13.2 Numerical Evaluation of Integrals Over Several Dimensions

A. Purpose

This collection of subprograms estimates the value of the integral I_n, where

$$\{I_0\}_f = \emptyset$$

$$I_k = \int_{a_k(x_{k+1},...,x_n,\{I_{k'}\}_{a_k})}^{b_k(x_{k+1},...,x_n,\{I_{k'}\}_{b_k})} f_k(x_k,...,x_n,\{I_{k-1}\}_{f_k})\,dx_k, \quad 1 \le k \le n-1$$

$$I_n = \int_{a_n}^{b_n} f_n(x_n,\{I_{n-1}\}_{f_n})\,dx_n,$$

a_n and b_n are constants, a_1, ..., a_{n-1}, b_1, ..., b_{n-1} and f_1, ..., f_n are functions provided by the user. The notation $\{I_{k-1}\}_g$ means a set of zero or more integrals of dimension less than k, used in the calculation of g, and k' is less than the dimensionality of the integral enclosing I_k(k' may be greater than k). Usually, f_k is I_{k-1} for $k > 1$, and $\{I_{k'}\}_{a_k}$ and $\{I_{k'}\}_{b_k}$ are empty.

Using $n = 2$ for illustration, this formulation of the multiple integration problem includes the simple case of

$$I_2 = \int_{a_2}^{b_2} \int_{a_1(x_2)}^{b_1(x_2)} f_1(x_1,x_2)\,dx_1\,dx_2,$$

but it also allows for improved computational efficiency if the integrand can be factored as

$$I_2 = \int_{a_2}^{b_2} f_2(x_2) \int_{a_1(x_2)}^{b_1(x_2)} f_1(x_1,x_2)\,dx_1\,dx_2,$$

and in addition it allows for the more general case of

$$I_2 = \int_{a_2}^{b_2} f_2(x_2, I_1^{(a)}(x_2), I_1^{(b)}(x_2), ...)\,dx_1\,dx_2,$$

where $I_1^{(a)}(x_2)$, $I_1^{(b)}(x_2)$, ... are integrals over one dimension. This includes the case in which, for example, $I_1^{(a)}(x_2)$ is a limit for $I_1^{(b)}(x_2)$.

B. Usage

Described below under B.1 through B.5 are:

B.1 Program Prototype, Single Precision.

B.1.a The Calling Routine

INTEGER NDIMI, NWORK, IOPT($\ge k$)
[k depends on options used (≥ 2).]

REAL ANSWER, WORK(NWORK)

Assign values to NDIMI and NWORK.

Assign values to IOPT() and elements of WORK() referenced by options (see Section B.3). For simplest usage, set

IOPT(2) = 0

CALL SINTM (NDIMI, ANSWER, WORK, NWORK, IOPT)

If the computation is successful (see the description of values of IOPT(1) below) the result is in ANSWER and the estimate of the error in the result is in WORK(1).

B.1.b Argument Definitions

NDIMI [in] Number of dimensions of integration, n in Section A.

ANSWER [out] Estimate of the integral.

WORK() [inout] On completion, WORK(1) contains an estimate of the upper bound of the magnitude of the difference between ANSWER and the true value of the integral. During the integration, WORK(1:NDIMI) contains the abscissas x_1 through x_{NDIMI}, WORK(NDIMI+1:2×NDIMI) contains the lower limits a_1 through a_{NDIMI}, and WORK(2×NDIMI+1:3×NDIMI) contains the upper limits b_1 through b_{NDIMI}. WORK(NDIMI+1:NWORK) may be referenced by the option vector IOPT() as described below and in Section B.3, and to pass parameters to SINTF as described in Section B.1.f.

NWORK [in] Specifies the amount of space allocated for the array WORK(). NWORK must be $\ge 220 \times \text{NDIMI} - 217$.

Table of Options for SINTM

Option Number	Brief Description
0	No more options.
1	No effect. Reserved for future use. Do not use option 1.
2	Select level of diagnostic output.
3	Specify error tolerances.
4	Specify an a posteriori estimate of the absolute error in the calculated value of the integrand.
5	Specify an a priori estimate of the relative error expected in the calculated values of integrands.
6	Use reverse communication.
7	Specify minimum index of quadrature formula.
8	No effect. The parameter may provide information to SINTF.
9	Specify maximum number of integrand evaluations allowed.
10	Return number of integrand evaluations used.
11	Specify location of singularity or discontinuity in integrand.
12	Specify absolute errors in the limits.
13	Specify location in IOPT of notification of nonstandard dimension changes.

IOPT() [inout] Used to return status information to the user, to allow the selection of options, and to pass parameters to SINTF as described in Section B.1.f. IOPT(1) returns a status indicator with the possible values:

- −NDIMI Normal termination with either the absolute or relative error tolerance criteria satisfied (see option 3 below and in Section B.3).
- −NDIMI−1 Normal termination with neither the absolute nor relative error tolerance criteria satisfied, but the tolerance relative to the locally achievable precision is satisfied. This is the normal status value when no tolerances are specified. (See option 3 below and in Section B.3).
- −NDIMI−2 Normal termination with none of the error tolerance criteria satisfied. (See option 3 below and in Section B.3).
- −NDIMI−3 Error termination, NWORK is too small.
- −NDIMI−4 Bad value for an element of IOPT().
- −NDIMI−5 Too many function values needed (see option 9 below and in Section B.3).
- −NDIMI−k−5 Error termination. The k^{th} integrand apparently contains a non-integrable singularity. The approximate abscissa of the singularity in the k^{th} dimension is in ANSWER, and the abscissas of exterior dimensions are in WORK(k+1:NDIMI). WORK(1) contains a large number.
- −NDIMI−NDIMI−k−5 When applying the usage that allows changing the dimensionality of the next integral to be computed in a nonstandard way (Section B.1.e), the specified dimensionality k of an inner integral is not less than the dimensionality of the outer integral.

The remainder of IOPT() may be used to select options and to pass parameters to SINTF. If no options are selected, set IOPT(2) = 0.

See Section B.3 for a detailed description of options and the table on the left for an overview.

B.1.c The User-Supplied Subroutine SINTF to Calculate Integrands and Limits

SINTM requires that the user provide values of the integrand and values of the lower and upper limits, and allows the user to make functional transformations of inner integrals before they are used as integrands of outer integrals. In the case of simple usage these values are provided by a user-supplied subroutine of the form:

```
SUBROUTINE SINTF(ANSWER, WORK, IFLAG)
REAL  ANSWER, WORK(*)
INTEGER  IFLAG
. . .
```

B.1.d Argument Definitions for SINTF

ANSWER [inout] Usage depends on IFLAG.

WORK(1:NDIMI**)** [inout] Storage for the currently needed values of x_1, ..., x_{NDIMI}. When IFLAG $\neq 0$, x_1 is not needed and WORK(1) has a different usage described below.

WORK(NDIMI+1:2×NDIMI**)** [inout] Storage for lower integration limits, a_1, ..., a_{NDIMI}.

WORK(2×NDIMI+1:3×NDIMI) [inout] Storage for upper integration limits, b_1, ..., b_{NDIMI}.

WORK(220 × NDIMI − 216 : NWORK**)** [inout] May be used to pass extra information into SINTF, See Section B.1.f.

IFLAG [inout] On input, IFLAG indicates the action to be taken by SINTF. The value of IFLAG may be changed by SINTF. IFLAG may be used to pass extra information into SINTF, See Section B.1.f.

Most mathematical software that requires user defined function information allows the user to pass in the name of a subprogram for evaluating the function. The approach used here has the advantage of not requiring the user to declare his subprogram in an external statement (It's not unusual for this to be forgotten.), and of giving a meaningful diagnostic when no such routine is provided. If one is solving different problems with different programs, one can use different file names for the different function subprograms, all of which would use the same entry name. The linker must then be told which of these function subprograms should be used in forming the executable file. If one wants to solve several different problems in the same program, one should code the separate cases in one subprogram and select the one desired by passing a "case" variable into the routine. This can either be done as part of IFLAG as mentioned above, or can be done through the use of named common. One can also use reverse communication and call subprograms with different names for the different cases.

B.1.e Actions to be Accomplished by SINTF

IFLAG $= 0$ The Inner Integrand
When IFLAG = 0, SINTF must compute the inner integrand, f_1, as a function of x_1, ..., x_{NDIMI}, and return the result in ANSWER.

IFLAG < 0 After Computing I_{-IFLAG}
When IFLAG < 0, say IFLAG = $-i$, the current value of the integral I_i is provided in ANSWER, and the estimated error E_i in this value is provided in WORK(1). Frequently, one will have $f_{i+1} = I_i$, and a_i and b_i will not depend on integrals, in which case $\varepsilon_{i+1} = E_i =$ WORK(1), and SINTF can simply RETURN, taking no action.

In more complicated cases, SINTF must compute the integrand f_{i+1} as a function of x_{i+1}, ..., x_{NDIMI}, I_i, and perhaps other integrals as described in the next paragraph. The value of f_{i+1} must be in ANSWER on return, and the error ε_{i+1} in f_{i+1} must be in WORK(1).

If f_{i+1} depends on integral(s) not yet computed then SINTF must remember I_i and E_i for subsequent use, and set IFLAG to the number of dimensions of the next integral to be evaluated; IFLAG must be less than the number of dimensions of the enclosing integral. If option 13 is selected then K13 is used for this communication instead of IFLAG. IFLAG is the first element of the IOPT vector passed to SINTM (see Section B.1.f). It is the responsibility of SINTF to remember the state of computation of the several integrals upon which f_{i+1} depends. That is, to know when to change IFLAG (or K13), when not to change it, and when to evaluate f_{i+1}.

To estimate errors and control the selection of evaluation points during calculation of I_{i+1}, SINTM needs an estimate of the error in f_{i+1}. In the simple case when $f_{i+1} = I_i$, and a_i and b_i do not depend on integrals, one has $\varepsilon_{i+1} = E_i =$ WORK(1) and no special action is required. In general, however, one must compute

$$\text{WORK(1)} \;=\; \sum_{\{\, j | I_i^{(j)} \in \{I_i\}_f \,\}} |\partial f_{i+1}/\partial I_i^{(j)}|\; E_i^{(j)}$$

Where $E_i^{(j)}$ is the error in $I_i^{(j)}$. If some arguments of f_{i+1}, say y_j, cannot be precisely calculated and represented, then one must add $\Sigma\ |\partial f_{i+1}/\partial y_j|\ E(y_j)$, where E($y_j$) is the error in the calculation of y_j, onto the above sum.

IFLAG > 0 Before Computing I_{IFLAG}
When IFLAG > 0, say IFLAG = i, SINTF must compute a_i and b_i as functions of x_{i+1}, ..., x_{NDIMI}, storing a_i in WORK(NDIMI + IFLAG) and b_i in WORK(2×NDIMI + IFLAG). SINTF will be called only once with IFLAG = NDIMI, since the outer limits a_{NDIMI} and b_{NDIMI} must be constants, and thus need to be set only once. These may be stored into WORK() either by the user's main program before the initial call to SINTM or else by SINTF when it is called with IFLAG = NDIMI. Similarly, if the limits a_k and b_k of I_k are constant, they may be stored into WORK() by the user's main program before the initial call to SINTM, or on any call to SINTF for which IFLAG $\geq k$.

Suppose a limit a_i or b_i depends on variables of integration $\{x_m |\ i < k \leq m \leq$ NDIMI$\}$. Then a_i or b_i may be calculated any time that $i \leq$ IFLAG $< k$, but for maximum efficiency should be calculated when IFLAG $= k-1$. Similarly a subexpression of f_i that depends on $\{x_m |\ i < k \leq m \leq$ NDIMI$\}$ should be calculated when IFLAG $= k$, and used when IFLAG $= -i$.

For $1 \leq i <$ NDIMI the integral I_i, having the limits a_i and b_i, may subsequently be used as an argument in computing f_{i+1}. Let q be the maximum of $|\partial f_{i+1}/\partial I_i|$ over all integrals on which f_{i+1} depends. If $q \neq 1$, then during an entry at which a_i and b_i are being computed, SINTF must also compute q, or an upper bound for q, and store this value in WORK(1). This value is used internally to decide how much accuracy is needed for the coming integration. Note that if f_{i+1} is of the form $q(x_{i+1},\ \ldots,\ x_{NDIMI})I_i$, then the q computed here can be saved and reused for computing the values of ANSWER and WORK(1) when IFLAG $= -(i+1)$.

If it is known that the integrand, f_i, has a single singularity affecting integration from a_i to b_i, SINTF should transmit information on the type and location of this singularity to SINTM during each entry that computes a_i and b_i. To do this, SINTF should store the location (on the x_i axis) of the singularity into WORK(|K11|), where K11 is transmitted to SINTM, either by a call

to SINTOP with Option 11, or by storing K11 into IFLAG. If the singularity is at one of the limits, then NDIMI $<$ |K11| $\leq 3 \times$ NDIMI is allowed. Otherwise, one should have |K11| $> 220 \times$ NDIMI $- 217$. The sign of K11 affects the internal transformations made to cope with the singularity as described in Section B.3 of Chapter 13.1. If singularities are present in more than one dimension of the multiple integration, one must use a different value of |K11| for each different singularity location.

SINTM provides for the case that f_{i+1} might depend on several integrals, some of which have fewer than i dimensions. When it is necessary to evaluate an integral of dimension less than i, set IFLAG (or K13 if option 13 has been selected) to the number of dimensions of the integral to be evaluated, and set the appropriate limits into WORK() as described above.

To illustrate nonstandard changes in dimensionality, suppose that the integrand of a three dimensional integral depends on a function of several one- and two-dimensional integrals, and suppose one chooses to evaluate the one-dimensional integrals first. When SINTM first requests SINTF to provide the limits of the inner (two-dimensional) integral (IFLAG = 2), change IFLAG to 1. Upon completion of each but the last of the one dimensional integrals (IFLAG = -1), set IFLAG to 1. Upon completion of the last of the one-dimensional integrals, and upon completion of all but the last two-dimensional integral, set IFLAG to 2. SINTF must keep track of it's state using SAVE variables. That is, SINTF is responsible for knowing which integrand to evaluate when IFLAG = 0, which transformation to apply when IFLAG < 0, and which limits to supply when IFLAG > 0. For this example, an error will be signaled with IOPT(1) $= -14 = -$NDIMI $-$ NDIMI $- 3 - 5$ if one sets IFLAG ≥ 3.

If either of the limits a_i or b_i are imprecisely known or imprecisely representable (*e.g.*, they depend on integrals, or there is significant cancellation in their evaluation), then for maximum reliability the errors in the limits should be made known to SINTM by invoking SINTOP and selecting option 12. See Section B.3.

B.1.f Passing Extra Information into SINTF

The argument WORK of SINTF is the vector WORK passed from the user's calling routine to SINTM. WORK() may be used to provide information for options as described in Section B.3, and to pass floating point information from the user's calling routine into SINTF.

The argument IFLAG of SINTF is the first element of the IOPT() vector passed from the user's calling routine to SINTM. Elements of IOPT() (after the first) that are not used for options as described in Section B.3 may be used to pass integer valued information into SINTF. In addition, the parameter of option 8 described in Section B.3 may be examined by SINTF. If IFLAG is used in this way, it must be declared

INTEGER IFLAG(*)

and IFLAG(1) must be examined to determine the action.

B.2 Program Prototype, Single Precision, Reverse Communication.

B.2.a The Calling Routine

INTEGER NDIMI, NWORK, IOPT(≥ 3)

REAL ANSWER, WORK(NWORK)

Assign values to NDIMI, NWORK and IOPT() and elements of WORK() referenced by options. Constant limits may be stored in the appropriate positions in WORK() as described in Section B.1.b. Option 6 must be selected (see Section B.3). For simple usage

IOPT(2) = 6
IOPT(3) = 0

CALL SINTM (NDIMI, ANSWER, WORK, NWORK, IOPT)

DO

CALL SINTMA (ANSWER, WORK, IOPT)

```
   IF (IOPT(1) .GT. 0) THEN
      {Calculate the limits as described in
         Section B.1.d when IFLAG > 0.}
   ELSE IF (IOPT(1) .LT. 0) THEN
      IF (IOPT(1) + NDIMI .LE. 0) EXIT
      {Transform the integral as described in
         Section B.1.d when IFLAG < 0.}
   ELSE
      ANSWER = value of innermost integrand,
         f1, at (WORK(1), ..., WORK (NDIMI)).
   END IF
END DO
{Integration is complete.}
```

Multi-dimensional quadrature can use significant amounts of computer time. The usage described above can be modified to reduce the execution time slightly at a cost of the user's code becoming more complicated. To do this, replace the single line

ANSWER = ...

by

```
IOPT(1) = 0
DO WHILE (IOPT(1) .EQ. 0)
   ANSWER = Value of the innermost
      integrand, f1, at (WORK(1), ...,
      WORK(NDIMI))

   CALL SINTA (ANSWER, WORK, IOPT)

   END DO
   IF (IOPT(1) .GT. 0) THEN
      {Values of IOPT(1) produced by SINTMA and
         SINTA have different meanings. Calculate
         IOPT(1) as described in Section B.1.b.}
      IOPT(1) = -(IOPT(1) + NDIMI)
      EXIT  (from DO — END DO in which this
               code is embedded)
   END IF
```

This modification reduces by one the number of subroutine calls for every evaluation of the innermost integrand.

B.2.b Argument Definitions

NDIMI, WORK(), NWORK, IOPT() Used as described in Section B.1.b, except if $-\text{NDIMI} < \text{IOPT}(1) \leq \text{NDIMI}$, IOPT(1) is used as described for IFLAG in Section B.1.e.

ANSWER [inout] On completion, ANSWER contains an estimate of the integral. During integration, ANSWER provides a value of the integrand to SINTMA.

B.3 Methods to Request Unusual Usage Through the Arguments IOPT and WORK

All options other than option 13 have the same qualitative effect as when calculating an integral over one dimension, as described in Section B.3 of Chapter 13.1, but some options have separate effects in separate dimensions. Direct calls to SINTOP should be made as described in Section B.4 of Chapter 13.1. Options different from those in 13.1 are:

2 (Argument K2) K2 is an NDIMI decimal digit integer, where the low order digit selects the level of diagnostic output during integration over x_1, the next digit selects the level of diagnostic output during integration over x_2, etc. The meaning of each digit is the same as the meaning of K2 described in Section B.3 of Chapter 13.1.

9 (Argument K9) K9 specifies only the maximum number of evaluations of the inner integrand (f_1).

11 (Argument K11) WORK(|K11|) specifies the location of a singularity or discontinuity in the outer dimension. Singularities or discontinuities in inner dimensions may be specified as described in Section B.1.e. In either case, a different value of K11 must be used for each different abscissa of a singularity or discontinuity. However if several dimensions happen to have singularities or discontinuities at the same abscissa, the same value of K11 can be used for all of them. The sign of K11 affects the internal transformations made to cope with the singularity as described in Section B.3 of Chapter 13.1.

12 (Argument K12) The errors in the lower and upper limits are stored in WORK(K12) and WORK(K12+1) respectively prior to a call to SINTOP. The error in the lower limit a_i is defined by $\Sigma\,|\partial a_i/\partial x_j|\,\epsilon\,x_j + \Sigma\,|\partial a_i/\partial y_j|\,E(y_j) + \Sigma\|\partial a_i/\partial I_j|\,E_j$, where x_j are variables of integration of outer integrals, ϵ is the round-off level for the appropriate precision, y_j are arguments of a_i that are neither variables of integration of outer integrals nor integrals, $\text{E}(y_j)$ is the error in y_j, I_j are integrals that are arguments of a_i, and E_j is the error in I_j. The error in the integral due to error in the limit is the error in the limit times the integrand evaluated near the limit. The error in the upper limit is calculated similarly. If this option is not selected, or K12 = 0, then the limits will be assumed to be exact.

13 (Argument K13) In the IOPT vector passed to SINTM, K13 may be used to notify SINTM of nonstandard changes in the dimensionality of integration. If option 13 is not selected, IOPT(1) may be used for this purpose. But notice that reporting singularities might also use IOPT(1).

B.4 Changing the Selection of Some Options During the Computation

The method of changing the selection of options 1, 2, 6, 7, 9, 12 and 13 during the integration is described in Chapter 13.1, Section B.4.

B.5 Modifications for Double Precision Usage

For double precision usage, change all REAL type statements to DOUBLE PRECISION and change the prefix of all subroutine names from SINT to DINT.

C. Examples and Remarks

See DRSINTMF and ODSINTMF, or DRSINTMR and ODSINTMR for an example of the use of SINTM to compute

$$\int_0^{\pi} \int_0^{y} \frac{x \cos y}{x^2 + y^2}\, dx\, dy = 0.$$

The difference between DRSINTMF and DRSINTMR is that DRSINTMF uses forward communication, while DRSINTMR uses reverse communication.

DRSINTMF and DRSINTMR demonstrate the use of functional transformation of the inner integrand to reduce the cost of calculating the integral. The factor $\cos y$ in the integrand does not depend on the inner variable of integration. In this context, $f_1 = x\ /(x^2 + y^2)$, and $f_2 = I_1 \cos\ y$. Special care is taken when y is near zero, as the denominator of f_1 might underflow.

The above integral is really the product of two one-dimensional integrals (let $z = x/y$). This situation is not uncommon. Since the cost of estimating multi-dimensional integrals by nested estimation over one dimension is exponential in the number of dimensions, the possibility of this situation should always be considered.

D. Functional Description

The integral over several dimensions is computed by repeated integration over one dimension, as shown in Section A. The integral over one dimension is estimated using the subprograms described in Chapter 13.1. See Section D of Chapter 13.1 for a functional description.

Extensive test results are given in [1]. Since there are no other multi-dimensional quadrature subprograms extant that provide the functionality of SINTM and DINTM, it is not practical to carry out extensive comparative tests. Since SINT1 and DINT1 are, however, more reliable and require fewer function values than other one-dimensional quadrature routines, it is to be expected that SINTM and DINTM are more reliable and require fewer function values than other subprograms that evaluate multi-dimensional integrals by repeated integration over one dimension.

References

1. Fred T. Krogh and W. Van Snyder, **Preliminary Test Results for Multi-dimensional Quadrature**. Internal Computing Memorandum 442, Jet Propulsion Laboratory (July 1978).

E. Error Procedures and Restrictions

Error messages are printed using the extended error message processor described in Chapter 19.3. If an error does not result in a "stop," error signals are returned to the user by values of the status flag in IOPT(1). Printing, when enabled by Option 2, is executed in subroutines SINTO for single precision and DINTO for double precision. Both of these programs also use the message processor described in Chapter 19.3. One can change the action on errors and parameters affecting the output of messages, by calling the message/error routine MESS before calling this routine.

F. Supporting Information

The source language for these subroutines is ANSI Fortran 77.

Common blocks referenced: SINTC and SINTEC in the single precision versions, and DINTC and DINTEC in the double precision versions. SINTC and DINTC are written using several COMMON statements, for maintenance purposes. This usage conforms to the ANSI Fortran-77 standard, but at least one compiler interprets it improperly. If you experience inscrutable errors, try re-writing the COMMON statements for SINTC (or DINTC) into a single statement.

Entry	Required Files
DINTM	AMACH, DCOPY, DINTA, DINTDL, DINTDU, DINTF, DINTM, DINTMA, DINTNS, DINTO, DINTOP, DINTSM, DMESS, MESS
DINTMA	AMACH, DCOPY, DINTA, DINTDL, DINTDU, DINTF, DINTMA, DINTNS, DINTO, DINTSM, DMESS, MESS
SINTM	AMACH, MESS, SCOPY, SINTA, SINTDL, SINTDU, SINTF, SINTM, SINTMA, SINTNS, SINTO, SINTOP, SINTSM, SMESS
SINTMA	AMACH, MESS, SCOPY, SINTA, SINTDL, SINTDU, SINTF, SINTMA, SINTNS, SINTO, SINTSM, SMESS

Designed by Fred T. Krogh and W. Van Snyder, JPL, 1977. Programmed by W. Van Snyder, 1977. Revised by W. Van Snyder, 1986, 1988.

Error/Message handling revised by F. T. Krogh, March 1992.

Two demonstration drivers are shown below, with the output they produced when run on an IBM PC/AT with an 80287 floating point coprocessor. The first demonstrates forward communication usage; the second demonstrates reverse communication usage.

DRSINTMF

```
c      DRSINTM
c>> 2003-03-30 DRSINTMF Krogh Changed way of dealing with 0 denominator.
c>> 2001-05-22 DRSINTMF Krogh Minor change for making .f90 version.
c>> 1994-11-02 DRSINTMF Krogh  Changes to use M77CON
c>> 1994-08-08 DRSINTMF Snyder  Took '0' out of formats for C conversion
c>> 1993-05-05 DRSINTMF Krogh   Adjusted to simplify conversion to C.
c>> 1987-12-09 DRSINTMF Snyder  Initial Code.
c--S replaces "?": DR?INTM, DR?INTMF, ?INTM, ?INTF
c
c      DEMO DRIVER for multi-dimensional quadrature subprogram SINTM.
c
c      Compute the integral for Y = 0.0 to Y = PI of
c              the integral for X = 0.0 to X = Y of
c              X * COS(Y) /(X*X+Y*Y).  The ANSWER should be zero.
c
c      The integrand and all limits are provided by the subprogram SINTF,
c      although the limits of the outer dimension, and the lower limit of
c      the inner dimension, being constants, could have been provided
c      here.
c
       integer NDIMI, NWPD, NWORK, IOPT(10)
       parameter (NDIMI=2, NWPD=217, NWORK=3*NDIMI+NWPD*(NDIMI-1))
       real              ANSWER, WORK(NWORK)
10     format (/
     1' DRSINTMF:'/
     2' Compute the integral for Y = 0.0 to Y = PI of'/
     3'          the integral for X = 0.0 to X = Y of'/
     4'          X * COS(Y) /(X*X+Y*Y).  The ANSWER should be zero.')
20     format (/' ANSWER =          ',G15.8/
     1          ' ERROR ESTIMATE =',G15.8/
     2          ' STATUS FLAG =    ',I3/
     3          ' FUNCTION VALUES (INNERMOST INTEGRALS) =',I6)
c
       print 10
       IOPT(2) = 10
       IOPT(3) = 0
       IOPT(4) = 0
       call SINTM (NDIMI, ANSWER, WORK, NWORK, IOPT)
       print 20, ANSWER, WORK(1), IOPT(1), IOPT(3)
       stop
       end
c    End of DRSINTMF

       subroutine SINTF (ANSWER, WORK, IFLAG)
c
c      Subroutine to provide integrand and limits for SINTM.
c
c      This sample subroutine demonstrates the use of functional
c      transformation of the inner integral to reduce the cost of the
c      overall computation.  The factor "cos y" does not depend on the
c      variable of integration for the inner integral, and therefore is
c      factored out.  Similarly, the term y*y in the denominator of the
c      inner integrand is pre-computed when the limits of the inner
c      integral are requested.
c
c      This example also demonstrates the necessity to cope with overflow
```

```
c      of the integrand, even though the integral is well behaved.
c
       real              ANSWER, WORK(*)
       integer IFLAG
       integer NDIMI
       real              COSY, DENOM, XDY, YSQ
       save COSY, YSQ
       data NDIMI /2/
c
       if (IFLAG .EQ. 0) then
c
c        IFLAG = 0, compute innermost integrand.
c
         DENOM = WORK(1) * WORK(1) + YSQ
         if (DENOM .NE. 0.0E0) then
c          This test will not detect overflow of the integrand if the
c          arithmetic underflows gradually.
           ANSWER = WORK(1) / DENOM
         else
c      Special care to avoid a zero denominator.
           XDY = WORK(1) / WORK(2)
           ANSWER = XDY / (WORK(2) * (1.E0 + XDY**2))
         end if
c
       else if (IFLAG .EQ. 1) then
c
c        Compute limits of inner dimension.
c
c    Set WORK(1) = COS(WORK(2)) = Partial derivative, with respect to
c    the integral over the inner dimension, of the transformation
c    applied when integration over the inner dimension is complete.
c    This is so sintm knows how much accuracy it needs for this integral.
c
         WORK(NDIMI+IFLAG) = 0.0E0
         WORK(2*NDIMI+IFLAG) = WORK(2)
         YSQ = WORK(2) * WORK(2)
         COSY = COS(WORK(2))
         WORK(1) = COSY
c
       else if (IFLAG .EQ. 2) then
c
c        Compute limits of outer dimension.
c
         WORK(NDIMI+IFLAG) = 0.0E0
         WORK(2*NDIMI+IFLAG) = 4.0E0*ATAN(1.0E0)
c
       else
c
c        IFLAG < 0, transform inner integrand.
c
         ANSWER = ANSWER * COSY
         WORK(1) = WORK(1) * COSY
c
       end if
c
       return
c
       end
```

ODSINTMF

```
DRSINTMF:
Compute the integral for Y = 0.0 to Y = PI of
        the integral for X = 0.0 to X = Y of
        X * COS(Y) /(X*X+Y*Y).  The ANSWER should be zero.

ANSWER =          -0.21448164E-07
ERROR ESTIMATE =  0.45539418E-05
STATUS FLAG =     -3
FUNCTION VALUES (INNERMOST INTEGRALS) =     75
```

DRSINTMR

```
c     program DRSINTMR
c>> 2003-03-30 DRSINTMR Krogh Changed way of dealing with 0 denominator.
c>> 2001-05-22 DRSINTMR Krogh Minor change for making .f90 version.
c>> 1994-10-19 DRSINTMR Krogh  Changes to use M77CON
c>> 1994-08-31 DRSINTMR Snyder  Moved formats for C conversion
c>> 1994-08-08 DRSINTMR Snyder  Took '0' out of formats for C conversion
c>> 1991-11-20 CLL Edited for Fortran 90.
c>> 1987-12-09 DRSINTMR Snyder  Initial Code.
c--S replaces "?": DR?INTMR, ?INTM, ?INTMA, ?INTA
c
c     DEMO DRIVER for multi-dimensional quadrature subprogram SINTM.
c
c     Compute the integral for Y = 0.0 to Y = PI of
c             the integral for X = 0.0 to X = Y of
c             X * COS(Y) /(X*X+Y*Y).  The ANSWER should be zero.
c
c     The integrand and all limits are provided by reverse
c     communication.
c
c     This sample demonstrates the use of functional transformation of
c     the inner integral to reduce the cost of the overall computation.
c     The factor "cos y" does not depend on the variable of integration
c     for the inner integral, and therefore is factored out.  Similarly,
c     the term y*y in the denominator of the inner integrand is pre-
c     computed when the limits of the inner integral are requested.
c
c     This example also demonstrates the necessity to cope with overflow
c     of the integrand, even though the integral is well behaved.
c
      integer IFLAG, IOPT(10), NDIMI, NWPD, NWORK
      parameter (NDIMI=2, NWPD=217, NWORK=3*NDIMI+NWPD*(NDIMI-1))
      real              ANSWER, WORK(NWORK)
      real              COSY, DENOM, XDY, YSQ
c
10    format (/' DRSINTMR:'/
     1' Compute the integral for Y = 0.0 to Y = PI of'/
     2'         the integral for X = 0.0 to X = Y of'/
     3'         X * COS(Y) /(X*X+Y*Y).  The ANSWER should be zero.')
20    format (/' ANSWER =          ',G15.8/
     1         ' ERROR ESTIMATE =',G15.8/
     2         ' STATUS FLAG =    ',I3/
     3         ' FUNCTION VALUES (INNERMOST INTEGRALS) =',I6)
c
      print 10
```

```
      IOPT(2) = 10
      IOPT(3) = 0
      IOPT(4) = 6
      IOPT(5) = 0
      call SINTM (NDIMI, ANSWER, WORK, NWORK, IOPT)
 30   continue
      call SINTMA (ANSWER, WORK, IOPT)
      IFLAG = IOPT(1)
      if (IFLAG .eq. 0) then
c
c     IFLAG = 0, compute innermost integrand.
c
c     Iterate on the innermost integrand by calling SINTA here.
c     The code would be slightly simpler, but slightly slower,
c     if control of the iteration were relegated to SINTMA.
c
 40      continue
         DENOM = WORK(1) * WORK(1) + YSQ
         if (DENOM .ne. 0.0E0) then
c     This test will not detect overflow of the integrand if
c     the arithmetic underflows gradually.
            ANSWER = WORK(1) / DENOM
         else
c     Special care to avoid a zero denominator.
            XDY = WORK(1) / WORK(2)
            ANSWER = XDY / (WORK(2) * (1.E0 + XDY**2))
         end if
         call SINTA (ANSWER, WORK, IOPT)
         if(IOPT(1) .eq. 0) goto 40
c
         if (IOPT(1) .gt. 0) then
c                                  Done with the integration
            IOPT(1) = -(IOPT(1)+NDIMI)
            go to 60
         end if
c
      else if (IFLAG .eq. 1) then
c
c     Compute limits of inner dimension.
c
c   Set WORK(1) = COS(WORK(2)) = Partial derivative, with respect to
c   the integral over the inner dimension, of the transformation
c   applied when integration over the inner dimension is complete.
c   This is so sintm knows how much accuracy it needs for this integral.
c
         WORK(NDIMI+IFLAG) = 0.0E0
         WORK(2*NDIMI+IFLAG) = WORK(2)
         YSQ = WORK(2) * WORK(2)
         COSY = COS(WORK(2))
         WORK(1) = COSY
c
      else if (IFLAG .eq. 2) then
c
c     Compute limits of outer dimension.
c
         WORK(NDIMI+IFLAG) = 0.0E0
         WORK(2*NDIMI+IFLAG) = 4.0E0*ATAN(1.0E0)
c
      else
```

```
         if (IFLAG+NDIMI .le. 0) goto 60
c
c      IFLAG < 0, transform inner integrand.
c
         ANSWER = ANSWER * COSY
         WORK(1) = WORK(1) * COSY
c
      end if
      go to 30
c
60    continue

      print 20, ANSWER, WORK(1), IOPT(1), IOPT(3)
      stop
      end
```

ODSINTMR

```
DRSINTMR:
Compute the integral for Y = 0.0 to Y = PI of
        the integral for X = 0.0 to X = Y of
        X * COS(Y) /(X*X+Y*Y).  The ANSWER should be zero.

ANSWER =          -0.21448164E-07
ERROR ESTIMATE =  0.45539418E-05
STATUS FLAG =     -3
FUNCTION VALUES (INNERMOST INTEGRALS) =     75
```