NAG Toolbox for MATLAB

d01al

1 Purpose

d01al is a general purpose integrator which calculates an approximation to the integral of a function $f(x)$ over a finite interval $[a, b]$:

$$I = \int_a^b f(x)\, dx$$

where the integrand may have local singular behaviour at a finite number of points within the integration interval.

2 Syntax

```
[result, abserr, w, iw, ifail] = d01al(f, a, b, points, epsabs, epsrel,
'npts', npts, 'lw', lw, 'liw', liw)
```

3 Description

d01al is based on the QUADPACK routine QAGP (see Piessens *et al.* 1983). It is very similar to d01aj, but allows you to supply 'break points', points at which the integrand is known to be difficult. It employs an adaptive algorithm, using the Gauss 10-point and Kronrod 21-point rules. The algorithm, described in de Doncker 1978, incorporates a global acceptance criterion (as defined by Malcolm and Simpson 1976) together with the ϵ-algorithm (see Wynn 1956) to perform extrapolation. The user-supplied 'break points' always occur as the end points of some sub-interval during the adaptive process. The local error estimation is described in Piessens *et al.* 1983.

4 References

de Doncker E 1978 An adaptive extrapolation algorithm for automatic integration *ACM SIGNUM Newsl.* **13 (2)** 12–18

Malcolm M A and Simpson R B 1976 Local versus global strategies for adaptive quadrature *ACM Trans. Math. Software* **1** 129–146

Piessens R, de Doncker–Kapenga E, Überhuber C and Kahaner D 1983 *QUADPACK, A Subroutine Package for Automatic Integration* Springer–Verlag

Wynn P 1956 On a device for computing the $e_m(S_n)$ transformation *Math. Tables Aids Comput.* **10** 91–96

5 Parameters

5.1 Compulsory Input Parameters

1: **f – string containing name of m-file**

f must return the value of the integrand f at a given point.

Its specification is:

```
[result] = f(x)
```

Input Parameters

1: **x – double scalar**

The point at which the integrand f must be evaluated.

Output Parameters

1: **result – double scalar**

The result of the function.

2: **a – double scalar**

a, the lower limit of integration.

3: **b – double scalar**

b, the upper limit of integration. It is not necessary that $a < b$.

4: **points**$(*)$ **– double array**

Note: the dimension of the array **points** must be at least $\max(1, \mathbf{npts})$.

The user-specified break points.

Constraint: the break points must all lie within the interval of integration (but may be supplied in any order).

5: **epsabs – double scalar**

The absolute accuracy required. If **epsabs** is negative, the absolute value is used. See Section 7.

6: **epsrel – double scalar**

The relative accuracy required. If **epsrel** is negative, the absolute value is used. See Section 7.

5.2 Optional Input Parameters

1: **npts – int32 scalar**

Default: The dimension of the array **points**.

The number of user-supplied break points within the integration interval.

Constraint: $\mathbf{npts} \geq 0$.

2: **lw – int32 scalar**

Default: The dimension of the array **w**.

The value of **lw** (together with that of **liw**) imposes a bound on the number of sub-intervals into which the interval of integration may be divided by the function. The number of sub-intervals cannot exceed $(\mathbf{lw} - 2 \times \mathbf{npts} - 4)/4$. The more difficult the integrand, the larger **lw** should be.

Suggested value: a value in the range 800 to 2000 is adequate for most problems.

Default: 800

Constraint: $\mathbf{lw} \geq 2 \times \mathbf{npts} + 8$.

3: **liw – int32 scalar**

Default: The dimension of the array **iw**.

The number of sub-intervals into which the interval of integration may be divided cannot exceed $(\mathbf{liw} - \mathbf{npts} - 2)/2$.

Suggested value: $\mathbf{liw} = \mathbf{lw}/2$.

Default: $\mathbf{lw}/2$

Constraint: $\mathbf{liw} \geq \mathbf{npts} + 4$.

5.3 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **result – double scalar**

The approximation to the integral I.

2: **abserr – double scalar**

An estimate of the modulus of the absolute error, which should be an upper bound for $|I - \textbf{result}|$.

3: **w(lw) – double array**

Details of the computation, as described in Section 8.

4: **iw(liw) – int32 array**

iw(1) contains the actual number of sub-intervals used. The rest of the array is used as workspace.

5: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Note: d01al may return useful information for one or more of the following detected errors or warnings.

ifail = 1

The maximum number of subdivisions allowed with the given workspace has been reached without the accuracy requirements being achieved. Look at the integrand in order to determine the integration difficulties. If the position of a local difficulty within the interval can be determined (e.g., a singularity of the integrand or its derivative, a peak, a discontinuity, etc.) it should be supplied to the function as an element of the vector **points**. If necessary, another integrator, which is designed for handling the type of difficulty involved, must be used. Alternatively, consider relaxing the accuracy requirements specified by **epsabs** and **epsrel**, or increasing the amount of workspace.

ifail = 2

Round-off error prevents the requested tolerance from being achieved. Consider requesting less accuracy.

ifail = 3

Extremely bad local integrand behaviour causes a very strong subdivision around one (or more) points of the interval. The same advice applies as in the case of **ifail** = 1.

ifail = 4

The requested tolerance cannot be achieved because the extrapolation does not increase the accuracy satisfactorily; the returned result is the best which can be obtained. The same advice applies as in the case of **ifail** = 1.

ifail = 5

The integral is probably divergent, or slowly convergent. Please note that divergence can occur with any nonzero value of **ifail**.

ifail $= 6$

The input is invalid: break points are specified outside the integration range, **npts** $>$ LIMIT or **npts** < 0. **result** and **abserr** are set to zero.

ifail $= 7$

On entry, **lw** $< 2 \times$ **npts** $+ 8$,
or **liw** $<$ **npts** $+ 4$.

7 Accuracy

d01al cannot guarantee, but in practice usually achieves, the following accuracy:

$$|I - \mathbf{result}| \leq tol,$$

where

$$tol = \max\{|\mathbf{epsabs}|, |\mathbf{epsrel}| \times |I|\},$$

and **epsabs** and **epsrel** are user-specified absolute and relative error tolerances. Moreover, it returns the quantity **abserr** which, in normal circumstances, satisfies

$$|I - \mathbf{result}| \leq \mathbf{abserr} \leq tol.$$

8 Further Comments

The time taken by d01al depends on the integrand and the accuracy required.

If **ifail** $\neq 0$ on exit, then you may wish to examine the contents of the array **w**, which contains the end points of the sub-intervals used by d01al along with the integral contributions and error estimates over these sub-intervals.

Specifically, for $i = 1, 2, \ldots, n$, let r_i denote the approximation to the value of the integral over the sub-interval $[a_i, b_i]$ in the partition of $[a, b]$ and e_i be the corresponding absolute error estimate. Then, $\int_{a_i}^{b_i} f(x)\,dx \simeq r_i$ and **result** $= \sum_{i=1}^{n} r_i$ unless d01al terminates while testing for divergence of the integral (see Section 3.4.3 of Piessens *et al.* 1983). In this case, **result** (and **abserr**) are taken to be the values returned from the extrapolation process. The value of n is returned in **iw**(1), and the values a_i, b_i, e_i and r_i are stored consecutively in the array **w**, that is:

$a_i = \mathbf{w}(i)$,

$b_i = \mathbf{w}(n + i)$,

$e_i = \mathbf{w}(2n + i)$ and

$r_i = \mathbf{w}(3n + i)$.

9 Example

d01al_f.m

```
function [result] = d01alf_f(x)
  a = abs(x-1.0/7.0);
  if (a ~= 0.0)
    result = a^(-0.5);
  else
    result = 0;
  end
```

```
a = 0;
```

```
b = 1;
points = [0.1428571428571428];
epsabs = 0;
epsrel = 0.001;
[result, abserr, w, iw, ifail] = d01al('d01al_f', a, b, points, epsabs,
epsrel)
```

```
result =
    2.6076
abserr =
   8.0824e-14
w =
     array elided
iw =
     array elided
ifail =
           0
```