# NAG Library Routine Document

# D01EAF

**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.

## 1 Purpose

D01EAF computes approximations to the integrals of a vector of similar functions, each defined over the same multidimensional hyper-rectangular region. The routine uses an adaptive subdivision strategy, and also computes absolute error estimates.

## 2 Specification

```
SUBROUTINE D01EAF (NDIM, A, B, MINCLS, MAXCLS, NFUN, FUNSUB, ABSREQ,       &
                   RELREQ, LENWRK, WRKSTR, FINEST, ABSEST, IFAIL)

INTEGER            NDIM, MINCLS, MAXCLS, NFUN, LENWRK, IFAIL
REAL (KIND=nag_wp) A(NDIM), B(NDIM), ABSREQ, RELREQ, WRKSTR(LENWRK),       &
                   FINEST(NFUN), ABSEST(NFUN)
EXTERNAL           FUNSUB
```

## 3 Description

D01EAF uses a globally adaptive method based on the algorithm described by van Dooren and de Ridder (1976) and Genz and Malik (1980). It is implemented for integrals in the form:

$$\int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n}(f_1,f_2,\ldots,f_m)\,dx_n\ldots dx_2dx_1,$$

where $f_i = f_i(x_1,x_2,\ldots,x_n)$, for $i=1,2,\ldots,m$.

Upon entry, unless MINCLS has been set to a value less than or equal to 0, D01EAF divides the integration region into a number of subregions with randomly selected volumes. Inside each subregion the integrals and their errors are estimated. The initial number of subregions is chosen to be as large as possible without using more than MINCLS calls to FUNSUB. The results are stored in a partially ordered list (a heap). The routine then proceeds in stages. At each stage the subregion with the largest error (measured using the maximum norm) is halved along the coordinate axis where the integrands have largest absolute fourth differences. The basic rule is applied to each half of this subregion and the results are stored in the list. The results from the two halves are used to update the global integral and error estimates (FINEST and ABSEST) and the routine continues unless $\|\mathrm{ABSEST}\| \leq \max(\mathrm{ABSREQ}, \|\mathrm{FINEST}\| \times \mathrm{RELREQ})$ where the norm $\|.\|$ is the maximum norm, or further subdivision would use more than MAXCLS calls to FUNSUB. If at some stage there is insufficient working storage to keep the results for the next subdivision, the routine switches to a less efficient mode; only if this mode of operation breaks down is insufficient storage reported.

## 4 References

Genz A C and Malik A A (1980) An adaptive algorithm for numerical integration over an N-dimensional rectangular region *J. Comput. Appl. Math.* **6** 295–302

van Dooren P and de Ridder L (1976) An adaptive algorithm for numerical integration over an N-dimensional cube *J. Comput. Appl. Math.* **2** 207–217

## 5 Parameters

1: NDIM – INTEGER *Input*

*On entry*: $n$, the number of dimensions of the integrals.

*Constraint*: NDIM $\geq 1$.

2: A(NDIM) – REAL (KIND=nag_wp) array *Input*

*On entry*: the lower limits of integration, $a_i$, for $i = 1, 2, \ldots, n$.

3: B(NDIM) – REAL (KIND=nag_wp) array *Input*

*On entry*: the upper limits of integration, $b_i$, for $i = 1, 2, \ldots, n$.

4: MINCLS – INTEGER *Input/Output*

*On entry*: must be set either to the minimum number of FUNSUB calls to be allowed, in which case MINCLS $\geq 0$ or to a negative value. In this case, the routine continues the calculation started in a previous call with the same integrands and integration limits: no parameters other than MINCLS, MAXCLS, ABSREQ, RELREQ or IFAIL must be changed between the calls.

*On exit*: gives the number of FUNSUB calls actually used by D01EAF. For the continuation case (MINCLS $< 0$ on entry) this is the number of new FUNSUB calls on the current call to D01EAF.

5: MAXCLS – INTEGER *Input*

*On entry*: the maximum number of FUNSUB calls to be allowed. In the continuation case this is the number of new FUNSUB calls to be allowed.

*Constraints*:

MAXCLS $\geq$ MINCLS;
MAXCLS $\geq r$;
where $r = 2^n + 2n^2 + 2n + 1$, if $n < 11$, or $r = 1 + n(4n^2 - 6n + 14)/3$, if $n \geq 11$.

6: NFUN – INTEGER *Input*

*On entry*: $m$, the number of integrands.

*Constraint*: NFUN $\geq 1$.

7: FUNSUB – SUBROUTINE, supplied by the user. *External Procedure*

FUNSUB must evaluate the integrands $f_i$ at a given point.

The specification of FUNSUB is:

```
SUBROUTINE FUNSUB (NDIM, Z, NFUN, F)

INTEGER             NDIM, NFUN
REAL (KIND=nag_wp) Z(NDIM), F(NFUN)
```

1: NDIM – INTEGER *Input*

*On entry*: $n$, the number of dimensions of the integrals.

2: Z(NDIM) – REAL (KIND=nag_wp) array *Input*

*On entry*: the coordinates of the point at which the integrands must be evaluated.

3: NFUN – INTEGER *Input*

*On entry*: $m$, the number of integrands.

4: F(NFUN) – REAL (KIND=nag_wp) array *Output*

*On exit*: the value of the $i$th integrand at the given point.

FUNSUB must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which D01EAF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

8: ABSREQ – REAL (KIND=nag_wp) *Input*

*On entry*: the absolute accuracy required by you.

*Constraint*: ABSREQ $\geq 0.0$.

9: RELREQ – REAL (KIND=nag_wp) *Input*

*On entry*: the relative accuracy required by you.

*Constraint*: RELREQ $\geq 0.0$.

10: LENWRK – INTEGER *Input*

*On entry*: the dimension of the array WRKSTR as declared in the (sub)program from which D01EAF is called.

*Suggested value*: LENWRK $\geq 6n + 9m + (n + m + 2)(1 + p/r)$, where $p$ is the value of MAXCLS and $r$ is defined under MAXCLS. If LENWRK is significantly smaller than this, the routine will not work as efficiently and may even fail.

*Constraint*: LENWRK $\geq 8 \times$ NDIM $+ 11 \times$ NFUN $+ 3$.

11: WRKSTR(LENWRK) – REAL (KIND=nag_wp) array *Input/Output*

*On entry*: if MINCLS $< 0$, WRKSTR must be unchanged from the previous call of D01EAF.

*On exit*: contains information about the current subdivision which could be used in a continuation call.

12: FINEST(NFUN) – REAL (KIND=nag_wp) array *Output*

*On exit*: FINEST$(i)$ specifies the best estimate obtained from the $i$th integral, for $i = 1, 2, \ldots, m$.

13: ABSEST(NFUN) – REAL (KIND=nag_wp) array *Output*

*On exit*: ABSEST$(i)$ specifies the estimated absolute accuracy of FINEST$(i)$, for $i = 1, 2, \ldots, m$.

14: IFAIL – INTEGER *Input/Output*

*On entry*: IFAIL must be set to 0, $-1$ or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction 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 parameters 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).

Errors or warnings detected by the routine:

IFAIL $= 1$

MAXCLS was too small for D01EAF to obtain the required accuracy. The arrays FINEST and ABSEST respectively contain current estimates for the integrals and errors.

IFAIL $= 2$

LENWRK is too small for the routine to continue. The arrays FINEST and ABSEST respectively contain current estimates for the integrals and errors.

IFAIL $= 3$

On a continuation call, MAXCLS was set too small to make any progress. Increase MAXCLS before calling D01EAF again.

IFAIL $= 4$

On entry, NDIM $< 1$,
or NFUN $< 1$,
or MAXCLS $<$ MINCLS,
or MAXCLS $< r$ (see MAXCLS),
or ABSREQ $< 0.0$,
or RELREQ $< 0.0$,
or LENWRK $< 8 \times \mathrm{NDIM} + 11 \times \mathrm{NFUN} + 3$.

IFAIL $= -99$

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

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

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

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

## 7 Accuracy

An absolute error estimate for each integrand is output in the array ABSEST. The routine exits with IFAIL $= 0$ if

$$\max_i(\mathrm{ABSEST}(i)) \leq \max\left(\mathrm{ABSREQ}, \mathrm{RELREQ} \times \max_i|\mathrm{FINEST}(i)|\right).$$

## 8 Parallelism and Performance

D01EAF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

## 9 Further Comments

Usually the running time for D01EAF will be dominated by the time in FUNSUB, so the maximum time that could be used by D01EAF will be proportional to MAXCLS multiplied by the cost of a call to FUNSUB.

On a normal call, you should set MINCLS $= 0$ on entry.

For some integrands, particularly those that are poorly behaved in a small part of the integration region, D01EAF may terminate prematurely with values of ABSEST that are significantly smaller than the actual absolute errors. This behaviour should be suspected if the returned value of MINCLS is small relative to the expected difficulty of the integrals. When this occurs D01EAF should be called again, but with an entry value of MINCLS $\geq 2r$, (see specification of MAXCLS) and the results compared with those from the previous call.

If the routine is called with MINCLS $\geq 2r$, the exact values of FINEST and ABSEST on return will depend (within statistical limits) on the sequence of random numbers generated internally within D01EAF by calls to G05SAF. Separate runs will produce identical answers unless the part of the program executed prior to calling D01EAF also calls (directly or indirectly) routines from Chapter G05, and, in addition, the series of such calls differs between runs.

Because of moderate instability in the application of the basic integration rule, approximately the last $1 + \log_{10}(n^3)$ decimal digits may be inaccurate when using D01EAF for large values of $n$.

## 10 Example

This example computes

$$\int_0^1 \int_0^1 \int_0^1 \int_0^1 (f_1, f_2, \ldots, f_{10})\, dx_4\, dx_3\, dx_2\, dx_1,$$

where $j = 1, 2, \ldots, 10$, $f_j = \ln(x_1 + 2x_2 + 3x_3 + 4x_4)\sin(j + x_1 + 2x_2 + 3x_3 + 4x_4)$. The program is intended to show how to exploit the continuation facility provided with D01EAF: the routine exits with IFAIL $= 1$ (printing an explanatory error message) and is re-entered with MAXCLS reset to a larger value. The program can be used with any values of NDIM and NFUN, except that the expression for $r$ must be changed if NDIM $> 10$ (see specification of MAXCLS).

### 10.1 Program Text

```
!   D01EAF Example Program Text
!   Mark 25 Release. NAG Copyright 2014.

    Module d01eafe_mod

!     D01EAF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                              :: funsub
!     .. Parameters ..
      Integer, Parameter                  :: mulcls = 1
      Integer, Parameter, Public          :: ndim = 4, nfun = 10, nout = 6
      Integer, Parameter                  :: ircls = 2**ndim + 2*ndim*(ndim + &
                                             1) + 1
      Integer, Parameter, Public          ::                               &
```

```
                                              lenwrk = (ndim+nfun+2)*(10+mulcls)
      Integer, Parameter, Public           :: mxcls = mulcls*ircls
    Contains
      Subroutine funsub(ndim,z,nfun,f)

!       .. Scalar Arguments ..
        Integer, Intent (In)                 :: ndim, nfun
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out)     :: f(nfun)
        Real (Kind=nag_wp), Intent (In)      :: z(ndim)
!       .. Local Scalars ..
        Real (Kind=nag_wp)                   :: sum
        Integer                              :: i, n
!       .. Intrinsic Procedures ..
        Intrinsic                            :: log, real, sin
!       .. Executable Statements ..
        sum = 0.0E0_nag_wp

        Do n = 1, ndim
          sum = sum + real(n,kind=nag_wp)*z(n)
        End Do

        Do i = 1, nfun
          f(i) = log(sum)*sin(real(i,kind=nag_wp)+sum)
        End Do

        Return

      End Subroutine funsub
    End Module d01eafe_mod
    Program d01eafe

!     D01EAF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d01eaf, nag_wp
      Use d01eafe_mod, Only: funsub, lenwrk, mxcls, ndim, nfun, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                   :: absreq, relreq
      Integer                              :: i, ifail, maxcls, mincls, mulfac
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable      :: a(:), absest(:), b(:),           &
                                              finest(:), wrkstr(:)
!     .. Executable Statements ..
      Write (nout,*) 'D01EAF Example Program Results'
      Flush (nout)

      Allocate (a(ndim),absest(nfun),b(ndim),finest(nfun),wrkstr(lenwrk))

      a(1:ndim) = 0.0_nag_wp
      b(1:ndim) = 1.0_nag_wp
      mincls = 0
      maxcls = mxcls
      absreq = 0.0_nag_wp
      relreq = 1.0E-3_nag_wp

      If (ndim<=10) Then
        mulfac = 2**ndim
      Else
        mulfac = 2*ndim**3
      End If

loop: Do

        ifail = -1
        Call d01eaf(ndim,a,b,mincls,maxcls,nfun,funsub,absreq,relreq,lenwrk, &
          wrkstr,finest,absest,ifail)

        Select Case (ifail)
        Case (1,3)
          Write (nout,*)
          Write (nout,99999) mincls
          Write (nout,99998)

          Do i = 1, nfun
            Write (nout,99997) i, finest(i), absest(i)
          End Do

          Write (nout,*)
          Flush (nout)
          mincls = -1
          maxcls = maxcls*mulfac
        Case (0)
          Write (nout,*)
          Write (nout,99996) mincls
          Write (nout,99998)

          Do i = 1, nfun
            Write (nout,99997) i, finest(i), absest(i)
          End Do

          Exit loop
        Case Default
          Exit loop
        End Select

      End Do loop

99999 Format (1X,'Results so far (',I7,' FUNSUB calls in last call of D01EAF)' &
        )
99998 Format (/1X,'   I       Integral   Estimated error')
99997 Format (1X,I4,2F14.4)
99996 Format (1X,'Final Results (',I7,' FUNSUB calls in last call of D01EAF)')
    End Program d01eafe
```

## 10.2 Program Data

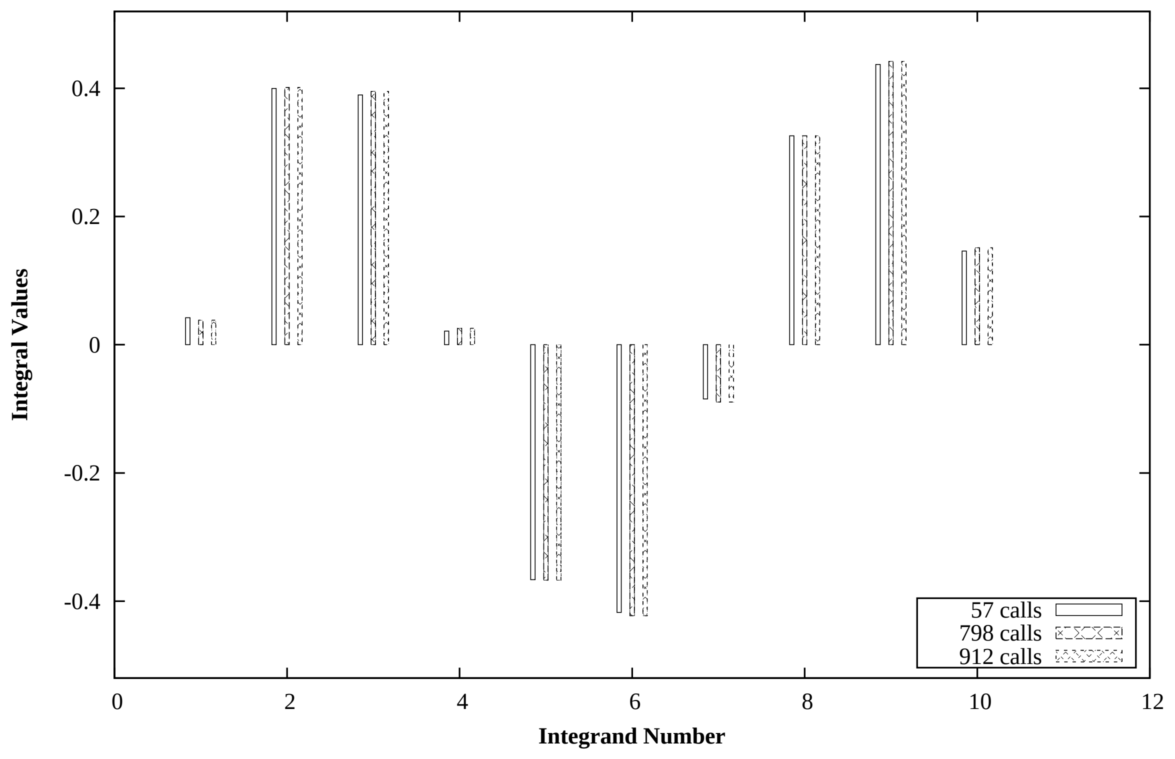
None.

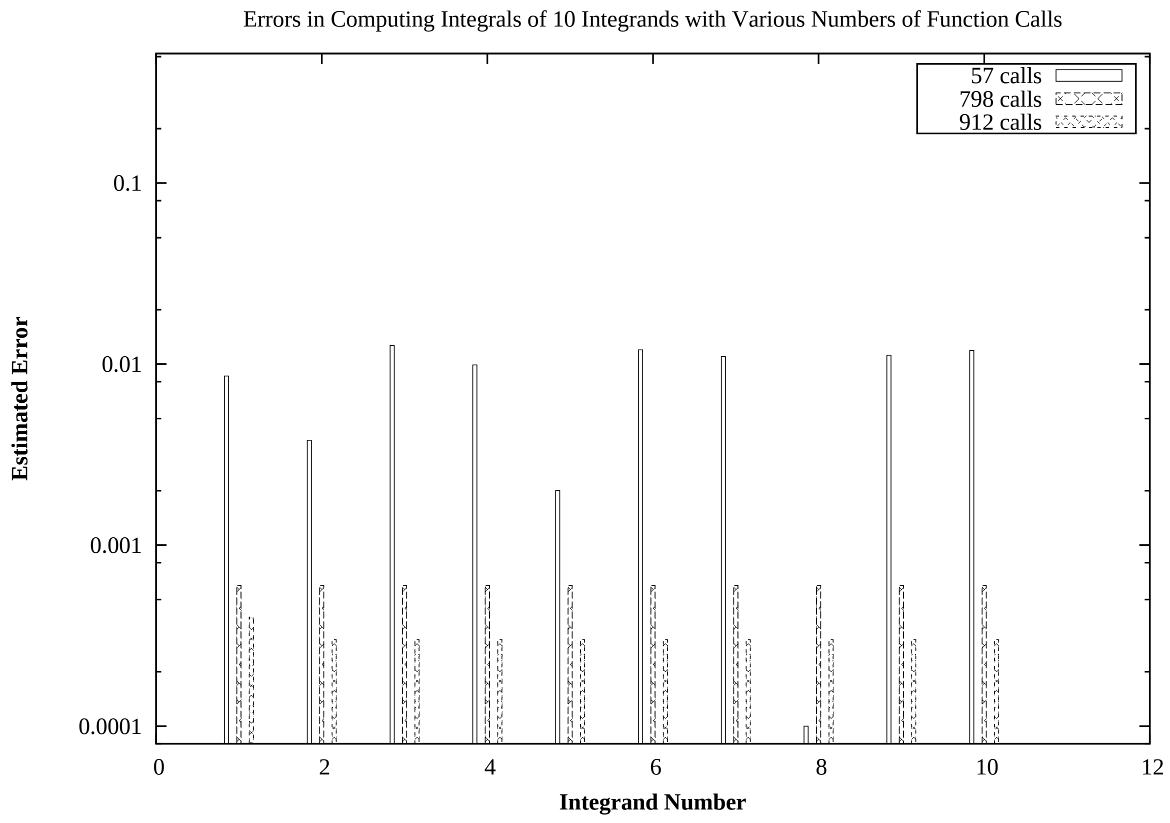
## 10.3 Program Results

```
 D01EAF Example Program Results
 ** MAXCLS too small to obtain required accuracy.
 ** MAXCLS =                     57.
 ** ABNORMAL EXIT from NAG Library routine D01EAF: IFAIL =     1
 ** NAG soft failure - control returned

 Results so far (     57 FUNSUB calls in last call of D01EAF)

    I       Integral   Estimated error
    1        0.0422        0.0086
    2        0.3998        0.0038
    3        0.3898        0.0127
    4        0.0214        0.0099
    5       -0.3666        0.0020
    6       -0.4176        0.0120
    7       -0.0846        0.0110
    8        0.3261        0.0001
    9        0.4371        0.0112
   10        0.1461        0.0119

 ** MAXCLS too small to obtain required accuracy.
 ** MAXCLS =                    912.
 ** ABNORMAL EXIT from NAG Library routine D01EAF: IFAIL =     1
 ** NAG soft failure - control returned

 Results so far (    798 FUNSUB calls in last call of D01EAF)

    I       Integral   Estimated error
```

```
     1        0.0384        0.0006
     2        0.4012        0.0006
     3        0.3952        0.0006
     4        0.0258        0.0006
     5       -0.3673        0.0006
     6       -0.4227        0.0006
     7       -0.0895        0.0006
     8        0.3260        0.0006
     9        0.4417        0.0006
    10        0.1514        0.0006

 Final Results (    912 FUNSUB calls in last call of D01EAF)

     I       Integral   Estimated error
     1        0.0384        0.0004
     2        0.4012        0.0003
     3        0.3952        0.0003
     4        0.0258        0.0003
     5       -0.3672        0.0003
     6       -0.4227        0.0003
     7       -0.0895        0.0003
     8        0.3260        0.0003
     9        0.4417        0.0003
    10        0.1514        0.0003
```

**Example Program**
Multi-dimensional Integrals of 10 Integrands with Various Numbers of Function Calls




Errors in Computing Integrals of 10 Integrands with Various Numbers of Function Calls
57 calls
798 calls
912 calls
0.1
0.01
0.001
0.0001
Estimated Error
0
2
4
6
8
10
12
Integrand Number