NAG Toolbox for MATLAB

d01an

1 Purpose

d01an calculates an approximation to the sine or the cosine transform of a function g over $[a, b]$:

$$I = \int_a^b g(x)\sin(\omega x)\,dx \qquad \text{or} \qquad I = \int_a^b g(x)\cos(\omega x)\,dx$$

(for a user-specified value of ω).

2 Syntax

```
[result, abserr, w, iw, ifail] = d01an(g, a, b, omega, key, epsabs, epsrel, 'lw', lw, 'liw', liw)
```

3 Description

d01an is based on the QUADPACK routine QFOUR (see Piessens *et al.* 1983). It is an adaptive function, designed to integrate a function of the form $g(x)w(x)$, where $w(x)$ is either $\sin(\omega x)$ or $\cos(\omega x)$. If a sub-interval has length

$$L = |b - a|2^{-l}$$

then the integration over this sub-interval is performed by means of a modified Clenshaw–Curtis procedure (see Piessens and Branders 1975) if $L\omega > 4$ and $l \leq 20$. In this case a Chebyshev-series approximation of degree 24 is used to approximate $g(x)$, while an error estimate is computed from this approximation together with that obtained using Chebyshev-series of degree 12. If the above conditions do not hold then Gauss 7-point and Kronrod 15-point rules are used. The algorithm, described in Piessens *et al.* 1983, incorporates a global acceptance criterion (as defined in 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.

4 References

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

Piessens R and Branders M 1975 Algorithm 002. Computation of oscillating integrals *J. Comput. Appl. Math.* **1** 153–164

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: **g – string containing name of m-file**

g must return the value of the function g at a given point **x**.

Its specification is:

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

Input Parameters

1: **x – double scalar**

The point at which the function g 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: **omega – double scalar**

The parameter ω in the weight function of the transform.

5: **key – int32 scalar**

Indicates which integral is to be computed.

key $= 1$

$w(x) = \cos(\omega x)$.

key $= 2$

$w(x) = \sin(\omega x)$.

Constraint: **key** $= 1$ or 2.

6: **epsabs – double scalar**

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

7: **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**/2.

Suggested value: **liw** = **lw**/2.

Default: **lw**/2

Constraint: **liw** $\geq$ 2.

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: d01an 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 behaviour of $g(x)$ 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, **key** < 1,
or **key** > 2.

ifail $= 7$

On entry, **lw** < 4,
or **liw** < 2.

7 Accuracy

d01an 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 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 d01an 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 d01an 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, $\int_{a_i}^{b_i} g(x)w(x)\,dx \simeq r_i$ and $\mathbf{result} = \sum_{i=1}^{n} r_i$ unless d01an 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

```
d01an_g.m
```

```
function [result] = d01an_g(x)
  if (x > 0)
    result = log(x);
  else
    result = 0;
  end
```

```
a = 0;
b = 1;
omega = 31.41592653589793;
key = int32(2);
epsabs = 0;
epsrel = 0.0001;
[result, abserr, w, iw, ifail] = d01an('d01an_g', a, b, omega, key,
epsabs, epsrel)
```

```
result =
   -0.1281
abserr =
   3.5796e-06
w =
     array elided
iw =
     array elided
ifail =
           0
```