbitrary combinations.

(1) Adaptive Newton-Cotes 9-point rule.

(2) Clenshaw-Curtis integration—Formula that adds data points in a geometric progression (common ratio $\sqrt{2}$).

(3) Double exponential function type integration formula.

(4) Double exponential function type integration formula (semi-infinite interval).

(5) Double exponential function type integration formula (infinite interval).

(2) Directions

```
CALL AQNDS/D(ME, M, AFUN, BFUN, FUN, EPSA, EPSR, NMIN, NMAX, S, ERR, N, ICON)

CALL AQ3DS/D(ME, AFUN, BFUN, FUN, EPSA, EPSR, NMIN, NMAX, S, ERR, N, ICON)

CALL AQ2DS/D(ME, AFUN, BFUN, FUN, EPSA, EPSR, NMIN, NMAX, S, ERR, N, ICON)

CALL AQ1DS/D(ME, AFUN, BFUN, FUN, EPSA, EPSR, NMIN, NMAX, S, ERR, N, ICON)
```

Argument	Type and kind (*1)	Attribute	Content
MB	Integer type One-dimensional array	Input	One-dimensional array with M number of elements. The integration formula to be used for each dimensional direction is specified by the number. 1≦MB≦5. When MB=1, the adaptive Newton-Cotes 9-point rule is used. When MB=2, the Clenshaw and Curtis formula is used. When MB=3, the double exponential function formula (finite interval) is used. When MB=4, the double exponential function formula (semi-infinite interval) is used. When MB=5, the double exponential function formula (infinite interval) is used.
M	Integer type	Input	Multiplicity of integration. M=1 is assumed for AQ1DS/D, M=2 is assumed for AQ2DS/D, and M=3 is assumed for and AQ3DS/D. 1≦M≦3
AFUN BFUN	Real type Function subprogram	Input	Name of a function subprogram for calculating the lower limit (AFUN) and upper limit (BFUN) of a definite integral. Each has two arguments. The first argument (X) is the name of a one-dimensional array having M number of elements. The value of x_1 is set in X(1), the value of x_2 is set in X(2), and the value of x_3 is set in X(3). The second argument (K) contains the number of the dimensional direction in which calculation is being performed. When the region of the definite integral is defined by the function of an integration variable, the values of these arguments are called to define the values of AFUN and BFUN. Both arguments need to be declared in the EXTERNAL statement in the calling program.
FUN	Real type Function subprogram	Input	Name of a function subprogram that calculates integrand function f. It must be a function of only one one-dimensional array having M number of elements. This argument needs to be declared in the EXTERNAL statement in the calling program.
EPSA EPSR	Real type	Input	Upper limit ε_a of absolute error and upper limit ε_r of relative error of approximate value S of an integral. $\varepsilon_a, \varepsilon_r \geq 0$
NMIN NMAX	Integer type	Input	Lower and upper limits of the number of evaluations of f in each dimensional direction. NMIN=10 and NMAX=100 are suitable. However, if the value conflicts with those specific to the component routine for each integration formula, it is automatically replaced with a standard value.
S	Real type	Output	Approximate value of integral.
ERR	Real type	Output	Estimated absolute error of S.
N	Integer type	Output	Number of actual evaluations of f.

Argument	Type and kind (*1)	Attribute	Content
ICON	Integer type	Output	Condition code. The termination state of integration for x_1 is indicated at the place of 10,000, that for x_2 at the 100, and that for x_3 at the place of 1. Each time one of the following events occurs, the number given to it is added to each place. (1) Normal termination: 0 (2) Integration does not converge even when NMAX is exceeded: 2 (3) The event in (2) exceeds the number of times obtained by NMAX/10: 20 (4) An error other than (2) and (3) occurs: 1 (5) The event in (4) exceeds the number of times obtained by NMAX/10: 10 For the above events, S and ERR indicate an approximate value and estimated value of an error respectively. If N exceeds MAX=min(NMAX**M, 1000000), 5000 is added to the above-mentioned value. ICON=30000 indicates a parameter error.

*1 For double precision subroutines, all real types should be changed to double precision real types.

(3) Example

The program shown below uses the Clenshaw-Curtis formula for x_1, and double exponential function type formula for x_2 for the following definite integral:

$$\int_0^1 dx_1 \int_0^{1-x1} \frac{dx2}{\sqrt{x1+x2}}$$

```
C *** EXAMPLE (AQ2DS) ***
      EXTERNAL FUN,AFUN,BFUN
      DIMENSION ME(2)
      ME(1)=2
      ME(2)=3
      CALL AQ2DS(ME,AFUN,BFUN,FUN,1.E-3,1.E-3,10,100,S,ERR,N,
     *ICON)
      WRITE(6,610) S,ERR,ICON
  610 FORMAT(1H ,'S =',F13.5,'  ERR =',E10.2,'  ICON =',I6)
      STOP
      END
C
      FUNCTION AFUN(X,K)
      AFUN=0.0
      RETURN
      END
C
      FUNCTION BFUN(X,K)
      DIMENSION X(2)
      BFUN=1.0
      IF(K.EQ.2) BFUN=BFUN-X(1)
```

```
      RETURN
      END
C
      FUNCTION FUN(X)
      DIMENSION X(2)
      Y=ABS(X(1)+X(2))
      IF(Y.LT.1.E-70) GO TO 2
      FUN=1.0/SQRT(Y)
    1 RETURN
    2 FUN=0.0
      GO TO 1
      END
```

(4) Note

Read paper in bibliography [1),2)] for details of an automatic integration formula for one variable and the corresponding component subroutine. For selection of integration formulas, read paper in bibliography [1),4)].

Bibliography

1) Ninomiya and Hatano; "New SSL program", Nagoya University Computer Center News, Vol. 8, No. 3, p. 209 (1977).

2) Hasegawa, Torii, and Ninomiya; "Clenshaw-Curtis type integration increasing sample points with geometrical progression", Preprints of the 19th Symposium of Information Processing Soc. of Japan, p. 391 (1978).

3) Hatano, Hasegawa, and Ninomiya; "Creating automatic multiple integration subroutines", Preprints of the 20th Symposium of Information Processing Soc. of Japan, p. 447 (1978)

4) Ninomiya and Hatano; "State of use of library programs and selection of programs", Nagoya University Computer Center News, Vol. 11, No. 4, p. 399 (1980).

(87. 07. 31)

AQNN5S/D, AQNN7S/D, AQNN9S/D/Q

(Adaptive quadrature based on Newton-Cotes 5(7, 9) point rule)

Adaptive Quadrature Based on Newton-Cotes 5(7, 9) Point Rule

Programmed	Ichizo Ninomiya February, 1978
Format	Subroutine Language; FORTRAN Size; 93, 94, and 99, 100, 103, 104 lines

(1) Outline

Given integrand $f(x)$, lower limit a, upper limit b, and requested accuracy ε, the definite integral $\int_a^b f(x)dx$ is evaluated with the absolute error tolerance ε by using the adaptive quadrature. Adaptive quadrature is a very effective method adjusting the density of the sample points according to the behavior of the integrand. This routine is based on the Newton-Cotes 5(7, 9) point rule. A lot of new modifications (the error estimation, distribution of error to local subinterval, detecting and treatment of discontinuities and singularities etc.) are added, and it has higher reliability and requires the smaller number of samples compared with the existing methods. [2)]

(2) Directions

CALL AQNN5S/D(A, B, FUNC, S, EPS, LF, NF, ILL)

CALL AQNN7S/D(A, B, FUNC, S, EPS, LF, NF, ILL)

CALL AQNN9S/D/Q(A, B, FUNC, S, EPS, LF, NF, ILL)

Argument	Type and Kind *	Attribute	Content
A	Real type	Input	Lower limit of definite integral.
B	Real type	Input	Upper limit of definite integral. A<B is required.
FUNC	Real type Function subprogram	Input	Name of integrand. The user should prepare the function as the actual argument corresponding to this argument as a function subprogram with the integration variable as the only one argument.

Argument	Type and Kind *	Attribute	Content
S	Real type	Output	The value of definite integral is output.
EPS	Real type	Input	Positive number representing requested precision. For single precision 10^{-3}~10^{-6} is reasonable, and for double precision 10^{-5}~10^{-15}.
LF	Integer type	Input	Upper bound of sample frequency of functions. LF>12. Several thousands are suitable.
NF	Integer type	Output	Sample frequency of functions. The calculation is interrupted, and the control escapes from the routine when NF>LF.
ILL	Integer type	Output	The situation of the calculation in the routine is output. It is set to 0 first in the routine, and is increased by adding constants discribed below each time one of the following cases occurs. (1) When length of small subinterval becomes extremely small : 1 (2) When a discontinuity is detected : 10 (3) When a logarithmic singularity is detected : 100. (4) When an algebraic singularity is detected : 1000. (5) When NF>LF : 10000 (6) When the order of algebraic singularity is -1 or less : 20000 (7) When the input limitation is violated : 30000 When (5), (6), or (7) occurs, the calculation is interrupted.

* All real types should be changed to be double (quadruple) precision real number types in the case of double (quadruple) precision subroutine.

(3) Performance

Integrand is classified into the following five types according to the characteristic of its behavior.

(1) Smooth type : The function is smooth, and the change is slow.

(2) Peak type : There are steep peaks and valleys.

(3) Oscillatory : There is a violent vibration with short wave length.

(4) Discontinuous type : There is a discontinuous point in the value of function or the derivative.

(5) Singular type : There are logarithmic singularities or algebraic singularities.

This routine is strong for the peak type as well as the smooth type because of the locality of its algorithm. Even the integrands of the discontinuous type or the singular type can be handled effectively by this routine, when abnormal points are located at easily detectable places such as the ends or the middle point of integration interval. Hoever, this routine is comparatively weak for oscillatory type integrands. Though the evaluation frequency of integrand may increase, sometimes this routine is robust because it always gives an appropriate integral value.

Out of three routines, 9-point rule is recommended because of its high reliability. The following table shows the result of experiment of 21 test problems [1] of Kahaner, where reliability is the percentage of the cases where calculated value of definite integral actually satisfies requested accuracy.

Requested accuracy	10^{-3}		10^{-6}		10^{-9}	
Routine name	Reliability (%)	Average Number of samples	Reliability (%)	Average Number of samples	Reliability (%)	Average Number of samples
AQNN5D	95	61	90	106	90	346
AQNN7D	86	51	90	105	86	237
AQNN9D	95	82	95	123	90	234
QNC7*	86	79	86	201	81	437
QUAD**	95	149	90	269	86	465

* QNC7 is a subroutine based on 7-point method by Kahaner.

** QUAD is a subroutine based on 10-point method by Kahaner.

(4) Examples

The following shows a part of a program to calculate the value of definite integral $\int_0^1 (x^{-a}+1+\sin x)dx$ with AQNN9S changing values of a from 0.1 to 0.9 in stepsize of 0.1.

```
C       MAIN PROGRAM
        COMMON A
        EXTERNAL FUN
        DO 10 I=1,9
        A=FLOAT(I)*0.1
        CALL AQNN9S(0.0,1.0,FUN,S,1.E-4,5000,NF,IND)
         :
   10   CONTINUE
        STOP
        END

C       FUNCTION SUBPROGRAM FOR INTEGRAND
        FUNCTION FUN(X)
        COMMON A
        FUN=1.+SIN(X)
        IF(X.GT.0.) FUN=X**(-A)+FUN
        RETURN
        END
```

The auxiliary variable in integrand program (A in this example) is allocated in the COMMON region to make communication between main program and integrand subprogram. When the function value becomes infinity in the singular point, it is necessary to replace it with a suitable finit value (0 for instance) to use this routine.

(5) Note

1. If necessary, it is desirable to normalize the length of the integration interval and the absolute value of the integral value to the order of unity with suitable variable conversion. The name of integrand subprogram must be declared in EXTERNAL statement in the main program.

2. To improve accuracy, it is recommended to formulate the problem so that abnormal points should be situated at ends of the integration interval and, if possible, at origin of the integral variable. Refer to the explanation of the example.

3. As described in the directions of the variable ILL, the obtained integral value is not always invalid even if ILL≠0 (except ILL≧10000) in this routine. For instance, the integral value is correct though ILL=1000 when algebraic singularity is detected and handled correctly. If the obtained integral value is not sure, it is recommended to use two routines to compare their results.

2/6

Bibliography

1) D. K. Kahaner; "Comparison of Numerical Quadrature Formulas," J. R. Rice, ed. :Mathematical Software, Academic Press, pp. 229-259 (1971).

2) Ichizo Ninomiya; "Improvement of adaptive Newton-Cotes quadrature rule" Information processing, Vol. 21, No. 5, and pp. 504-512 (1980).

3) I. Ninomiya; "Improvement of Adaptive Newton-Cotes Quadrature Methods," JIP, Vol. 3, No. 3, pp. 162-170, (1980).

(1987. 06. 03) (1987. 08. 08) (1987. 08. 21)

AQOSCS/D (Finite Fourier Integral)

Finite Fourier Integral

Programmed by	Takemitsu Hasegawa, March 1983
Format	Subroutine Language: FORTRAN; Size: 874 and 878 lines respectively

(1) Outline

AQOSCS/D obtains the value of the finite Fourier integrals

$$Ic=\int_a^b f(x)\cos(2\pi\omega x)dx \ , \qquad Is=\int_a^b f(x)\sin(2\pi\omega x)dx$$

for a given function f(x) at the precision of the convergence criterion ε.

AQOSCS(D) is for single (double) precision.

(2) Directions

CALL AQOSCS/D(A, B, OMEGA, FUN, KEY, EPSA, EPSR, NMIN, NMAX, SC, SS, N, ERR, ICON)

Argument	Type and kind (*1).	Attribute	Content
A	Real type	Input	Lower limit of integral domain.
B	Real type	Input	Upper limit of integral domain. A<B
OMEGA	Real type	Input	Frequency ω.

FUN	Function subprogram of real number type.	Input	Given function f(x). A function as an actual argument for this function should be prepared as a single-variable function subprogram with integration variables only.
KEY	Integer type	Input	If KEY=0, only Ic is calculated. If KEY=1, only Is is calculated. If KEY=2, Ic and Is are calculated at the same time. $0 \leq KEY \leq 2$
EPSA EPSR	Real type	Input	Prescribed absolute error εa (EPSA) and relative error εr (EPSR) for approximate integration values SC and SS. $EPSA \geq 0$.
NMIN NMAX	Integer type	Input	Lower and upper limits (NMIN and NMAX) of the number of evaluations of the function FUN. The adequate values are NMIN = 9 and NMAX = several hundreds. If $NMAX \geq 513$ is specified, we set NMAX=513 (single precision). If $NMAX \geq 2049$ is specified, we set NMAX=2049 (double precision). $0 < NMIN < NMAX$.
SC SS	Real type	Output	SC is the approximate value of Ic. SS is the approximate value of Is.
N	Integer type	Output	Total number of evaluations of the function FUN.
ERR	Real type	Output	Estimated absolute value for SC and SS.

ICON	Integer type	Output	ICON=0: Normal termination. If ICON=9999; (b-a) $\|\omega\|<0.01$, usual integration method is used. The result is correct. ICON=10000: When the accuracy of the approximate value of integration reaches the level of a rounding error. ICON=20000: When the integration does not converge even if the number of evaluations of a function reaches NMAX. ICON=30000: Parameter error.

*1 For double precision subroutines, all real types should be set double precision real types.

(3) Calculation method

The function f(x) is expanded in the Chebyshev polynomial and termwisely integrated. (1) The number of expansion terms is more slowly increased than doubling until the required precision ε is satisfied. (2) Expansion coefficients are calculated by using the fast Fourier transform. (3) The value of termwise integration is stably and efficiently calculated by using the 3-term recurrence formula stably and efficiently. These three features enable efficient automatic integration.

(4) Example

$$\int_0^1 exp(\alpha x)\cos(2\pi\omega x)dx \ , \qquad \int_0^1 exp(\alpha x)\sin(2\pi\omega x)dx$$

are calculated on the assumption that ω = 8, 32, 128 while the values of parameter α are 4 and 8.

Assume that $\varepsilon r = \varepsilon a = 1.E-3$.

```
C       EXAMPLE FOR AQOSCS
        COMMON ALPHA
        EXTERNAL FUN
        A=0.0
        B=1.0
        KEY=2
        EPSA=1.E-3
        EPSR=EPSA
        NMIN=9
        NMAX=100
        ALPHA=2.0
        WRITE(6,1)
      1 FORMAT(1HO,'TEST FOR AQOSCS'//1H ,'ALPHA  OMEGA',13X,
     *'COS',13X,'SIN',4X,'N',7X,'ERR  ICON')
        DO 10 I=1,2
        ALPHA=ALPHA+ALPHA
        DO 20 J=1,3
        OMEGA=8.0*4.0**(J-1)
        CALL AQOSCS(A,B,OMEGA,FUN,KEY,EPSA,EPSR,NMIN,NMAX,
     *SC,SS,N,ERR,ICON)
        WRITE(6,2) ALPHA,OMEGA,SC,SS,N,ERR,ICON
      2 FORMAT(1H ,F5.1,F7.1,2E16.7,I5,E10.2,I6)
     20 CONTINUE
     10 CONTINUE
        STOP
        END
        FUNCTION FUN(X)
        COMMON ALPHA
        FUN=EXP(ALPHA*X)
        RETURN
        END
```

If the integrand function contains a parameter (α in this example) as in this example, it is put in the COMMON region to communicate with the main program.

(5) Note

When the function f(x) diverges or has a sharp peak at the point x=p in the integration interval [a, ∞), this method should be used to approximate the integral over $[p+\delta, \infty)$ ($\delta>0$), and another method must be used for the integration of the interval $[a, a+\delta]$.

2. If both cosine and sine integral Ic, Is are required for thc same f(x), the calculation of the function f(x) is commonly used. Therefore, there is an advantage that both approximate values can be obtained by one of the function calculations of the two Is.

Bibliography

1) Takemitsu Hasegawa and Tatsuo Torii; "Semi-Infinite Oscillatory Integral Based on Chebyshev Series Expansion," Preprints of Working Group for Numerical Analysis, IPSJ 10-3 (1984).

(1987.08.11)

DEFINS/D and IMTDES/D/Q (Automatic numerical quadrature by double exponential formulas —— finite interval)

Automatic numerical Quadrature by Double Exponential Formulas ——Finite Interval——

Programmed by	Yasuyo Hatano; March 1977
Format	Subroutine language; FORTRAN Size; 264, 265, 136, and 137 lines respectively

(1) Outline

DEFINS/D and IMTDES/D/Q are automatic numerical integration routines, each of which calculates definite integral $\int_a^b f(x)dx$ with the precision within absolute error ε, using Takahashi-Mori's double exponential formula[1)2)3)], when integrand $f(x)$, lower limit a, upper limit b, and the required precision ε are given. Especially, it can obtain a high precision result by a small amount of calculation even if there are singular points of $x^{-\alpha}(0<\alpha<1)$ type at end points in the integral interval. Note, however, that $f(x)$ is assumed to be analytical except at end points.

(2) Directions

CALL DEFINS/D(A, B, F, S, EPS, N, ILL)

CALL IMTDES/D/Q(A, B, F, S, EPS, N, ILL)

Argument	Type and kind (1*)	Attribute	Content
A, B	Real type	Input	Upper and lower limits of a definite integral. A≠B
F	Real type Function subprogram	Input	Name of integrand function. The user should prepare a corresponding function subprogram having only one integration variable as an argument.
S	Real type	Output	The value of a definite integral is generated. If ILL is neither 0 nor 30000, an approximate value obtained last is generated.

Argument	Type and kind (1*)	Attribute	Content
EPS	Real type	Input	Positive number (ε) indicating required precision. A standard value for single precision is about 10^{-5} and that for double precision is about 10^{-10}.
N	Integer type	Output	Number of actual evaluations of function.
ILL	Integer type	Output	The situation of calculation in the routine is indicated. DEFINS/D: This argument is set to 0 at first in the routine, and the predetermined value is added to it each time the following conditions are met: (1) The required precision is automatically lowered because the function value increases rapidly near the lower limit of the interval: 1 (2) The event of (1) occurs near the upper limit of the interval: 2 (3) Convergence does not occur even if the maximum number of sample points available for the routine is used: 10000 (4) A restriction on the input argument is violated: 30000
ILL	Integer type	Output	IMTDES/D ILL=0: Normal termination. ILL=10000: Convergence does not occur even if the maximum number of sample points available for the routine is used. ILL=30000: No calculation is done because a restriction on the input argument is violated.

*1 For double precision routines, real types are all assumed to be double precision real types.

(3) Calculation method

Definite integral $I=\int_{-1}^{1} f(x)dx$ can be represented by $I=\int_{t_0}^{tn} f(\varphi(t))\varphi'(t)dt$ if it is subjected to x transformation $x=\varphi(t)$. I is then determined by applying the trapezoidal rule to this. The transformation formula used for DEFINS/D [2] is

$$x=\tanh\left(\frac{\pi}{2}\sinh t\right), -1\leqq x\leqq 1, -\infty\leqq t\leqq\infty$$

and that for IMTDES/D [3] is

$$x=\tanh\left\{\frac{\pi}{2}\sinh\frac{\pi}{2}\left(\frac{1}{1-t}-\frac{1}{1+t}\right)\right\}, -1\leqq x,t\leqq 1$$

(4) Performance

Each of these subroutines features a less frequency of evaluation. It is efficient for such an integrand function that shows a smooth change or relatively gradual vibration. Especially for those which show singularity of $x^{-\alpha}(0<\alpha<1)$ at end points, it shows a remarkable efficiency which is not available for any other routines. In this case, however, it may find it difficult to obtain precision of 10 digits or more. It is not suitable for those which have peaks at the center of an interval or which have a discontinuity point (see Table 1 on page 210 of bibliography [5]).

The following table shows the results of experiment of Kahaner's 21 test problems [4]. The reliability here indicates a ratio at which the calculated value of a definite integral actually satisfies the required precision.

Required precision	10^{-3}		10^{-6}		10^{-9}	
Name of routine	Reliability	Average number of evaluations	Reliability	Average number of evaluations	Reliability	Average number of evaluations
DEFIND	95%	81	90%	114	86%	138
IMTDED	86	59	90	113	81	131

224

(5) Example

```
C.... EXAMPLE OF DEFINS....
      EXTERNAL FUN
      A=-1.0
      B=1.0
      EPS=1.0E-4
      CALL DEFINS(A,B,FUN,S,EPS,NF,ILL)
      WRITE(6,610) ILL,S,NF
   10 CONTINUE
      STOP
  610 FORMAT(1H ,10X,5HILL= ,I5,3X,2HS=,E22.14,3X,2HN=,I5)
      END

      FUNCTION FUN(X)
      P=(1.0+X)*(1.0-X)
      FUN=0.0
      IF(P.GT.0.0) FUN=1.0/SQRT(P)
      RETURN
      END
```

Bibliography

1) H. Takahashi and M. Mori; "Double Exponential Formulas for Numerical Integration," Bull. R. I. M. S., Kyoto Univ., 9, pp. 721-741 (1974)

2) Masatake Mori; "Curve and Curved Surface," p. 24, Kyoiku Shuppan (1974)

3) Masatake Mori; "INT Type Double Exponential Integration Formulas," Collection at the Society of Numerical Calculation Algorithm, Kyoto University Mathematical Analysis Laboratory (1976)

4) D. K. Kahaner; "Comparison of Numerical Quadrature Formulas", J. R. Rice, ed., Mathematical Software, Academic Press, pp. 229-259 (1971)

5) Ichizo Ninomiya and Yasuyo Hatano; "New Registration SSL Program," Nagoya University Computer Center News, Vol. 8, No. 3, pp. 209-263 (1977)

(1987. 05. 29) (1987. 08. 21)

GASNS/D,GLBNS/D,GSCNS/D,GCSNS/D,GLGNS/D,GSLNS/D,GSHNS,/D

(Gaussian quadrature)

Gaussian Quadrature

Programmed by	Ichizo Ninomiya and Yasuyo Hatano: January 1984
Format	Subroutine language; FORTRAN Size; 50, 51, 28, 29, 27, 27, 600, 600, 322, 322, 58, 60, 68, and 69 lines respectively

(1) Outline

Each of these subroutines calculates a one-dimensional integral using the Gaussian quadrature rules.

GASNS or GASND calculates a finite interval integral

$$Y=\int_a^b f(x)dx$$

using the Gauss-Legendre rule.

GLBNS or GLBND calculates a finite interval integral

$$Y=\int_a^b f(x)dx$$

using the Gauss-Lobatto rule.

GSCNS or GSCND calculates a finite interval integral

$$Y=\int_a^b f(x)dx/\sqrt{(x-a)(b-x)}$$

using the Gauss-Chebyshev rule.

GCSNS or GCSND calculates a finite interval integral

$$Y=\int_a^b f(x)\cos\frac{\pi(2x-a-b)}{2(b-a)}dx$$

using the Gauss-cosine rule.

GLGNS or GLGND calculates a finite interval integral

$$Y=\int_a^b f(x)\log\frac{x-a}{b-a}dx$$

using the Gauss-logarith rule.

GSLNS or GSLND calculates a semi-infinite integral

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

using the Gauss-Laguerre rule.

GSHNS or GSHND calculates an infinite integral

$$Y=\int_{-\infty}^{\infty} e^{-x^2}f(x)dx$$

using the Gauss-Hermite rule.

(2) Directions

```
CALL GASNS/D(A, B, FUN, N, Y, ICON)
CALL GLBNS/D(A, B, FUN, N, Y, ICON)
CALL GSCNS/D(A, B, FUN, N, Y, ICON)
CALL GCSNS/D(A, B, FUN, N, Y, ICON)
CALL GLGNS/D(A, B, FUN, N, Y, ICON)
CALL GSLNS/D(FUN, N, Y, ICON)
CALL GSHNS/D(FUN, N, Y, ICON)
```

Argument	Type and kind (*1)	Attribute	Content
A	Real type	Input	Lower limit of definite integral.
B	Real type	Input	Upper limit of definite integral.
FUN	Real type function subprogram	Input	Name of integrand f(x). For the function as an actual argument of this function, the user should prepare a function subprogram with only the integration variable as an argument.
N	Integer type	Input	Number of sample points. GASNS, GLBNS, GSCNS, GSLNS, GSHNS···1≦N≦20. GASND, GLBND, GSCND···1≦N≦50. GSLND, GSHND···1≦N≦38. GCSNS, GCSND···1≦N≦33. GLGNS, GLGND···1≦N≦17.
Y	Real type	Output	The value of the definite integral is output.
ICON	Integer type	Output	ICON=0: Normal termination. ICON=30000: The restriction on input argument N was not observed.

*1 For double precision subroutines, all real types should be changed to double precision real types.

(3) Example

```
C***** EXAMPLE FOR GASNS ****
      EXTERNAL FUN
      A=0.0
      B=1.0
      E=EXACT(A,B)
      DO 1000 N=1,20
      CALL CLOCKM(ITA)
      CALL GASNS(A,B,FUN,N,Y,ICON)
      CALL CLOCKM(ITB)
      ERR=E-Y
      IT=ITB-ITA
      WRITE(6,610) N,ICON,Y,ERR,IT
```

```
 610 FORMAT(1H ,'GASNS',I3,' POINTS ICON =',I6,E25.15,
    *E10.3,I5)
1000 CONTINUE
2000 STOP
     END

     FUNCTION FUN(X)
     FUN=EXP(-X)
     RETURN
     END
     FUNCTION EXACT(A,B)
     EXACT=-EXP(-B)+EXP(-A)
     RETURN
     END
```

(1987.07.25)

HINFAS/D/Q and HINFES/D (Numerical Quadrature by Double Exponential Formulas —— Semiinfinite Interval ——)

Numerical Quadrature by Double Exponential Formulas ——Semiinfinite Interval——

Programmed by	Yasuyo Hatano, May 1977
Format	Subroutine Language: FORTRAN; Size: 261, 262, 261, and 262 lines respectively.

(1) Outline

HINFAS/D/Q and HINFES/D are automatic integration routines for calculating the definite integral $\int_0^\infty f(x)dx$ over a semiinfinite interval with an absolute error of ε or less using the double exponential function type integration formula [1] of Takahashi and Mori when the integrand function $f(x)$ and the required precision ε are given. Especially HINFES(D) uses a formula to be represented in the form of $f(x)=g(x)e^{-x}$ using a stable function $g(x)$. Even if $f(x)$ is slow in conversion to 0 with $x\to\infty$, a high-precision solution can be obtained at $x=0$ with a singularity of about $x^{-\alpha}(0<\alpha<1)$.

(2) Directions

CALL HINFAS/D/Q (F, S, EPS, N, ILL)

CALL HINFES/D (F, S, EPS, N, ILL)

Argument	Type and kind (*1)	Attribute	Content
F	Real type Function subprogram	Input	Name of an integrand function. The user should prepare a subprogram for this integrand function as the one that has only one integration variable as an argument.
S	Real type	Output	The value of a definite integral is output. If ILL is neither 0 nor 30000, the last obtained approximation value is output.

Argument	Type and kind (*1)	Attribute	Content
EPS	Real type	Input	Positive number (ε) that represents a required precision. 10^{-5} is adequate for single precision, and 10^{-8} is adequate for double precision. Retained.
N	Integer type	Output	Actual number of evaluations of a function.
ILL	Integer type	Output	Indicates a calculation state in the routine. This argument is first set to 0 in the routine. Each time one of the following states is activated, a certain value is added correspondingly. (1) 1 when required precision is automatically lowered because the function value increases sharply with $x \rightarrow 0$. (2) 2 when required precision is automatically lowered because the function value is slow in convergence to 0 with $x \rightarrow \infty$. (3) 10000 if the function does not converge with a maximum allowable number of samples of the routine are used. (4) 30000 when EPS<0 is specified.

*1 For double precision routines, all real types should be double precision real types.

(3) Calculation method

Trapezoidal rules should be applied to the integration variable x that is converted as described below in the definite integral $\int_0^\infty f(x)dx$.

$$x=exp\left(\frac{\pi}{2}sinht\right),\ -\infty \leq t \leq \infty$$

is used for HINFAS(D)[1], and

$$x=exp\left\{\frac{\pi}{2}(t-exp(-t))\right\},\ -\infty \leq t \leq \infty$$

is used for HINFBS(D)[1].

(4) Performance

A considerably good result is obtained even if the function value is slow in convergence to 0 with $x \to \infty$ or an integral cannot be effectively obtained with the Gauss-Laguerre formula. However, it is difficult to obtain a precision of 10 digits or more (refer to Table 1 (page 210) of the bibliography[2]).

Bibliography

1) Masatake Mori; "Curve and Curved Surface," p. 24, Kyoiku Shuppan (1974).

2) Ichizo Ninomiya and Yasuyo Hatano; "Newly Registered Program SSL," Nagoya University Computer Center News Vol. 8, No. 3, pp. 209-263 (1977).

(1987. 05. 13) (1987. 08. 21)

INFINS/D (Numerical Quadrature by Double Exponential Formulas —— Infinite Interval——)

Numerical Quadrature by Double Exponential Formulas ——Infinite Interval——

Programmed by	Yasuyo Hatano, April 1977
Format	Subroutine Language: FORTRAN; Size: 250 and 251 lines respectively.

(1) Outline

INFINS/D is an automatic integration routine for calculating the definite integral $\int_{-\infty}^{\infty} f(x)dx$ over an indefinite interval with an absolute error of ε or less using the double exponential function type integration formula [1] of Takahashi and Mori when the integrand function $f(x)$ and the required precision ε are given. A high-precision solution can be obtained even when $f(x)$ is slow in convergence to 0 with $x \to \pm\infty$.

(2) Directions

CALL INFINS/D (F, S, EPS, N, ILL)

Argument	Type and kind (*1)	Attribute	Content
F	Real type Function subprogram	Input	Name of an integrand function. The user should prepare a function subprogram for this integrand function as the one that has only one integration variable as an argument.
S	Real type	Output	The value of a definite integral is output. If ILL is neither 0 nor 30000, the last obtained approximation value is output.
EPS	Real type	Input	Positive number (ε) that represents a required precision. 10^{-5} is adequate for single precision, and 10^{-8} is adequate for double precision.
N	Integer type	Output	Actual number of evaluations of a function.

Argument	Type and kind (*1)	Attribute	Content
ILL	Integer type	Output	Indicates a calculation state in the routine. This argument is first set to 0 in the routine. Each time one of the following states is activated, a certain value is added correspondingly. (1) 1 when required precision is automatically lowered because the function value is slow in convergence to 0 with $x \to -\infty$. (2) 2 when the state of (1) is activated with $x \to +\infty$. (3) When the function does not converge even if a maximum allowable number of sample points of the routine are used. 10000 (4) 30000 when EPS<0 is specified.

*1 For double precision routines, all real types should be double precision real types.

(3) Calculation method

Trapezoidal rules should be applied to the integration variable x that is converted as described below in the definite integral $\int_{-\infty}^{\infty} f(x)dx$ [1].

$$x=\sinh\left(\frac{\pi}{2}\sinh t\right),\ -\infty \leqq x, t \leqq \infty$$

(4) Note

A considerably good result is obtained even if the function value is slow in convergence to 0 with $x \to \pm\infty$ or an integral is not effectively obtained with the Gauss-Hermite formula. However, this routine is inadequate for the function that has a peak point or oscillates violently near the origin (refer to Table 1 (page 210) in bibliography [2]).

Bibliography

1) Masatake Mori; "Curve and Curved Surface," p. 24, Kyoiku Shuppan (1974).

2) Ichizo Ninomiya and Yasuyo Hatano; "Newly Registered Program SSL," Nagoya University Computer

Center News, Vol. 8, No. 3, pp. 209-263 (1977).

(1987. 08. 11) (1987. 08. 21)

MQFSRS/D (Multiple quadrature by fully symmetric rules)

Multiple Quadrature by Fully Symmetric Rules

Programmed by	Ichizo Ninomiya: April 1981
Format	Subroutine language; FORTRAN Size; 250 lines each

(1) Outline

When the following multiple integral for n-dimensional ($1 \leqq n \leqq 50$) hyperrectangle ($a_i \leqq x_i \leqq b_i$, $i=1, \cdots, n$) is given;

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

MQFSRS or MQFSRD calculates its values using the 3rd, 5th, 7th, and 9th fully symmetric rules. MQFSRS is for single precision and MQFSRD is for double precision.

(2) Directions

CALL MQFSRS/D(N, A, B, FUN, MBT, ND, S, ILL)

Argument	Type and kind (*1)	Attribute	Content
N	Integer type	Input	Multiplicity of integral. $1 \leqq N \leqq 50$
A	Real type One-dimensional array	Input	Lower limit of integral domain.
B	Real type One-dimensional array	Input	Upper limit of integral domain.
FUN	Real type Function subprogram	Input	Integrand function. The user must prepare the actual argument as a function subprogram that uses only integration variables as arguments.

Argument	Type and kind (*1)	Attribute	Content
MET	Integer type One-dimensional array	Input	One-dimensional array with two elements. MET(1) shows the order of the fully symmetric rule and should be one of 3, 5, 7, and 9. MET(2) is used only for 7th or 9th rule. The value is 1 or 2, having the following meanings: MET(2)=1: Sample points near edges of a region are used. MET(2)=2: Sample points near the center of a region are used.
ND	Integer type One-dimensional array	Input	Number of equipartitions of a side in the direction of each coordinate axis. $1 \leq ND(I), I=1, \cdots, N$
S	Real type	Output	Approximate value of integral.
ILL	Integer type	Output	ILL=0: Normal termination. ILL=K: $ND(K) \leq 0$. ILL=30000: N and MET violated the limits.

*1 For double precision subroutines, all real types should be changed to double precision real types.

(3) Calculation method

To make explanation simple, suppose a region is the product $[-1,1]^n$ of the interval $[-1,1]$. The fully symmetric rule is a nonproduct-type multiple numerical integration rule, which uses sample points consisting of a group of small number of non-zero coordinate elements that are arranged to be fully symmetric (about exchange and inversion coordinate axes) in respect to the origin. The weight factor by which the function value at each data point is multiplied is determined so that the integration rule becomes accurate for a polynomial with the highest order. This routine uses the third, fifth, seventh, and ninth integration rules. The fully symmetric rules are not so good in precision, but the increase of the number of sample points along with an increase of dimension n is comparatively moderate. It can thus be applied to rather high dimensions. The number of sample points F is given by dimension n as follows:

3rd: $F = 2n + 1$ 7th: $F = 4/3n(n^2 + 2) + 1$

5th: $F = 2n^2 + 1$ 9th: $F = 2/3n(n - 2)(n^2 - 1) + 4n(2n - 1) + 1$

The table below lists the values of F when N ranges from 3 to 20.

N	3	5	7	9
3	7	19	45	77
4	9	33	97	193
5	11	51	181	421
6	13	73	305	825
7	15	99	477	1485
8	17	129	705	2497
9	19	163	997	3973
10	21	201	1361	6041
11	23	243	1805	8845
12	25	289	2337	12545
13	27	339	2965	17317
14	29	393	3697	23353
15	31	451	4541	30861
16	33	513	5505	40065
17	35	579	6597	51205
18	37	649	7825	64537
19	39	723	9197	80333
20	41	801	10721	98881

The weight factor depends on the number of dimensions n, and vibrates harder as n increases (this tendency is eminent as the order becomes large). Therefore, this routine is not expected to have good precision with fully symmetric rules. It is suitable for integration of higher dimensional, smooth functions with low accuracy.

(4) Example

We calculate the integral

$$\frac{1}{64}\int_{-1}^{1}\int_{-1}^{1}\cdots\cdots\int_{-1}^{1}\cos\left(3(1-x_1)x_2x_3x_4x_5x_6+\frac{1}{2}\right)dx_1dx_2\cdots dx_6$$

in six-dimensional area $[-1,1]^6$.

Fully symmetric rules are applied to 64 small areas produced by halving the area in the direction of each coordinate axis. The program and its output are as follows.

```
C       EXAMPLE FOR MQFSRS
        DIMENSION A(6),B(6),MET(2),ND(6)
        EXTERNAL FUN
        N=6
        MET(2)=1
        EXACT=0.8585247
        DO 10 I=1,N
        ND(I)=2
        A(I)=-1.0
   10   B(I)=1.0
        WRITE(6,600)
  600   FORMAT(1HO,'TEST FOR MQFSRS'//
       *1H ,'N ORDER',8X,'EXACT',7X,'RESURT'
       *,3X,'ABS ERR',3X,'REL ERR'/)
        DO 20 J=3,9,2
```

```
      MET(1)=J
      CALL MQFSRS(N,A,B,FUN,MET,ND,S,ILL)
      AER=S-EXACT
      RER=AER/EXACT
   20 WRITE(6,610) N,J,EXACT,S,AER,RER
  610 FORMAT(1H ,I1,I6,2E13.5,2E10.2)
      STOP
      END

      FUNCTION FUN(X)
      DIMENSION X(6)
      FUN=COS((1.0-X(1))*X(2)*X(3)*X(4)*X(5)*X(6)
     **3.0+0.5)/64.0
      RETURN
      END
```

```
      TEST FOR MQFSRS

      N ORDER        EXACT       RESURT    ABS ERR    REL ERR


      6     3  0.85852E+00  0.86449E+00  0.60E-02  0.69E-02

      6     5  0.85852E+00  0.85897E+00  0.44E-03  0.52E-03

      6     7  0.85852E+00  0.85769E+00 -0.84E-03 -0.97E-03

      6     9  0.85852E+00  0.85800E+00 -0.52E-03 -0.61E-03
```

Bibliography

1) J. McNamee & F. Stenger; "Construction of Fully Symmetric Numerical Integration Formulas", Numer. Math., Bd, 10, pp.327-344 (1967).

(1987.05.25)

MQNCDS/D (Multiple Quadrature by Product of Newton-Cotes Rules (Data Input))

Multiple Quadrature by Product of Newton-Cotes Rules (Data Input)

Programmed by	Ichizo Ninomiya, April 1981
Format	Subroutine Language: FORTRAN; Size: 60 and 61 lines respectively

(1) Outline

MQNCDS/D obtains the value of an n dimensional multiple integral

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

using the product of Newton-Cotes rules when the value of an integrand function f is given as data on the equally spaced mesh points of an n dimensional $(1 \leqq n \leqq 10)$ hyperrectagular region $(a_i \leqq x_i \leqq b_i, i=1, \cdots, n)$.

MQNCDS(D) is for single (double) precision.

(2) Directions

CALL MQNCDS/D(F, N, NC, NP, H, S, ILL)

Argument	Type and kind (*1)	Attribute	Content
F	Real type N-dimensional array	Input	The values at the mesh points of an integrand function should be input. The value of each subscript in the array declaration of F should be exactly equal to the number of sample points in the corresponding direction of the coordinates of the hyperrectangle.
N	Integer type	Input	Multiplicity of an integral. $1 \leqq N \leqq 10$
NC	Integer type One-dimensional array	Input	Number of divisions in each direction of coordinates. $1 \leqq NC(I)$, I=1, ···, N
NP	Integer type One-dimensional array	Input	Number of sample points of Newton-Cotes rules used in each direction of coordinates. $1 \leqq NP(I) \leqq 11$, I=1, ···, N. NC(I)·(NP(I)-1) is the number of sample points in each direction of coordinate (see "Example").

Argument	Type and kind (*1)	Attribute	Content
H	Real type One-dimensional array	Input	Mesh interval in the each direction of coordinates.
S	Real type	Output	Approximate value of the integral.
ILL	Integer type	Output	ILL=0: Normal termination. ILL=K: The input argument corresponding to the K-th direction exceeded the limits. ILL=30000: N≦0 or N>10.

*1 For double precision subroutines, all real types should be changed to double precision real types.

(3) Example

A 2-dimensional square region $0 \leqq x_1 \leqq 1, 0 \leqq x_2 \leqq 1$ is divided into 20 parts in each axial direction, and the value of a function $f(x_1, x_2) = e^{x_2} \sin x_1$ is given to each mesh point as data. Then, the integral

$$\int_0^1 \int_0^1 f(x_1, x_2) dx_1 dx_2$$

is found by this subroutine.

The subinterval in each axial direction is divided into two or four and Newton-Cotes 11-point or 6-point rules are used correspondingly. The program and its output are as follows:

```
00010 C *** EXAMPLE FOR MQNCDS ***
00020       GENERIC
00040       DIMENSION NC(2),NP(2),H(2),A(21,21)
00050       H(1)=0.05
00060       H(2)=0.05
00065       WRITE(6,620)
00070       DO 20 I=1,21
00080       F=SIN(FLOAT(I-1)/20.E0)
00090       DO 20 J=1,21
00095    20 A(I,J)=EXP(FLOAT(J-1)/20.E0)*F
00098       T=(1.-COS(1.E0))*(EXP(1.E0)-1.)
00100       DO 30 I1=2,4,2
00110       NC(1)=I1
00120       NP(1)=20/I1+1
00130       DO 30 I2=2,4,2
00140       NC(2)=I2
00150       NP(2)=20/I2+1
00160       CALL MQNCDS(A,2,NC,NP,H,S,ILL)
00180       E=S-T
```

```
00187    620 FORMAT(9X,4H ILL,7X,1HS,14X,1HT,14X,1HE)
00190        WRITE(6,610) I1,I2,ILL,S,T,E
00200    610 FORMAT(3I4,3E15.5)
00210     30 CONTINUE
00220        STOP
00230        END
```

```
      ILL       S             T             E
  2   2   0   0.78989E+00 0.78989E+00 -0.23842E-05
  2   4   0   0.78989E+00 0.78989E+00 -0.30398E-05
  4   2   0   0.78989E+00 0.78989E+00 -0.30398E-05
  4   4   0   0.78989E+00 0.78989E+00 -0.30398E-05
```

(4) Note

The formulas of up to 11 sample points are prepared. However, it is generally better to divide each side into several equal parts (by increasing the value of NC) and use the formulas with relatively small number of sample points, rather than the formulas with unnecessarily large number of sample points.

(1987.08.08)

MQPRRS/D (Multiple Quadrature by Product Rules)

Multiple Quadrature by Product Rules

Programmed by	Ichizo Ninomiya, March 1979
Format	Subroutine Language: FORTRAN; Size: 79 and 80 lines respectively.

(1) Outline

MQPRRS/D calculates the value of the n dimensional multiple definite integral

$$\int_{a_1}^{b_1} w_1(x_1)dx_1 \int_{a_2}^{b_2} w_2(x_2)dx_2 \cdots \int_{a_n}^{b_n} w_n(x_n)dx_n \cdot f(x_1, x_2, \cdots, x_n)$$

using product formulas of various one-dimensional rules when dimensions $n(1 \leqq n \leqq 10)$, lower limits $a_1, \cdots, a_2, \cdots, a_n$, upper limits b_1, b_2, $\cdots$, b_n, and an integrand function $f(x_1, x_2, \cdots, x_n)$ are given.

The following are available as one-dimensional integration rules.

(1) Newton-Cotes rule $(w(x)=1)$

(2) Gauss-Lobatto rule $(w(x)=1)$

(3) Gauss-Legendre rule $(w(x)=1)$

(4) Gauss-Laguerre rule $(w(x)=e^{-x},\ a=0,\ b=\infty)$

(5) Gauss-Hermite rule $(w(x)=e^{-x^2},\ a=-\infty,\ b=\infty)$

(2) Directions

CALL MQPRRS/D(N, A, B, FUN, MET, NPT, NDV, S, P, W, ISW, ILL)

Argument	Type and kind (*1)	Attribute	Content
N	Integer type	Input	Multiplicity of an integral. $1 \leqq N \leqq 10$
A	Real type One-dimensional array	Input	Indicates the lower limit of an integral domain. Elements corresponding to infinite integral rules are arbitrary.

Argument	Type and kind (*1)	Attribute	Content
B	Real type One-dimensional array	Input	Indicates the upper limit of an integral domain. Elements corresponding to infinite integral rules are arbitrary.
FUN	Real type Function subprogram	Input	Integrand function. A function as an actual argument for this integrand function should be prepared as a function subprogram with integration variables only.
MET	Integer type One-dimensional array	Input	Represents the integration method used in each direction of the coordinate axes. MET=1 Newton-Cotes rule MET=2 Gauss-Lobatto rule MET=3 Gauss-Legendre rule MET=4 Gauss-Laguerre rule MET=5 Gauss-Hermite rule
NPT	Integer type One-dimensional array	Input	Represents the number of sample points of the integration method used in each direction of the coordinate axes. $1 \leq NPT(I) \leq 20$ However, assume $1 \leq NPT(I) \leq 11$ for Newton-Cotes rule.
NDV	Integer type One-dimensional array	Input	Number of equipartitions of a side in each direction of the coordinate axes. $1 \leq NDV(I)$ Elements corresponding to infinite integral rules are arbitrary.
S	Real type	Output	Approximate value of the integral.
P	Real type One-dimensional array	Work area	One-dimensional array of a size larger than the total number of data points (number of data points times number of divisions) in each direction of the coordinate axes.
W	Real type One-dimensional array	Work area	Work area of the same size as P.
ISW	Integer type	Input	If ISW=0, sample points and weights are calculated. If ISW≠0, calculation of sample points and weights is omitted, and those of previous call are reused.
ILL	Integer type	Output	ILL=0: Normal termination. ILL=30000: Limits on N are exceeded. ILL=K: The argument concerning the direction of the K-th axis exceeded the limits.

*1 For double precision subroutines, all real types should be changed to double precision real types.

(3) Example

The value of the triple integral

$$\int_0^{\infty} e^{-x}dx \int_{-1}^{1} dy \int_0^{1} dz \cdot x^2 \sin(q+y)/(1+z^2)$$

is obtained changing the value of the auxiliary variable q. Gauss-Laguerre 10-point rules are used in the x direction, Newton-Cotes 3-point rules (Simpson rule) are used in the y direction, and Gauss-Legendre 5-point rules are used in the z direction.

Further, the interval [-1, 1] is equally divided into 10 parts in the y direction, and the interval [0, 1] is equally divided into two parts in the z direction.

```
   C      MAIN PROGRAM
 1        DIMENSION A(3),B(3),MET(3),NPT(3),NDV(3),P(100),
         *W(100)
 2        EXTERNAL FUN
 3        COMMON Q
 4        N=3
 5        A(2)=-1.0
 6        B(2)=1.0
 7        A(3)=0.0
 8        B(3)=1.0
 9        MET(1)=4
10        MET(2)=1
11        MET(3)=2
12        NPT(1)=10
13        NPT(2)=3
14        NPT(3)=5
15        NDV(2)=10
16        NDV(3)=2
17        DO 10 IQ=1,50
18        Q=FLOAT(IQ)/10.0
19        ISW=1Q-1
20        CALL MQPRRS(N,A,B,FUN,MET,NPT,NDV,S,P,W,ISW,ICON)
21     10 WRITE(6,610) Q,S,ICON
22    610 FORMAT(1H ,3HQ =,F5.1,2X,3HS =,E13.5,2X,6HICON =,
        *       I6)
23        STOP
24        END

 1        FUNCTION FUN(X)
 2        COMMON Q
 3        DIMENSION X(3)
 4        FUN=X(1)**2*SIN(X(2)+Q)/(X(3)**2+1.0)
 5        RETURN
 6        END
```

If the integrand function contains an auxiliary variable as in this example (Q in this example), it is put in the common area for communication between the main program and integrand function subprogram. If the same integration formula is repeatedly used in the same region as in this example, it is better to use the ISW function and omit the calculation of sample points

weights after the first call.

(4) Note

1. The name of the integrand function subprogram must be defined in the EXTERNAL declaration in the calling program.

2. Because A, B, and NDV are arbitrary in the Gauss-Laguerre and Gauss-Hermite rules, the corresponding elements need not be defined.

3. Up to 11 sample points are prepared for Newton-Cotes rule, and up to 20 sample points are prepared for other rules. However, it is generally better to divide the interval into a number of equal parts and use in each subinterval the formula with relatively small number of sample points rather than the formula with unnecessarily large number of sample points.

4. This subroutine calls the following slave subroutines to obtain sample points and weights.

(1) Newton-Cotes rule (TNCOTS/D)

(2) Gauss-Lobatto rule (TGLOBS/D)

(3) Gauss-Legendre rule (TGLEGS/D)

(4) Gauss-Laguerre rule (TGLAGS/D)

(5) Gauss-Hermite rule (TGHERS/D)

(1987.05.07)

QDAPBS/D (A Quadrature of Interpolatory Type Increasing the Sample Points with Arithmetic Progression)

A Quadrature of Interpolatory Type Increasing the Sample Points with Arithmetic Progression

Programmed by	Takemitsu Hasegawa, April 1977
Format	Subroutine Language: FORTRAN; Size: 142 and 302 lines respectively.

(1) Outline

QDAPBS/D is an automatic integration routine for calculating the definite integral $\int_a^b f(x)dx$ with the highest precision that can be obtained with a computer when the integrand function $f(x)$ and the lower and upper ends a and b of the integration interval are given. In this routine, the number of sample points is increased by arithmetical progression (in units of eight points). Therefore, it rarely wastes the samples, and is efficient. Also, it is high in precision for smooth integrand functions.

(2) Directions

CALL QDAPBS/D(A, B, F, S, EPS, N, ILL)

Argument	Type and kind (*1)	Attribute	Content
A	Real type	Input	Lower end of integration interval.
B	Real type	Input	Upper end of integration interval.
F	Real type Function subprogram	Input	Name of an integrand function. User should prepare a function subprogram with one integral variable only.
S	Real type	Output	The approximate value of a definite integral is output. If ILL=10000, the last obtained approximate value is output.
EPS	Real type	Output	Estimation of errors of the approximate integral S.
N	Integer type	Input/output	The lower limit of the number of sample points is assumed to be the input. The number of samples actually used is output. N=16(QDAPBS) or N=32(QDAPBD) should be used as the input.

Argument	Type and kind (*1)	Attribute	Content
ILL	Integer type	Output	ILL=0: Normal termination. ILL=10000: When the approximate integral does not converge even if 200 sample points (QDAPBS) or 512 sample points (QDAPBD) are used. ILL=30000: B≦A.

*1 For double precision subroutines, all real types should be changed to double precision real types.

(3) Calculation method

The integration interval [A, B] is converted into [-1, 1], and sequence of interpolation formulas are prepared and integrated by progressively adding a set of eight sample points at a time that is a subset of a sequence of Chebysheve distribution in the interval [-1, 1].

Bibliography

1) Ichizo Ninomiya and Yasuyo Hatano; "Newly Registered Program SSL," Nagoya University Computer Center News, Vol. 8, No. 3, pp. 209-263 (1977).

2) Tatsuo Torii, Takemitsu Hasegawa, and Ichizo Ninomiya; "Interpolatory Automatic Integration Method That Increases the Number of Samples by Arithmetic Progression," Information Processing, Vol. 19, No. 3, pp. 248-255 (1978).

(1987. 05. 07) (1987. 08. 08)

ROMBGS/D (Romberg Quadrature)

Romberg Quadrature

Programmed by	Ichizo Ninomiya, April 1977
Format	Subroutine Language: FORTRAN; Size: 30 and 31 lines respectively

(1) Outline

ROMBGS/D is an automatic integration routine based on classic Romberg quadrature. It calculates the definite integral $\int_a^b f(x)dx$ with the precision ε, when the lower and upper ends a and b of the integration interval, integrand function $f(x)$, and convergence criterion ε are given.

(2) Directions

CALL ROMBGS/D(A, B, F, S, EPS, ILL)

Argument	Type and kind (*1)	Attribute	Content
A	Real type	Input	Lower end of an integration interval.
B	Real type	Input	Upper end of an integration interval.
F	Real type Function subprogram	Input	Name of an integrand function. A function as an actual argument for this integrand function should be prepared as a function subprogram with a single integration variable only.
S	Real type	Output	The value of a definite integral is entered. If ILL=1, the last obtained approximate value is contained.
EPS	Real type	Input	Convergence criterion. If the absolute value of the difference between two consecutive approximate values becomes smaller than EPS, it is assumed that convergence has been attained. EPS>0
ILL	Integer type	Output	ILL=0: Normal termination. ILL=30000: EPS≦0 ILL=1: When convergence is not attained even if the integration interval is divided into 8192 parts in ROMBGS, and 16384 parts in ROMBGD.

*1 For double precision subroutines, all real types should be changed to double precision real types.

(3) Example

Let

$$J_0(x)=\frac{1}{\pi}\int_0^{\pi}\cos(x\sin\theta)d\theta$$

be an example of integration of a function containing an auxiliary variable.

The following program obtains $J_0(x)$ changing the auxiliary variable x from 0.1 to 5.0 in steps of 0.1.

```
C       MAIN PROGRAM
        COMMON X
        EXTERNAL BES
        PI=3.141593
        X=0.0
        DO 1 J=1, 50
        X=X+0.1
        CALL ROMBGS(0.0,PI,BES,S,1.0E-6,ILL)
        B=S/PI
        :
      1 CONTINUE
        :
        END

C       FUNCTION SUBPROGRAM FOR INTEGRAND
        FUNCTION BES(THETA)
        COMMON X
        BES=COS(X*SIN(THETA))
        RETURN
        END
```

(4) Note

This routine is adequate only when the integrand function is well behaved. If the integrand function varies sharply, other routines such as adaptive automatic integration routines should be used.

(1987.08.11)

TNCOTS/D/Q,TGLEGS/D/Q,TGLAGS/D/Q,

TGCHBS/D/Q,TGHERS/D/Q,TGLOBS/D/Q

(Tables of weights and sample points for quadrature formulas)

Tables of Weights and Sample Points for Quadrature Formulas

Programmed by	Ichizo Ninomiya and Yasuyo Hatano : January 1984
Format	Subroutine language; FORTRAN Size; 31, 32, 32, 35, 36, 36, 36, 37, 37, 45, 46, 46, 41, 42, and 42 lines respectively

(1) Outline

Each of these subroutines calculates the tables of weights and sample points for a quadrature formula.

1. TNCOTS, TNCOTD, or TNCOTQ calculates the sample point xk and weight Wk for the Newton-Cotes rule

$$\int_{-1}^{1} f(x)dx=\sum_{k=1}^{n} W_k f(x_k)+E_n$$

2. TGLEGS, TGLEGD, or TGLEGQ calculates the sample point xk and weight Wk for the Gauss-Legendre rule

$$\int_{-1}^{1} f(x)dx=\sum_{k=1}^{n} W_k f(x_k)+E_n$$

3. TGLAGS, TGLAGD, or TGLAGQ calculates the sample point xk and weight Wk for the Gauss-Laguerre rule

$$\int_0^\infty e^{-x} f(x)dx = \sum_{k=1}^{n} W_k f(x_k) + E_n$$

4. TGCHBS, TGCHBD, or TGCHBQ calculates the sample point xk and weight Wk for the Gauss-Chebyshev rule

$$\int_{-1}^{1} f(x)dx/\sqrt{1-x^2} = \sum_{k=1}^{n} W_k f(x_k) + E_n$$

5. TGHERS, TGHERD, or TGHERQ calculates the sample point xk and weight Wk for the Gauss-Hermite rule

$$\int_{-\infty}^{\infty} e^{-x^2} f(x)dx = \sum_{k=1}^{n} W_k f(x_k) + E_n$$

6. TGLOBS or TGLOBD calculates the sample point xk and weight Wk for the Gauss-Lobatto rule

$$\int_{-1}^{1} f(x)dx = W_1 f(-1) + \sum_{k=2}^{n-1} W_k f(x_k) + W_n F(1) + E_n$$

(2) Directions

CALL TNCOTS/D/Q(N, X, W, EPS, ICON)

CALL TGLEGS/D/Q(N, X, W, EPS, ICON)

CALL TGLAGS/D/Q(N, X, W, EPS, ICON)

CALL TGCHBS/D/Q(N, X, W, EPS, ICON)

CALL TGHERS/D/Q(N, X, W, EPS, ICON)

CALL TGLOBS/D/Q(N, X, W, EPS, ICON)

Argument	Type and kind (*1)	Attribute	Content
N	Integer type	Input	Number of sample points n. TNCOTS/D/Q···2≦N≦11. TGLEGS/D/Q···2≦N≦37. TGLAGS/D/Q···1≦N≦39. TGCHBS/D/Q···1≦N≦50. TGHERS/D/Q···1≦N≦40. TGLOBS/D/Q···1≦N≦20.
X	Real type One-dimensional array	Output	Size N. The table of sample points xk is output (k=1, 2, ···, n).
W	Real type One-dimensional array	Output	Size N. The table of weights wk is output (k=1, 2, ···, n).
EPS	Real type	Input	Convergence criterion in the Newton method for calculating sample points xk (for instance, the zero point of Legendre polynomial Pn(x) for TGLEGS).

Argument	Type and kind (*1)	Attribute	Content
ICON	Integer type	Output	ICON=0: Normal termination. ICON=10000: EPS was too small, so it was raised as follows: Single precision ••• 10^{-6} Double precision ••• 10^{-15} Quadruple precision ••• 10^{-32}. ICON=30000: The restriction on input argument N was not observed.

*1 For double or quadruple precision subroutines, all real types should be changed to double or quadruple real types.

Bibliography

1) Yamauchi, Uno, and Hitotsumatsu; "Numerical analysis method III for computers", Baifukan, p.279 (1971).

(1987.07.25)

TRAPZS/D (Numerical Quadrature by Trapezoidal Rule —— Infinite Interval ——)

Numerical Quadrature by Trapezoidal Rule ——Infinite Interval——

Programmed by	Yasuyo Hatano, April 1977
Format	Subroutine Language: FORTRAN; Size: 81 and 82 lines respectively.

(1) Outline

TRAPZS/D is an automatic integration routine for calculating the definite integral $\int_{-\infty}^{\infty} f(x)dx$ with an absolute error of ε or less according to trapezoidal rules when the integrand function $f(x)$ and the required precision ε are given. It is effective when $f(x)$ is fast in convergence to 0 with $x \to \pm\infty$.

(2) Directions

CALL TRAPZS/D (F, S, H, EPS, N, MAXF, ILL)

Argument	Type and kind (*1)	Attribute	Content
F	Real type Function subprogram	Input	Name of an integrand function. The user should prepare a function subprogram for this integrand function as the one that has only one integration variable as an argument.
S	Real type	Output	Definite integral values are output. If ILL is neither 0 nor 3000, the last obtained approximation value is output.
H	Real type	Input/output	An initial value of the step size is assumed to be the input. The input is decreased 50% by 50% during the calculation, and the value of the step size at the final stage is output. H>0
EPS	Real type	Input	Positive number (ε) that represents a required precision. 10^{-5} is minimum with single precision, and 10^{-15} is minimum with double precision.

Argument	Type and kind (*1)	Attribute	Content
N	Integer type	Output	Actual number of evaluations of a function.
MAXF	Integer type	Input	Upper limit of the number of evaluations of a function. 5≦MAXF
ILL	Integer type	Output	Indicates a calculation state in the routine. This argument is first set to 0 in the routine. Each time one of the following states is activated, a certain value is added correspondingly. (1) 1 when required precision is automatically increased because the function value is slow in convergence to 0 with $x \to -\infty$, and the required precision cannot be obtained within MAXF. (2) 2 when the state of (1) is activated with $x \to +\infty$. (3) 10000 when the function does not converge even with N<MAXF. (4) 30000 when limits on the input argument are exceeded.

*1 For double precision subroutines, all real types should be double precision real types.

(3) Performance

This routine is effective even when a solution is not well obtained with Gauss-Hermite or double exponential function type formulas if $f(x)$ is fast in convergence to 0 with $x \to \pm\infty$ (refer to Table 1 (p. 210) of bibliography [1]).

(4) Note

If this routine terminates with ILL=1 or 2, the initial value H of the step size should be increased.

Bibliography

1) Ichizo Ninomiya and Yasuyo Hatano; "Newly Registered Program SSL," Nagoya University Computer Center News, Vol. 8, No. 3, pp. 209-263 (1977).

(1987. 08. 11) (1987. 08. 21)

9. Ordinary differential equation

ODEBSS/D/Q (Solution of initial value problems for systems of first order differential equations by the rational extrapolation method)

Solution of Initial Value Problems for Systems of First Order Differential Equations by the Rational Extrapolation Method

Programmed by	Ichizo Ninomiya: April 1980
Format	Subroutine language; FORTRAN Size; 146 and 147 lines respectively

(1) Outline

When system of n first order differential equations

$y'_i=f_i(x,y_1,y_2,\cdots,y_n),i=1,2,\cdots,n,$

initial condition $y_i(x_0)=\eta_i,i=1,2,\cdots,n,$

initial value h_0 of the step size of independent variable x, the number of integration steps m, and target value x_e of an independent variable are given, ODEBSS, ODEBSSD, or ODEBSQ outputs a solution and derivative $y_i(x_f),y'_i(x_f),i=1,2,\cdots,n$ at output point $x_f=min(x_m,x_e)$ and which is the value of the differential coefficient. To do this, the subroutine uses an automatic step size control algorithm based on the Bulirsh-Stoer rationai extrapolation method. Here x_m is the value of the independent variable that is reached after m steps of integral calculation from initial values x_0, h_0.

(2) Directions

CALL ODEBSS/D/Q(X, H, Y, N, DY, EPS, DIFFUN, NSTEP, XEND, ERR, NFUN, IND)

Argument	Type and kind (*1)	Attribute	Content
X	Real type	Input/output	When initial value x_0 of independent variable is input, final value x_f is output.
H	Real type	Input/output	When initial value h_0 of the step size is input, step size h adequate for further integration from x_f is output.

Argument	Type and kind (*1)	Attribute	Content
Y	Real type One-dimensional array	Input/output	When the initial value of the solution is input, the value of the solution in x_f is output. One-dimensional array of size N.
N	Integer type	Input	Number of unknowns n of equation. 0<N≦1000
DY	Real type One-dimensional array	Output	The derivative of the solution in x_f is output. One-dimensional array of size N.
EPS	Real type	Input	Error tolerance of solution.
DIFFUN	Subroutine name	Input	Subroutine to calculate derivatives as functions of X and Y. The user should prepare this subroutine in a form of DIFFUN (X, Y, DY).
NSTEP	Integer type	Input	Number of integration steps m. NSTEP≦0 is handled as if m=∞ and always causes $x_f=x_e$.
XEND	Real type	Input	Target value x_e of independent variable. (XEND-X)*H>0 must be satisfied.
ERR	Real type One-dimensional array	Output	Estimate value of absolute truncation error of each element of solution at final step.
NFUN	Integer type	Input/output	Input: When this routine is called for the first time or there is a discontinuous change in the equation, NFUN should be 0. To calculate only the final step again, NFUN<0 should be input. Output: The total number of times the derivative have been calculated after NFUN≦0 is output.
IND	Integer type	Input/output	Input: When IND=0, the rational extrapolation method is used. When IND≠0, the polynomial extrapolation method is used. Output: IND=0: Normal termination. IND=10000: Required accuracy could not be obtained after the process was repeated six times with the step size halved. IND=30000: Argument error. A value other than the above shows the total number of times the process has been repeated because it failed to obtain required accuracy when called.

*1 For double precision subroutines, all real types should be changed to double precision real types.

(3) Calculation method

Refer to paper in bibliography [1].

(4) Example

The program shown below solves initial value problem

$$\begin{cases} y'_1 = y_1/(2(x+1)) - 2xy_2, \\ y'_2 = y_2/(2(x+1)) - 2xy_1, \end{cases} \qquad \begin{cases} y_1(0) = 1 \\ y_2(0) = 0 \end{cases}$$

The exact solution is shown as follows.

$$\begin{cases} y_1 = \sqrt{x+1}\cos x^2 \\ y_2 = \sqrt{x+1}\sin x^2 \end{cases}$$

```
C       TEST FOR ODEBSD
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION Y(2),DY(2),ERR(2),Z(2)
        EXTERNAL RHS
        X=0.DO
        H=0.1DO
        Y(1)=1.DO
        Y(2)=0.DO
        N=2
        EPS=1.D-7
        NS=0
        NF=0
        XE=2.5DO
        IND=0
        CALL ODEBSD(X,H,Y,N,DY,EPS,RHS,NS,XE,ERR,NF,IND)
        XX=X*X
        S=DSQRT(X+1.DO)
        Z(1)=DCOS(XX)*S
        Z(2)=DSIN(XX)*S
        WRITE(6,600) H,X,(Y(J),Z(J),ERR(J),J=1,2),NF,IND
  600   FORMAT(1H ,2D13.5,2(2D15.7,D11.3),2I8)
        STOP
        END
C
C       SUBROUTINE FOR DERIVATIVES
        SUBROUTINE RHS(X,Y,DY)
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION Y(2),DY(2)
        DY(1)=Y(1)*0.5DO/(X+1.DO)-Y(2)*X*2.DO
        DY(2)=Y(2)*0.5DO/(X+1.DO)+Y(1)*X*2.DO
        RETURN
        END
```

(5) Notes

1. These routines are used in the following two major ways:

(1) Only the solution with a target value is output. To do this, the target value should be put in XEND and NSTEP should be set to 0 as shown in the above example.

(2) Results on the way to the target value are output. These two methods can be used for it:

(a) The routine is called repeatedly with the target value kept in XEND and with relatively small positive values put in NSTEP. In this case, when the routine returns from the subroutine, the value of X is irregular because of automatic step size control.

(b) Output points are set appropriately (in equal intervals for instance) until the target value is reached. The routine is called repeatedly while these output points are put in XEND one by one. This has an advantage that output is obtained at regular points. If, however, output points are set too often, the step size is forcibly changed each time the routine escapes at an output point. This may deteriorate the original function of automatic step size control.

2. What can be controlled with EPS and estimated with ERR is a local truncation error and not a true error.

3. If EPS is 1 or less, it means an absolute error for each component of the solution. If it exceeds 1, it means an error relative to the maximum value.

4. Note the way of input of NFUN.

5. If IND indicates a value other than 0, 10000, and 30000, the value of the solution is not necessarily inaccurate.

6. The RKF4AS, RKF4AD, RKM4AS, and RKM4AD routines are very similar to these routines. Select the most appropriate one to your purpose.

Bibliography

1) R. Bulirsch and J. Stoer; "Numerical Treatment of Ordinary Differential Equations by Extrapolation", Numer. Math., Vol. 8, pp. 1-13 (1966).

(1987. 06. 29) (1987. 08. 21)

RK4S/D/Q/C/B

(Solution of initial value problems of systems of ordinary differential equations of the first order by the classic Runge-Kutta method of the fourth order)

Solution of Initial Value Problems of Systems of Ordinary Differential Equations of The First Order by The Classic Runge-Kutta Method of The Fourth Order

Programmed by	Ichizo Ninomiya; March 1979
Format	Subroutine language; FORTRAN Size; 27, 28, 28, 28, and 29 lines respectively

(1) Outline

When systems of n ordinary differential equations of the first order $y'_i=f_i(x,y_1,y_2,\cdots,y_n), i=1,2,\cdots,n$, initial condition $y_i(x_0)=\eta_i$, $i=1,2,\cdots,n$, step size h of independent variable x, and number of steps of integration m are given, this routine calculates numerical solution $y_i(x_r)$, $i=1,2,\cdots n$ for

$$x_r=x_0+rh, \quad r=1,2,\cdots,m$$

using the classic Runge-Kutta method of the fourth order, then outputs

$$y_i(x_m), y'_i(x_m), \quad i=1,2,\cdots n$$

(2) Directions

CALL RK4S/D/Q/C/B(X, H, Y, N, DY, DIFFUN, NSTEP, NFUN, ILL)

Argument	Type and kind (*)	Attribute	Content
X**	Real type	Input/output	When initial value x_0 of independent variable x is input, value $x_m=x_0+mh$ after m number of steps is output.

Argument	Type and kind (*)	Attribute	Content
H**	Real type	Input	Step size h of independent variable. H≠0. H<0 is also acceptable.
Y*	Real type One-dimensional array	Input/output	When the initial value of the solution is input, the value of the solution for $x=x_n$ is output. One-dimensional array of size N.
N	Integer type	Input	Number of unknowns of equation. N>0.
DY*	Real type One-dimensional array	Output	The values of the derivatives of the solution in $x=x_n$ are stored in the first N components. One-dimensional array with the size of 3N.
DIFFUN	Subroutine name	Input	Subroutine to calculate derivatives as functions of X and Y. The user should prepare this subroutine in the form of DIFFUN(X, Y, DY).
NSTEP	Integer type	Input	Number of steps of integration m. $m \geqq 1$
NFUN	Integer type	Input/output	NFUN must be set to 0 when this routine is called for the first time. Thereafter, the total number of times the derivatives subroutine has been called is output to this argument.
ILL	Integer type	Output	ILL=0: Normal termination. ILL=30000: N≦0, NSTEP≦0 or H=0.0.

* For RK4D (RK4Q, RK4C, RK4B), Y and DY should be changed to double precision real type (quadruple precision real type, complex type, double precision complex type).
** For RK4D (RK4Q, RK4C, RK4B), X and H should be changed to double precision real type (quadruple precision real type, real type, double precision real type).

(3) Example

Suppose we solve linear equations

$$\begin{cases} y'_1 = -y_2 \\ y'_2 = y_1 - y_2/x \end{cases}$$

that are obtained by variable transformation $y_1=y, y_2=-y'$ from Bessel's differential equations $y''+y'/x+y=0$ of the 0th order, under the initial condition $y_1(0)=1, y_2(0)=0$. The exact solution is $y_1=J_0(x), y_2=J_1(x)$.

The program prints intermediate results and errors every five steps with step size h of x 0.1. It thus integrates 50 steps.

```
      DIMENSION Y(2),DY(6),E(2)
      EXTERNAL BES
      N=2
      X=0.0
      H=0.1
```

```
      Y(1)=1.0
      Y(2)=0.0
      NFUN=0
      NSTEP=5
      WRITE(6,600) X,Y
      DO 10 I=1,10
      CALL RK4S(X,H,Y,N,DY,BES,NSTEP,NFUN,ILL)
      E(1)=Y(1)-BJ0(X)
      E(2)=Y(2)-BJ1(X)
   10 WRITE(6,600) X,Y,E
  600 FORMAT(1H ,10X,F5.1,2E15.7,2E11.3)
      STOP
      END
C
      SUBROUTINE BES(X,Y,DY)
      DIMENSION Y(2),DY(6)
      DY(1)=-Y(2)
      IF(ABS(X).LT.1.0E-2) DY(2)=0.5-X*X*0.1875
      IF(ABS(X).GE.1.0E-2) DY(2)=Y(1)-Y(2)/X
      RETURN
      END
```

(4) Notes

1. The name of the subroutine for derivative must be declared in the EXTERNAL statement in the calling program.

2. When this routine is called for the first time or when it is called at the point where the values of the derivative change discontinuously, NFUN must be set to 0. Because NFUN is used for both input and output, do not place a constant in this argument.

3. The local truncation error of the Runge-Kutta method of the fourth order is given by $\varepsilon_l=\alpha h^5$ for step size h. One cannot say anything definitely about the value of α because it depends on the equations. But, it can be considered about 1 when solutions change slowly. When selecting a value for h, consider this factor together with the length of integration interval. In the above example, for instance, h should be about 0.1 to obtain solutions in 4 or 5 digit precision. If it is set to 0.01, not only a truncation error becomes too small (even smaller than the minimum unit of rounding error in single precision), but also the number of integrations increases. This makes rounding errors larger and causes poor results.

4. This subroutine is useful when solutions change gradually and step size h need not be changed. When solutions change violently and there is a difficulty to select step sizes, RKF4AS/D, RKM4AS/D, or ODEBSS/D having the function of automatic step size control should be used.

(87.06.29)

RKF4AS/D

(Solution of initial value problems of systems of first order differential equations by the Runge-Kutta-Fehlberg fourth order method)

Solution of Initial Value Problems for Systems of First Order Differential Equations by the Runge-Kutta-Fehlberg Fourth Order Method

Programmed by	Ichizo Ninomiya; April 1980
Format	Subroutine language; FORTRAN Size; 77 and 78 lines respectively

(1) Outline

When a system of n first order differential equations

$y'_i=f_i(x,y_1,y_2,\cdots,y_n),\ i=1,2,\cdots,n,$

initial condition $y_i(x_0)=\eta_i,\ i=1,2,\cdots,n,$

initial value h_0 for the step size of independent variable x, the number of steps of integration m, target value x_e of the independent variable are given, RKF4AS/D uses the automatic step size control algorithm based on the combination of the Runge-Kutta-Fehlberg fourth and fifth order methods and outputs the solution at the output point $x_f=min(x_m,x_e)$ and the differential coefficient value $y_i(x_f),\ y'_i(x_f),\ i=1,2,\cdots,n.$ Where, x_m is the independent variable variable obtained after integral calculation of m steps from initial values x_0, h_0.

(2) Directions

CALL RKF4AS/D(X, H, Y, N, DY, EPS, DIFFUN, NSTEP, XEND, ERR, NFUN, ILL)

Argument	Type and kind (*1)	Attribute	Content
X	Real type	Input/output	When initial value x_0 of an independent variable is input, final value x_f is output.
H	Real type	Input/output	When initial value h_0 of a step size is input, proper step size h for further integration from x_f is output.
Y	Real type One-dimensional array	Input/output	When the initial value of a solution is input, the value of the solution in x_f is output. One-dimensional array of size N

is shown below.

The theoretical solution is as follows:

$$\begin{cases} y_1=\sqrt{x+1}\cos x^2 \\ y_2=\sqrt{x+1}\sin x^2 \end{cases}$$

```
C       TEST FOR RKF4AD
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION Y(2),DY(12),ERR(2),Z(2)
        EXTERNAL RHS
        X=0.DO
        H=0.1DO
        Y(1)=1.DO
        Y(2)=0.DO
        N=2
        EPS=1.D-7
        NS=0
        NF=0
        XE=2.5DO
        CALL RKF4AD(X,H,Y,N,DY,EPS,RHS,NS,XE,ERR,NF,ILL)
        XX=X*X
        S=DSQRT(X+1.DO)
        Z(1)=DCOS(XX)*S
        Z(2)=DSIN(XX)*S
        WRITE(6,600) H,X,(Y(J),Z(J),ERR(J),J=1,2),NF,ILL
  600   FORMAT(1H ,2D13.5,2(2D15.7,D11.3),2I8)
        STOP
        END
C
C       SUBROUTINE FOR DERIVATIVES
        SUBROUTINE RHS(X,Y,DY)
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION Y(2),DY(2)
        DY(1)=Y(1)*0.5DO/(X+1.DO)-Y(2)*X*2.DO
        DY(2)=Y(2)*0.5DO/(X+1.DO)+Y(1)*X*2.DO
        RETURN
        END
```

(5) Notes

1. This routine is used in two major objectives below:

(1) To output only the solution for an target value. To do this, specify the target value in XEND as in the example, then specify NSTEP=0.

(2) To output intermediate results until the target value is reached. There are two methods for this.

(a) While keeping the target value in XEND, put a comparatively small, positive value in NSTEP and repeat calling the subroutine. In this case, the values of X obtained after returning from the subroutine are irregular because of automatic control of step size.

(b) Define output points properly, for instance, at equal intervals, before the target

Argument	Type and kind (*1)	Attribute	Content
N	Integer type	Input	Number n of equations. 0<N≦1000
DY	Real type One-dimensional array	Output	The differential coefficient of the solution in x_f is stored in the original N elements and output. One-dimensional array with size of 6N.
EPS	Real type	Input	Error tolerance of solution.
DIFFUN	Subroutine name	Input	Subroutine used to calculate differential coefficients as a function of X and Y. The user needs to prepare this subroutine in the form of DIFFUN(X, Y, DY).
NSTEP	Integer type	Input	Number of steps of integration m. NSTEP≦0 gives the same effects as $m=\infty$ and always causes $x_f=x_e$.
XEND	Real type	Input	Target value x_e of independent variable. This argument must satisfy (XEND-X)*H>0.
ERR	Real type One-dimensional array	Output	Estimated value of absolute truncation error of each element of solution at final step.
NFUN	Integer type	Input/output	Input: Specify NFUN=0 when this routine is called for the first time or there is a discontinuous change in the equation. To calculate only final step again, specify NFUN<0. Output: If NFUN≦0 is specified, the total number of evaluations of differential coefficients is output.
ILL	Integer type	Output	ILL = 0: Normal termination ILL = 10000: Predetermined accuracy was not achieved after operation was repeated 10 times after change of the step size. ILL = 30000: The limitations on the argument were violated. Any value other than the above indicates the total number of iterations of operation done if this routine failed to achieve the expected accuracy in one call.

*1 For double precision subroutines, real types should be cahnged to double precision real types.

(3) Calculation method

Refer to the reference in bibliography [1].

(4) Example

A program to solve initial value problem

$$\begin{cases} y'_1=y_1/(2(x+1))-2xy_2, \\ y'_2=y_2/(2(x+1))-2xy_1, \end{cases} \qquad \begin{cases} y_1(0)=1 \\ y_2(0)=0 \end{cases}$$

value. Repeat calling the subroutine while putting such output points one by one in XEND. This method has the advantage of obtaining output at regular points. If output points are given too densely, however, the step size is forcibly changed each time an escape is made at each output point. This results in deterioration of the original automatic step size control function.

2. What can be controlled by EPS and estimated by ERR is a local truncation error but not a true error.

3. EPS is used to mean an absolute error for each element of a solution when it is 1 or less or a relative error for the maximum value of the size when it exceeds 1.

4. Note the input method of NFUN.

5. Even if ILL takes a value other than 0, 10000, and 30000, the value of the solution is not always inaccurate.

6. There are sister routines RKM4AS/D and ODEBSS/D which can be used in almost the same way as this routine. Use them properly as the situation demands.

Bibliography

1) E. Fehlberg; "Klassische Runge-Kutta-Formeln vierter und niedrigerer Ordnung mit Schrittweiten -Kontrolle und ihre Anwendung auf Wärmeleitungs-probleme," Computing, Vol. 6, pp. 61-71 (1970)

(1987. 07. 31) (1987. 09. 03)

RKM4AS/D

(Solution of initial value problems for systems of first order differential equations by the Runge-Kutta-Merluzzi fourth order method)

Solution of Initial Value Problems for Systems of First order Differential Equations by the Runge-Kutta-Merluzzi Fourth Order Method

Programmed	Ichizo Ninomiya; April 1980
Format	Subroutine language; FORTRAN Size; 79 and 80 lines respectively

(1) Outline

When system of n first order differential equations

$y'_i=f_i(x,y_1,y_2,\cdots,y_n),i=1,2,\cdots,n,$

initial condition $y_i(x_0)=\eta_i,i=1,2,\cdots,n,$

Initial value h_0 of the step size of independent variable x, the number of integration steps m, and target value x_e of an independent variable are given, RKM4AS or RKM4AD outputs a solution and values of derivative $y_i(x_f),i=1,2,\cdots,n$ at output point $x_f=min(x_m,x_e)$. To do this, the subroutine uses an automatic step size control algorithm based on the Runge-Kutta-Merluzzi fourth order method having the capability of evaluating accumulated truncation errors. Here x_m is the value of the independent variable that is reached after m steps of integral calculation from initial values x_0, h_0.

(2) Directions

CALL RKM4AS/D(X, H, Y, N, DY, EPS, DIFFUN, NSTEP, XEND, ERR, NFUN, ILL)

Argument	Type and kind (*1)	Attribute	Content
X	Real type	Input/output	When initial value x_0 of the independent variable is input, final target value x_f is output.
H	Real type	Input/output	When initial value h_0 of the step size is input, step size h adequate for integration from x_f is output.

Argument	Type and kind (*1)	Attribute	Content
Y	Real type One-dimensional array	Input/output	When the initial value of the solution is input, the value of the solution in x_f is output. One-dimensional array of size N.
N	Integer type	Input	Number of unknowns n of equation. $0<N\leqq 1000$
DY	Real type One-dimensional array	Work area	One-dimensional array with size 6N.
EPS	Real type	Input	Error tolerance of solution.
DIFFUN	Subroutine name	Input	Subroutine to calculate derivative as a function of X and Y. The user should prepare this subroutine in the form of DIFFUN(X, Y, DY).
NSTEP	Integer type	Input	Number of integration steps m. NSTEP$\leqq$0 is handled as if $m=\infty$, and $x_f=x_e$ always results.
XEND	Real type	Input	Target value x_e of independent variable. (XEND-X)*H>0 must be satisfied.
ERR	Real type One-dimensional array	Output	Estimated value of absolute truncation error of each element of solution at the final step.
NFUN	Integer type	Input/output	Input: When this routine is called for the first time or there is a discontinuous change in the equation, NFUN<0 should be input. To calculate only the final step again, NFUN<0 should be input. Output: The total number of times the differential coefficients have been calculated after NFUN$\leqq$0 is output.
ILL	Integer type	Output	ILL=0: Normal termination. ILL=10000: Required accuracy could not be obtained after the process was repeated ten times with different step sizes. ILL=30000: The restriction on the argument was not observed. A value other than the above shows the total number of times the process has been repeated because it failed to obtain required accuracy.

*1 For double precision subroutines, all real types should be changed to double precision real types.

(3) Calculation method

Refer to paper in bibliography [1].

(4) Example

The program shown below solves initial value problem

$$\begin{cases} y'_1 = y_1/(2(x+1)) - 2xy_2, \\ y'_2 = y_2/(2(x+1)) - 2xy_1, \end{cases} \qquad \begin{cases} y_1(0) = 1 \\ y_2(0) = 0 \end{cases}$$

The exact solution is shown as follows.

$$\begin{cases} y_1 = \sqrt{x+1}\cos x^2 \\ y_2 = \sqrt{x+1}\sin x^2 \end{cases}$$

```
C       TEST FOR RKM4AD
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION Y(2),DY(12),ERR(2),Z(2)
        EXTERNAL RHS
        X=0.D0
        H=0.1D0
        Y(1)=1.D0
        Y(2)=0.D0
        N=2
        EPS=1.D-7
        NS=0
        NF=0
        XE=10.D0
        CALL RKM4AD(X,H,Y,N,DY,EPS,RHS,NS,XE,ERR,NF,ILL)
        XX=X*X
        S=DSQRT(X+1.D0)
        Z(1)=DCOS(XX)*S
        Z(2)=DSIN(XX)*S
        WRITE(6,600) H,X,(Y(J),Z(J),ERR(J),J=1,2),NF,ILL
  600   FORMAT(1H ,2D13.5,2(2D15.7,D11.3),2I8)
        STOP
        END
C
C       SUBROUTINE FOR DERIVATIVES
        SUBROUTINE RHS(X,Y,DY)
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION Y(2),DY(2)
        DY(1)=Y(1)*0.5D0/(X+1.D0)-Y(2)*X*2.D0
        DY(2)=Y(2)*0.5D0/(X+1.D0)+Y(1)*X*2.D0
        RETURN
        END
```

(5) Notes

1. These routines are used in the following two major ways:

(1) Only the solution with a target value is output. To do this, the target value should be put in XEND and NSTEP should be set to 0 as shown in the above example.

(2) Results on the way to the target value are output. These two methods can be used for it:

(a) The routine is called repeatedly with the target value kept in XEND and with relatively

small positive values put in NSTEP. In this case, when the routine returns from the subroutine, the value of X is irregular because of automatic step size control.

(b) Output points are set in appropriately (in equal intervals for instance) until the target value is reached. The routine is called repeatedly while these output points are put in XEND one by one. This has an advantage that output is obtained at regular points. If, however, output points are set too often, the step size is forcibly changed each time the routine escapes at an output point. This may deteriorate the original function of automatic step size control.

2. What can be controlled with EPS and estimated with ERR is an accumulated truncation error and not a true error.

3. If EPS is 1 or less, it means an absolute error for each component of the solution. If it exceeds 1, it means an error relative to the maximum value.

4. Note the way of input of NFUN.

5. If ILL indicates a value other than 0, 10000, and 30000, the value of the solution is not necessarily inaccurate.

6. The RKF4AS, RKF4AD, ODEBSS, and ODEBSD routines are very similar to these routines. Select the most appropriate one to your purpose.

Bibliography

1) P. Merluzzi et al; "Runge-Kutta Integration Algorithms with Built-in Estimation of the Accumulated Truncation Error", Computing, Vol. 20, pp. 1-16 (1978).

(1987. 06. 29)

10. Elementary function

ALANGV/DLANGV (Langevin Function)

Langevin Function

Programmed by	Ichizo Ninomiya, April 1981
Format	Function Language; 21 and 26 lines respectively

(1) Outline

ALANGV (DLANGV) calculates Langevin functions $L(x)=\coth x-1/x$ for a single (double) precision real numbers x with single (double) precision.

(2) Directions

1. ALANGV(X) and DLANGV(D)

X(D) is an arbitrary expression of a single (double) precision real number type. DLANGV requires the declaration of double precision.

2. Range of argument

There is no limit on arguments.

(3) Calculation method

1. If $|x|\leq 4$, $L(x)=xR(x^2)$ is calculated with the optimal rational approximation R.
2. If $4<|x|<10$ (or $4<|x|<20$ in case of DLANGV),

$$L(x)=sign(x)\,(2e^{-|x|}/(1-e^{|x|})-1/|x|+1)$$ is calculated.

3. If $|x|\geq 10$ (or $|x|\geq 20$ in case of DLANGV), $L(x)=sign(x)\,(1-1/|x|)$ is calculated.

(4) Note

If $L(x)$ is calculated based on the definition formula, precision is lost near $x=0$.

(1987.03.31)

ALOG1/DLOG1/QLOG1/CLOG1/CDLOG1/CQLOG1 (Function log(1+x))

Function log(1+x)

Programmed by	Ichizo Ninomiya, April 1981, April 1977, December 1987
Format	Function Language: FORTRAN; Size: 18, 24, 34, 14, 15, and 15 lines respectively

(1) Outline

ALOG1(DLOG1, QLOG1) calculates $\log(1+x)$ for single (double, quadruple) precision real numbers x with single (double, quadruple) precision.

CLOG1(CDLOG1, CQLOG1) calculates $\log(1+z)$ for single (double, quadruple) precision complex numbers z with single (double, quadruple) precision.

(2) Directions

1. ALOG1(X), DLOG1(D), QLOG1(Q), CLOG1(C), CDLOG1(B), and CQLOG1(Z)

   X(D, Q) is an arbitrary expression of a single (double, quadruple) precision real number type.

   C(B, Z) is an arbitrary expression of a single (double, quadruple) precision complex number type. Functions other than ALOG1 require the declarations of corresponding types.

2. Range of argument

   X>-1, D>-1, and Q>-1 for ALOG1, etc.

   C≠-1, B≠-1, and Z≠-1 for CLOG1, etc.

3. Error processing

   If an argument outside the range is given, an error message is printed, and the calculation is continued with the function value as 0. (See "FNERST.")

(3) Calculation method

1. ALOG1/DLOG1/QLOG1

   (1) If $x \leq -1$, an error is assumed.

   (2) If $-1/2 \leq x < 1$, transformation $y=x/(x+2)$ is performed, and $\log(1+x)=\log(1+y)/(1-y)$ is calculated using the polynomial approximation of y.

   (3) If $-1 < x < -1/2$ or $x \geq 1$, $\log(1+x)$ is calculated as it is by the elementary function $\log(x)$.

2. CLOG1/CDLOG1/CQLOG1

(1) If $|z| \leqq 1$

$log(1+(2+x)\cdot x+y^2)/2+i\,tan^{-1}(y/(1+x))$ is calculated. Where, $z=x+iy$.

$log(1+(2+x)x+y^2)$ is calculated using ALOG1/DLOG1/QLOG1, and $tan^{-1}(y/(1+x))$ is calculated using the standard function ATAN2/DATAN2/QATAN2.

(2) If $|1+z| \neq 0$, $log(1+z)$ is calculated as defined.

(3) If $|1+z|=0$, an error results.

(4) Note

If the function in this section is calculated using the standard function as defined, precision is lost near the origin.

(1987. 07. 31) (1988. 02. 15)

ASINH/DASINH/QASINH, ACOSH/DACOSH/QACOSH, and ATANH/DATANH/QATANH (Inverse Hyperbolic Function)

Inverse Hyperbolic Function

Programmed by	Ichizo Ninomiya, April 1974, revised in April 1977
Format	Function Language: FORTRAN; Size: 18, 26, 37, 11, 12, 11, 18, 24, and 34 lines respectively

(1) Outline

ASINH(DASINH, QASINH), ACOSH(DACOSH, QACOSH), and ATANH(DATANH, QATANH) calculate $sinh^{-1}x$, $cosh^{-1}x$, and $tanh^{-1}x$ respectively with single (double, quadruple) precision for a single (double, quadruple) precision real number x,

where,

$$sinh^{-1}x=log(x+\sqrt{1+x^2})$$

$$cosh^{-1}x=log(x+\sqrt{x^2-1})$$

$$tanh^{-1}x=\frac{1}{2}\log\frac{1+x}{1-x}$$

(2) Directions

1. ASINH(X), ACOSH(X), ATANH(X), DASINH(D), DACOSH(D), DATANH(D), QASINH(Q), QACOSH(Q), QATANH(Q)

X(D, Q) are arbitrary expressions of a single (double, quadruple) precision real type. The name of a function of double (quadruple) precision requires the declaration of double (quadruple) precision.

2. Range of argument

An inverse hyperbolic sine function has no limitation on arguments.

$X\geq 1$, $D\geq 1$, and $Q\geq 1$ for an inverse hyperbolic cosine function.

$|X|<1$, $|D|<1$, and $|Q|<1$ for an inverse hyperbolic tangent function.

3. Error processing

If an argument outside the range is given, an error message is printed, and the calculation is continued with the function value as 0. (See FNERST.)

(3) Calculation method

1. ASINH(DASINH)

(1) If $|x|<3/4$, $sinh^{-1}x$ is calculated by polynomial approximation.

(2) If $|x|\geqq 3/4$, the following calculation is executed.

$$y=|x|, sinh^{-1}x=signx\cdot sinh^{-1}y, sinh^{-1}y=log(y+\sqrt{1+y^2})$$

(3) If $y\geqq 4096 (y\geqq 3\cdot 10^8)$, $sinh^{-1}y=log2y$.

2. ACOSH(DACOSH)

(1) If $x<1$, an error is assumed.

(2) If $1<x<4096 (1<x<3\cdot 10^8)$, the following calculation is executed.

$$cosh^{-1}x=log(x+\sqrt{x^2-1})$$

(3) If $x\leqq 4096 (x\geqq 3\cdot 10^8)$, $cosh^{-1}x=log2x$.

3. ATANH(DATANH)

(1) If $|x|\geqq 1$, an error is assumed.

(2) If $|x|\leqq 1/3$, $tanh^{-1}x$ is calculated by polynomial approximation.

(3) If $1/3<|x|<1$, the following calculation is executed.

$$tanh^{-1}x=\frac{1}{2}\log\frac{1+x}{1-x}$$

(4) Note

All the functions in this section are simple functions that are defined with a logarithmic function. However, if they are calculated as described in the definition expressions, precise values cannot be obtained for the argument of small absolute values. Since special measures are taken for the argument of small absolute values, values of the functions of this section do not suffer the drop of accuracy.

(1987. 06. 30)

CABS1/CDABS1/CQABS1 (Sum of Moduli of Real and Imaginary Parts of a Complex Number)

Sum of Moduli of Real and Imaginary Parts of a Complex Number

Programmed by	Ichizo Ninomiya, January 1980
Format	Function Language: Assembler; Size: 42 lines

(1) Outline

CABS1 (CDABS1, CQABS1) calculates $\|z\|_1=|x|+|y|$ for a single (double or quadruple) precision complex number $z=x+iy$ with single (double or quadruple) precision.

(2) Directions

1. CABS1(C), CDABS1(B), and CQABS1(Z)

   C (B, Z) is an arbitrary expression of a single (double, quadruple) precision complex number type. CDABS1 (CQABS1) requires the declaration of double (or quadruple) precision.

2. There is no limit on arguments.

(3) Note

1. The elementary external function CABS (CDABS, CQABS) gives the absolute value $\|z\|_2=|z|=\sqrt{x^2+y^2}$ of a usual meaning of complex numbers $z=x+iy$. However, it contains a square root and cannot be calculated directly. Thus, it is slow in calculation speed.

   In a convergence test, the smallness of complex numbers can be fully checked with the sum of simple absolute values $\|z\|_1$. This is the raison d'etre of the present function routine. By the way, in the tests with the same ϵ, that is, in $\|z\|_2<\epsilon$ and $\|z\|_1<\epsilon$, the latter is stronger. That is, if $\|z\|_1<\epsilon$, we always have $\|z\|_2<\epsilon$.

2. CDABS1 (CQABS1) can be replaced with DCABS1 (QCABS1).

(1987.06.30)

COMB/DCOMB/QCOMB (Binomial Coefficient)

Binomial Coefficient

Programmed by	Ichizo Ninomiya, April 1982
Format	Function Language: FORTRAN; Size: 25, 26, and 26 lines respectively

(1) Outline

COMB (DCOMB, QCOMB) calculates the following binomial coefficient for integers m,n with single (double or quadruple) precision.

$$mCn=\left(\begin{array}{c} m \\ n \end{array}\right)=\frac{m!}{n!(m-n)!}$$

(2) Directions

1. COMB(M, N), DCOMB(M, N), and QCOMB(M, N)

   M and N are arbitrary expressions of an integer type.

2. Range of argument

   $1\leqq M,\ 0\leqq N\leqq M$

   However, the range that function values overflow is excluded.

3. Error processing

   If an argument outside the range is given, an error message is printed, and the calculation is continued with the function value as 0. (See "FNERST.")

(3) Calculation method

1. Let $k=min(n,m-n)$.
2. If $m\leqq 56$ and $k>8$, we compute as follows; calling FCTRL (DFCTRL, QFCTRL).

   $$mCk=\frac{m!}{k!(m-k)!}$$

3. In a case other than the above,

   the recurrence formula

   $$mC_r=mC_{r-1}\cdot\frac{m-r+1}{r}$$

is repeated, beginning from $mC_0=1$.

(1987. 06. 30) (1987. 08. 10)

EXP1/DEXP1/QEXP1, CEXP1/CDEXP1/CQEXP1 (Function exp(x)-1)

Function e^x-1

Programmed by	Ichizo Ninomiya; December 1987, April 1981
Format	Function Language; FORTRAN Size; 21, 25, 28, 19, 21, and 21 lines respectively

(1) Outline

EXP1, DEXP1, and QEXP1 each calculate e^x-1, with single, double, or quadruple precision, for a single, double, or quadruple real number x.

CEXP1, CDEXP1, and CQEXP1 each calculate e^z-1, with single, double, or quadruple precision, for a single, double, or quadruple complex number z.

(2) Directions

1. EXP1(X), DEXP(D), QEXP1(Q), CEXP1(C), CDEXP1(B), and CQEXP1(Z)

   X, D, and Q are arbitrary single, double, and quadruple real expressions respectively.

   C, B, and Z are arbitrary single, double, and quadruple complex expressions respectively.

   The function names other than those for single precision need the declaration of the corresponding types.

2. Range of argument

   EXP1 etc.: $X \leq 174.673, D \leq 174.673, Q \leq 174.673$

   CEXP1 etc.: $REAL(C) \leq 174.673, REAL(B) \leq 174.673, REAL(Z) \leq 174.673$

   $|IMAG(C)| \leq 2^{18}\pi, |IMAG(B)| \leq 2^{56}\pi, |IMAG(Z)| \leq 2^{106}\pi.$

3. Error processing

   If the specified argument is outside the range, an error message is printed but calculation continues with the function value assumed to be 0. (See FNERST.)

(3) Calculation method

1. EXP1/DEXP1/QEXP1

   (1) $f(x)=-1$ in case of $x<-18.421$ (in case of $x<-41.447$ with DEXP1, and in case of $x<-77.633$ with QEXP1).

   (2) In case of $|x| \leq 1$, polynomial approximations P and Q are used to calculate

$e^x-1=2xP(x^2)/[Q(x^2)-xP(x^2)]$.

(3) In case of x other than those in (1) and (2), e^x-1 is calculated as defined.

2. CEXP1/CDEXP1/CQEXP1

(1) In case of $|x| \leq 1$,

$(e^x-1)cosy-2sin^2y/2+ie^xsiny$ is calculated. $x+iy$ is used as the argument.

EXP1/DEXP1/QEXP1 is called to calculate e^x-1.

(2) In case of $|x| >$, $e^x(cosy+isiny)-1$ is calculated as defined.

(4) Note

If the function in this section is calculated by the standard function as defined, severe cancellation occurs near the origin.

(1987. 07. 31) (1988. 01. 27)

FASTEE (Fast High Precision Calculation of e)

Fast High Precision Calculation of e

Programmed by	Ichizo Ninomiya, January 1983
Format	Subroutine Language; FORTRAN and assembler Size; 63 and 212 lines respectively

(1) Outline

FASTEE calculates and outputs the value of e at high speed with required precision.

(2) Directions

CALL FASTEE (N, P, W, ILL)

Argument	Type and kind (*1)	Attribute	Content
N	Integer type	Input	Number of decimal digits of e. 100≦N
P	Double precision real type One-dimensional array	Work area	Size of N/10.
W	Double precision real type One-dimensional array	Work area	Size of N/10.
ILL	Integer type	Output	Error code. ILL=0: Normal termination. ILL=30000: 100>N

(3) Calculation method

The Taylor series e=1+1/1!+1/2!+1/3!+··· is truncated at the n-th term where n!>10**(N) is established according to the required number of digits N. Then, it is arranged into e=2+1/2(1+1/3(1+···+(1/(n-1))(1+1/n))···)), and calculated in the order of n, n-1, ··.

(4) Note

1. The output is separated every 10 digits, and printed every 100 digits per line.

2. The calculation speed on the M-200 is listed below.

N	1000	10000	100000
CPU	0.012 second	1.0 second	100 seconds

(1987.08.11)

FASTPI (Fast High Precision Calculation of π)

Fast High Precision Calculation of π

Programmed by	Ichizo Ninomiya, January 1983
Format	Subroutine Language: FORTRAN and assembler; Size: 54 and 382 lines respectively

(1) Outline

FASTPI calculates and outputs the value of π with required precision.

(2) Directions

CALL FASTPI(N, P, W, ILL)

Argument	Type and kind (*1)	Attribute	Content
N	Integer type	Input	Number of decimal digits of π. $100 \leq N$
P	Double precision real type One-dimensional array	Work area	Size of N/10.
W	Double precision real type One-dimensional array	Work area	Size of N/10.
ILL	Integer type	Output	Error code. ILL=0: Normal termination. ILL=30000: 100>N

(3) Calculation method

Machin's formula: $\pi = 4\arctan(1/5) - 16\arctan(1/239)$ is used.

The Taylor series a r c t a n(1/m)=1/m-1/3m**3+1/5m**5-··· is truncated at the term of m**(2n+1)>10**N according to the required number of digits N and arranged into

a r c t a n(1/m)=1/m(1-1/m**2(1/3-1/m**2(1/5-···-1/m**2(1/(2n-1)-1/m**2(1/(2n+1))···))) before the terms of the two arc tangents are calculated in the order of 2n+1, 2n-1, ···.

(4) Note

1. The output is separated every 10 digits, and printed every 100 digits per line.
2. The calculation speed on the M-200 is listed below.

N	1000	10000	100000
CPU	0.056 second	5.1 seconds	643 seconds

Bibliography

1) Ichizo Ninomiya; "Calculation of π," Proceedings of the 25th Symposium of Information Processing Soc. of Japan (III), pp.1167-1168 (1982).

(1987.08.06)

SINHP/DSINHP/QSINHP, COSHP/DCOSHP/QCOSHP, TANHP/DTANHP/QTANHP and COTHP/DCOTHP/QCOTHP (Trigonometric Functions for the Argument $\pi/2 \cdot x$)

Trigonometric Functions for the Argument $\pi/2 \cdot x$

Programmed by	Ichizo Ninomiya, January 1980
Format	Function Language; Assembler (quadruple precision type is FORTRAN) Size; 117, 152, 47, 117, 152, 47, 134, 174, 55, 134, 174, and 55 lines respectively

(1) Outline

SINHP(DSINHP, QSINHP), COSHP (DCOSHP, QCOSHP), TANHP (DTANHP, QTANHP) and COTHP (DCOTHP, QCOTHP) calculate $\sin \pi/2 \cdot x$, $\cos \pi/2 \cdot x$, $\tan \pi/2 \cdot x$ and $\cot \pi/2 \cdot x$ respectively with single (double, quadruple) precision for a single (double, quadruple) precision real number x.

(2) Directions

1. SINHP(X), COSHP(X), TANHP(X), and COTHP(X), DSINHP(D), DCOSHP(D), DTANHP(D), DCOTHP(D), QSINHP(Q), QCOSHP(Q), QTANHP(Q), QCOTHP(Q)

   X (D, Q) is an arbitrary expression of a single (double, quadruple) precision real type. The name of a function of double (quadruple) precision requires the declaration of double (quadruple) precision.

2. Range of argument

$$|X| \leq 2^{19} \fallingdotseq 5.2 \cdot 10^5, |D| \leq 2^{51} \fallingdotseq 2.2 \cdot 10^{15}, |Q| \leq 2^{106} \fallingdotseq 8.1 \cdot 10^{31}$$

3. Error processing

   If a given argument is outside the range or a singular point, an error message is printed, and the calculation is continued with the function value as 0. (See FNERST.)

(3) Note

1. If an argument contains π as its factor, the value of a trigonometric function can be calculated using usual external elementary functions such as SIN and COS. However, it is more reasonable to use various functions in this section because of the following reasons:

(1) The value of π need not be written. (2) The speed is faster by two multiplications.

(3) Precision is higher.

For example, SINHP(X) is better than SIN(1.570796*X), and DCOSHP(X+X) is better than DCOS(3.14159265358979320*X).

COSHP(1.0) becomes precisely 0, but COS(1.570796) does not.

2. HP at the end of a function name means HALF PI. Because H at the end of a hyperbolic function name means HYPERBOLIC, do not confuse them with each other.

3. If an argument contains an error, the function value contains an error in its last digits. The number of incorrect digits is roughly the same as the number of digits of integral part of the argument. This is similar for standard functions.

4. Precision cannot be guaranteed for the function value near the pole of TANHP and COTHP.

(1987.06.29)