NAG Toolbox for MATLAB

d01as

1 Purpose

d01as calculates an approximation to the sine or the cosine transform of a function g over $[a\infty)$:

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

(for a user-specified value of ω).

2 Syntax

```
[result, abserr, lst, erlst, rslst, ierlst, iw, ifail] = d01as(g, a,
omega, key, epsabs, 'limlst', limlst, 'lw', lw, 'liw', liw)
```

3 Description

d01as is based on the QUADPACK routine QAWFE (see Piessens *et al.* 1983). It is an adaptive function, designed to integrate a function of the form $g(x)w(x)$ over a semi-infinite interval, where $w(x)$ is either $\sin(\omega x)$ or $\cos(\omega x)$.

Over successive intervals

$$C_k = [a + (k-1)c, a + kc], \qquad k = 1, 2, \ldots, \textbf{lst}$$

integration is performed by the same algorithm as is used by d01an. The intervals C_k are of constant length

$$c = \{2[|\omega|] + 1\}\pi/|\omega|, \qquad \omega \neq 0,$$

where $[|\omega|]$ represents the largest integer less than or equal to $|\omega|$. Since c equals an odd number of half periods, the integral contributions over succeeding intervals will alternate in sign when the function g is positive and monotonically decreasing over $[a\infty)$. The algorithm, described in Piessens *et al.* 1983, 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 by Piessens *et al.* 1983.

If $\omega = 0$ and **key** $= 1$, the function uses the same algorithm as d01am (with **epsrel** $= 0.0$).

In contrast to the other functions in Chapter D01, d01as works only with an **absolute** error tolerance (**epsabs**). Over the interval C_k it attempts to satisfy the absolute accuracy requirement

$$EPSA_k = U_k \times \textbf{epsabs},$$

where $U_k = (1-p)p^{k-1}$, for $k = 1, 2, \ldots$ and $p = 0.9$.

However, when difficulties occur during the integration over the kth sub-interval C_k such that the error flag **ierlst**(k) is nonzero, the accuracy requirement over subsequent intervals is relaxed. See Piessens *et al.* 1983 for more details.

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, 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 $e_m(\mathrm{S}_n)$ 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: **omega – double scalar**

The parameter ω in the weight function of the transform.

4: **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.

5: **epsabs – double scalar**

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

5.2 Optional Input Parameters

1: **limlst – int32 scalar**

Default: The dimension of the arrays **erlst**, **rslst**, **ierlst**. (An error is raised if these dimensions are not equal.)

an upper bound on the number of intervals C_k needed for the integration.

Suggested value: **limlst** $= 50$ is adequate for most problems.

Default: 50

Constraint: **limlst** ≥ 3.

2: **lw – int32 scalar**

Default: The dimension of the array .

The value of **lw** (together with that of **liw**) imposes a bound on the number of sub-intervals into which each interval C_k 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: a value in the range 800 to 2000 is adequate for most problems.

Default: 800

Constraint: **lw** ≥ 4.

3: **liw – int32 scalar**

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

The number of sub-intervals into which each interval C_k may be divided cannot exceed **liw**$/2$.

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

Default: **lw**$/2$

Constraint: **liw** ≥ 2.

5.3 Input Parameters Omitted from the MATLAB Interface

w

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 $|I -$ **result**$|$.

3: **lst – int32 scalar**

The number of intervals C_k actually used for the integration.

4: **erlst(limlst) – double array**

erlst(k) contains the error estimate corresponding to the integral contribution over the interval C_k, for $k = 1, 2, \ldots,$ **lst**.

5: **rslst(limlst) – double array**

rslst(k) contains the integral contribution over the interval C_k, for $k = 1, 2, \ldots,$ **lst**.

6: **ierlst(limlst) – int32 array**

ierlst(k) contains the error flag corresponding to **rslst**(k), for $k = 1, 2, \ldots,$ **lst**. See Section 6.

7: **iw(liw) – int32 array**

iw(1) contains the maximum number of sub-intervals actually used for integrating over any of the intervals C_k. The rest of the array is used as workspace.

8: **ifail – int32 scalar**

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

6 Error Indicators and Warnings

Note: d01as 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 d01as on the infinite subrange and an appropriate integrator on the finite subrange. Alternatively, consider relaxing the accuracy requirements specified by **epsabs** 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** $= 1$.

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 d01as on the infinite subrange and an appropriate integrator on the finite subrange. Alternatively, consider relaxing the accuracy requirements specified by **epsabs** or increasing the amount of workspace.

Please note that divergence can occur with any nonzero value of **ifail**.

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,
or **limlst** < 3.

ifail $= 7$

Bad integration behaviour occurs within one or more of the intervals C_k. Location and type of the difficulty involved can be determined from the vector **ierlst**.

ifail $= 8$

Maximum number of intervals C_k ($=$ **limlst**) allowed has been achieved. Increase the value of **limlst** to allow more cycles.

ifail $= 9$

The extrapolation table constructed for convergence acceleration of the series formed by the integral contribution over the intervals C_k, does not converge to the required accuracy.

ifail $= 10$

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

In the cases **ifail** $= 7$, 8 or 9, additional information about the cause of the error can be obtained from the array **ierlst**, as follows:

ierlst$(k) = 1$

The maximum number of subdivisions $= \min(\mathbf{lw}/4, \mathbf{liw}/2)$ has been achieved on the kth interval.

ierlst$(k) = 2$

Occurrence of round-off error is detected and prevents the tolerance imposed on the kth interval from being achieved.

ierlst$(k) = 3$

Extremely bad integrand behaviour occurs at some points of the kth interval.

ierlst$(k) = 4$

The integration procedure over the kth interval does not converge (to within the required accuracy) due to round-off in the extrapolation procedure invoked on this interval. It is assumed that the result on this interval is the best which can be obtained.

ierlst$(k) = 5$

The integral over the kth interval is probably divergent or slowly convergent. It must be noted that divergence can occur with any other value of **ierlst**(k).

7 Accuracy

d01as cannot guarantee, but in practice usually achieves, the following accuracy:

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

where **epsabs** is the user-specified absolute error tolerance. Moreover, it returns the quantity **abserr**, which, in normal circumstances, satisfies

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

8 Further Comments

None.

9 Example

```
d01as_g.m

function [result] = d01as_g(x)
  if (x > 0)
    result = 1/sqrt(x);
  else
    result = 0;
  end
```

```
a = 0;
omega = 1.570796326794897;
key = int32(1);
epsabs = 0.001;
[result, abserr, lst, erlst, rslst, ierlst, iw, ifail] = ...
    d01as('d01as_g', a, omega, key, epsabs)
```

```
result =
    1.0000
abserr =
   5.9234e-04
lst =
           6
erlst =
     array elided
rslst =
     array elided
ierlst =
     array elided
iw =
     array elided
ifail =
           0
```