NAG Toolbox for MATLAB

d01at

1 Purpose

d01at is a general purpose integrator which calculates an approximation to the integral of a function $f(x)$ over a finite interval $[a, b]$:

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

2 Syntax

```
[result, abserr, w, iw, ifail] = d01at(f, a, b, epsabs, epsrel, 'lw', lw, 'liw', liw)
```

3 Description

d01at is based on the QUADPACK routine QAGS (see Piessens *et al.* 1983). It is an adaptive function, 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 local error estimation is described in Piessens *et al.* 1983.

The function is suitable as a general purpose integrator, and can be used when the integrand has singularities, especially when these are of algebraic or logarithmic type.

d01at requires a (sub)program to evaluate the integrand at an array of different points and is therefore particularly efficient when the evaluation can be performed in vector mode on a vector-processing machine. Otherwise the algorithm is identical to that used by d01aj.

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 values of the integrand f at a set of points.

Its specification is:

```
[fv] = f(x, n)
```

Input Parameters

1: **x(n) – double array**

The points at which the integrand f must be evaluated.

2: **n – int32 scalar**

The number of points at which the integrand is to be evaluated. The actual value of **n** is always 21.

Output Parameters

1: **fv(n) – double array**

fv(j) must contain the value of f at the point **x**(j), for $j = 1, 2, \ldots,$ **n**.

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: **epsabs – double scalar**

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

5: **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: **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 **lw**$/4$. The more difficult the integrand, the larger **lw** should be.

Suggested value: **lw** $= 800$ to 2000 is adequate for most problems.

Default: 800

Constraint: **lw** ≥ 4.

2: **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 **liw**.

Suggested value: **liw** $=$ **lw**$/4$.

Default: **lw**$/4$

Constraint: **liw** ≥ 1.

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 $|I - \mathbf{result}|$.

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: d01at 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.) you will probably gain from splitting up the interval at this point and calling the integrator on the subranges. 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

On entry, **lw** < 4,
or **liw** < 1.

7 Accuracy

d01at 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

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 d01at along with the integral contributions and error estimates over the 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 $\mathbf{result} = \sum_{i=1}^{n} r_i$, unless d01at 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 $\mathbf{iw}(1)$, 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

d01at_f.m

```
function [fv] = d01at_f(x,n)
  fv = zeros(n,1);
  for i = 1:n
    fv(i) = x(i)*sin(30*x(i))/sqrt(1-x(i)^2/(4*pi^2));
  end
```

```
a = 0;
b = 6.283185307179586;
epsabs = 0;
epsrel = 0.0001;
[result, abserr, w, iw, ifail] = d01at('d01at_f', a, b, epsabs, epsrel)
```

```
result =
   -2.5433
abserr =
   1.2752e-05
w =
     array elided
iw =
     array elided
ifail =
           0
```