NAG Library Routine Document

D01RBF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

Note: please be advised that D01RBF has been deprecated. You are advised to follow the directions in Section 9.1 in D01RAF to replicate its functionality. Furthermore, any future diagnostic enhancements for D01RAF will only be available in this manner.

1 Purpose

D01RBF is an expert diagnostic routine associated with D01RAF to allow the arrays ICOM and COM to be examined. These arrays contain details of the adaptive process. Facilities are provided, via the argument MONIT, to refine the final answer by adjusting the contribution made by specific segments.

2 Specification

```
SUBROUTINE D01RBF (MONIT, NI, DINEST, ERREST, ICOM, LICOM, LICUSD, COM,    &
                   LCOM, LCUSD, IUSER, RUSER, IFAIL)
INTEGER            NI, ICOM(LICOM), LICOM, LICUSD, LCOM, LCUSD,            &
                   IUSER(*), IFAIL
REAL (KIND=nag_wp) DINEST(NI), ERREST(NI), COM(LCOM), RUSER(*)
EXTERNAL           MONIT
```

3 Description

The principal role of D01RBF is to take information about the adaptive process that is stored in COM and ICOM and present it in more intelligible arguments. A monitor routine, MONIT, allows you to report the progress of the adaptive process back to the calling program between those calls of D01RAF which have returned IREVCM = 11 or 12.

A secondary role, useful only if you are an expert, is to utilize MONIT in such a way that it can override aspects of that information. Specifically, you may choose to remove the contributions of one or more individual segments from the estimates for individual integrals contained in DINEST and ERREST, and replace such information with a more accurate approximation you have calculated by some other means. Clearly this facility should only be used with care. We recommend that it be used only after D01RAF has returned with IREVCM = 0, i.e., it has completed the computation of the approximations of the integrals.

4 References

None.

5 Arguments

1: MONIT – SUBROUTINE, supplied by the NAG Library or the user. *External Procedure*

MONIT allows you to examine details of the adaptive process after a call of D01RAF returning IREVCM = 11 or 12. Additionally, after a call of D01RAF returning IREVCM = 0, MONIT may enhance the computed solution.

If no monitoring is required, MONIT may be the dummy monitoring routine D01RBM. (D01RBM is included in the NAG Library.)

The specification of MONIT is:

```
SUBROUTINE MONIT (NI, NS, DINEST, ERREST, FCOUNT, SINFOI, EVALS,     &
                  LDI, SINFOR, FS, ES, LDR, IUSER, RUSER)
INTEGER            NI, NS, FCOUNT(NI), SINFOI(LDI,*),                &
                   EVALS(LDI,*), LDI, LDR, IUSER(*)
REAL (KIND=nag_wp) DINEST(NI), ERREST(NI), SINFOR(LDR,*),            &
                   FS(LDR,*), ES(LDR,*), RUSER(*)
```

1: NI – INTEGER *Input*

On entry: n_i, the number of integrands specified in D01RAF.

2: NS – INTEGER *Input*

On entry: n_s, the number of segments formed during the adaptive procedure that can currently be examined. $n_s = 2n_{sdiv} + s_{pri}$, where n_{sdiv} is the number of segments that have been subdivided (the number of subdivisions).

3: DINEST(NI) – REAL (KIND=nag_wp) array *Input/Output*

On entry: DINEST(j) contains the current estimate of the definite integral of integrand j, for $j = 1, 2, \ldots, n_i$.

On exit: you may refine the estimates in DINEST. This should only be done after D01RAF has returned IREVCM $= 0$.

4: ERREST(NI) – REAL (KIND=nag_wp) array *Input/Output*

On entry: ERREST(j) contains the current error estimate associated with integral j, for $j = 1, 2, \ldots, n_i$.

On exit: you may refine the estimates in ERREST. This should only be done after D01RAF has returned IREVCM $= 0$.

5: FCOUNT(NI) – INTEGER array *Input*

On entry: FCOUNT(j) contains the number of different approximations of integral j calculated so far, for $j = 1, 2, \ldots, n_i$.

6: SINFOI(LDI, $*$) – INTEGER array *Input*

On entry: SINFOI(l, k) contains information about the hierarchy of splitting, for $l = 1, 2, \ldots, 5$ and $k = 1, 2, \ldots, n_s$.

SINFOI($1, k$)
Contains a unique identifier, the SID, related to segment k.

SINFOI($2, k$)
Contains the parent segment number of segment k, the reference to the segment that was split to create segment k.

SINFOI($3, k$) and SINFOI($4, k$)
Contain the segment numbers of the two child segments formed from segment k, if segment k has been split. If segment k has not been split, these will be negative.

SINFOI($5, k$)
Contains the level at which the segment exists, corresponding to $n_a + 1$, where n_a is the number of ancestor segments of segment k.

If SINFOI($5, k$) < 0, this segment is considered too small for further splitting, and its level is $|$SINFOI($5, k$)$|$.

7: EVALS(LDI, $*$) – INTEGER array *Input*

On entry: contains information to indicate whether an estimate of the integral j has been obtained over segment k, and if so whether this evaluation still contributes to the approximation in DINEST(j), for $j = 1, 2, \ldots, n_i$ and $k = 1, 2, \ldots, n_s$.

EVALS$(j, k) = 0$
Indicates that integral j has not been evaluated over segment k.

EVALS$(j, k) = 1$
Indicates that integral j has been evaluated over segment k, and that this evaluation contributes to the total approximation DINEST(j).

EVALS$(j, k) = 2$
Indicates that integral j has been evaluated over segment k, that this evaluation contributes to the total approximation DINEST(j), and that you have requested no further evaluation of this integral at this segment by setting NEEDI$(j) < 0$.

EVALS$(j, k) = 3$
Indicates that integral j has been evaluated over segment k, and this evaluation no longer contributes to DINEST(j).

EVALS$(j, k) = 4$
Indicates that integral j has been evaluated over segment k, that this evaluation contributes to the total approximation DINEST(j), and that this segment is too small for any further splitting to be performed. Integral j also has a local error estimate over this segment above the requested tolerance. Such segments cause D01RAF to return IFAIL $= 2$ or 3, indicating extremely bad behaviour.

EVALS$(j, k) = 5$
Indicates that integral j has been evaluated over segment k, that this evaluation contributes to the total approximation of DINEST(j), and that this segment is too small for any further splitting to be performed. The local error estimate is however below the requested tolerance.

8: LDI – INTEGER *Input*

On entry: the leading dimension of arrays SINFOI and EVALS.

9: SINFOR(LDR, $*$) – REAL (KIND=nag_wp) array *Input*

On entry: SINFOR contains the bounds of each segment k, for $k = 1, 2, \ldots, n_s$.

SINFOR$(1, k)$
Contains the lower bound of segment k.

SINFOR$(2, k)$
Contains the upper bound of segment k.

10: FS(LDR, $*$) – REAL (KIND=nag_wp) array *Input*

On entry: FS(j, k) contains the definite integral estimate of the jth integral over the kth segment, for $j = 1, 2, \ldots, n_i$ and $k = 1, 2, \ldots, n_s$.

11: ES(LDR, $*$) – REAL (KIND=nag_wp) array *Input*

On entry: ES(j, k) contains the definite integral error estimate of the jth integral over the kth segment, for $j = 1, 2, \ldots, n_i$ and $k = 1, 2, \ldots, n_s$.

12: LDR – INTEGER *Input*

On entry: the leading dimension of arrays SINFOR, FS and ES.

13: IUSER(∗) – INTEGER array *User Workspace*
14: RUSER(∗) – REAL (KIND=nag_wp) array *User Workspace*

MONIT is called with the arguments IUSER and RUSER as supplied to D01RBF. You should use the arrays IUSER and RUSER to supply information to MONIT.

MONIT must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which D01RBF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

2: NI – INTEGER *Input*

On entry: n_i, the number of integrals.

3: DINEST(NI) – REAL (KIND=nag_wp) array *Input/Output*

On entry: DINEST(j) contains the current estimate of the definite integral of integrand j, for $j = 1, 2, \ldots, n_i$.

On exit: DINEST is not altered by D01RBF directly. It may be changed by MONIT.

4: ERREST(NI) – REAL (KIND=nag_wp) array *Input/Output*

On entry: ERREST(j) contains the current error estimate associated with integral j, for $j = 1, 2, \ldots, n_i$.

On exit: ERREST is not altered by D01RBF directly. It may be changed by MONIT.

5: ICOM(LICOM) – INTEGER array *Communication Array*

On entry: the elements of this array **must not** be changed since the call of D01RAF.

6: LICOM – INTEGER *Input*

On entry: the dimension of the array ICOM as declared in the (sub)program from which D01RBF is called. Normally this will be the same value LICOM passed to D01RAF.

Constraint: LICOM ≥ 50.

Note: if LICOM $<$ LICUSD, (i.e., LICOM is less than the value passed to D01RAF) not all segments may be investigated using MONIT (see argument NS) and D01RBF will return IFAIL $= 1$.

7: LICUSD – INTEGER *Output*

On exit: the number of elements of the array ICOM, passed to D01RAF, actually used.

8: COM(LCOM) – REAL (KIND=nag_wp) array *Communication Array*

On entry: the elements of this array **must not** be changed since the call of D01RAF.

9: LCOM – INTEGER *Input*

On entry: the dimension of the array COM as declared in the (sub)program from which D01RBF is called. Normally this will be the same value LCOM passed to D01RAF.

Constraint: LCOM ≥ 50.

Note: if LCOM $<$ LCUSD, (i.e., LCOM is less than the value passed to D01RAF) not all segments may be investigated using MONIT (see argument NS) and D01RBF will return IFAIL $= 1$.

10: LCUSD – INTEGER *Output*

On exit: the number of elements of COM used by D01RAF.

11: IUSER(∗) – INTEGER array *User Workspace*
12: RUSER(∗) – REAL (KIND=nag_wp) array *User Workspace*

IUSER and RUSER are not used by D01RBF, but are passed directly to MONIT and should be used to pass information to this routine.

13: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value −1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, because for this routine the values of the output arguments may be useful even if IFAIL $\neq 0$ on exit, the recommended value is −1. **When the value −1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Note: D01RBF may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, LICOM $= \langle value \rangle$, LCOM $= \langle value \rangle$.
LICOM or LCOM is insufficient for a complete examination of the communication arrays. A maximum $\langle value \rangle$ segments out of $\langle value \rangle$ are examinable.
Constraint: LICOM $\geq \langle value \rangle$ and LCOM $\geq \langle value \rangle$.

IFAIL $= 21$

On entry, NI is not consistent with that used to construct the communication arrays.
On entry, NI $= \langle value \rangle$.
Constraint: NI $= \langle value \rangle$.

IFAIL $= 61$

On entry, LICOM < 50.

IFAIL $= 91$

On entry, LCOM < 50.

IFAIL $= 1101$

Either the communication arrays have not been created by D01RAF, or they have become corrupted.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

Not applicable.

8 Parallelism and Performance

D01RBF is not threaded in any implementation.

9 Further Comments

When called after D01RAF has returned IREVCM $= 0$, D01RBF may be used to modify DINEST and ERREST. For example if EVALS$(j, \bar{k}) = 4$ for some $j \in 1, \ldots, n_i$, $\bar{k} \in 1, \ldots, n_s$, indicating integral j was badly behaved over segment $\bar{k}$, then one may potentially modify DINEST(j) as follows:

$a_{\bar{k}} =$ SINFOR(l, k)

$b_{\bar{k}} =$ SINFOR(l, k)

$F_{\bar{k}}^{\text{new}} = \int_{a_{\bar{k}}}^{b_{\bar{k}}} f_j(x)dx$: performed using accurate technique for specific function.

DINEST(j) = DINEST(j) − FS$(j, \bar{k})$ + $F_{\bar{k}}^{\text{new}}$

ERREST may be similarly updated if required.

Note: if integral j has been approximated successfully due to extrapolation, indicated by NEEDI$(j) = 1$ after D01RAF has completed, DINEST(j) will not be the direct sum of the contributing segments. You may however reconstruct the unextrapolated integral estimate as,

$$\text{DINEST}(j) = \sum_{K} \text{FS}(j, k) + \sum_{\bar{K}} \text{FS}(j, \bar{k}),$$

where $K = \{k \mid \text{EVALS}(\text{LDI} \times (k) + j) = 1, 2, 5\}$ and $\bar{K} = \left\{\bar{k} \mid \text{EVALS}(\text{LDI} \times \left(\bar{k}\right) + j) = 4\right\}$, the sets of all contributing segements where integral j has been evaluated accurately and inaccurately respectively. Some or all of the terms in the summation over $\bar{K}$ may be replaced with more accurate approximations $F_{\bar{k}}^{\text{new}}$ if available.

10 Example

See Section 10 in D01RAF.