NAG Library Routine Document

D02UBF

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

D02UBF evaluates a function, or one of its lower order derivatives, from its Chebyshev series representation at Chebyshev Gauss–Lobatto points on $[a, b]$. The coefficients of the Chebyshev series representation required are usually derived from those returned by D02UAF or D02UEF.

2 Specification

```
SUBROUTINE D02UBF (N, A, B, Q, C, F, IFAIL)

INTEGER              N, Q, IFAIL
REAL (KIND=nag_wp) A, B, C(N+1), F(N+1)
```

3 Description

D02UBF evaluates the Chebyshev series

$$S(\bar{x}) = \tfrac{1}{2}c_1 T_0(\bar{x}) + c_2 T_1(\bar{x}) + c_3 T_2(\bar{x}) + \cdots + c_{n+1} T_n(\bar{x}),$$

or its derivative (up to fourth order) at the Chebyshev Gauss–Lobatto points on $[a, b]$. Here $T_j(\bar{x})$ denotes the Chebyshev polynomial of the first kind of degree j with argument $\bar{x}$ defined on $[-1, 1]$. In terms of your original variable, x say, the input values at which the function values are to be provided are

$$x_r = -\tfrac{1}{2}(b - a)\cos(\pi(r - 1)/n) + \tfrac{1}{2}(b + a), \qquad r = 1, 2, \ldots, n + 1,$$

where b and a are respectively the upper and lower ends of the range of x over which the function is required.

The calculation is implemented by a forward one-dimensional discrete Fast Fourier Transform (DFT).

4 References

Canuto C (1988) *Spectral Methods in Fluid Dynamics* 502 Springer

Canuto C, Hussaini M Y, Quarteroni A and Zang T A (2006) *Spectral Methods: Fundamentals in Single Domains* Springer

Trefethen L N (2000) *Spectral Methods in MATLAB* SIAM

5 Parameters

1: N – INTEGER *Input*

On entry: n, where the number of grid points is $n + 1$. This is also the largest order of Chebyshev polynomial in the Chebyshev series to be computed.

Constraint: $\mathrm{N} > 0$ and N is even.

2: A – REAL (KIND=nag_wp) *Input*

On entry: a, the lower bound of domain $[a, b]$.

Constraint: $\mathrm{A} < \mathrm{B}$.

3: B – REAL (KIND=nag_wp) *Input*

On entry: b, the upper bound of domain $[a, b]$.

Constraint: $\mathrm{B} > \mathrm{A}$.

4: Q – INTEGER *Input*

On entry: the order, q, of the derivative to evaluate.

Constraint: $0 \leq \mathrm{Q} \leq 4$.

5: $\mathrm{C}(\mathrm{N} + 1)$ – REAL (KIND=nag_wp) array *Input*

On entry: the Chebyshev coefficients, c_i, for $i = 1, 2, \ldots, n + 1$.

6: $\mathrm{F}(\mathrm{N} + 1)$ – REAL (KIND=nag_wp) array *Output*

On exit: the derivatives $S^{(q)}x_i$, for $i = 1, 2, \ldots, n + 1$, of the Chebyshev series, S.

7: 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, if you are not familiar with this parameter, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: $\mathrm{IFAIL} = 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry $\mathrm{IFAIL} = 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

$\mathrm{IFAIL} = 1$

On entry, $\mathrm{N} \leq 0$ or N is odd.

$\mathrm{IFAIL} = 2$

On entry, $\mathrm{A} \geq \mathrm{B}$.

$\mathrm{IFAIL} = 3$

$\mathrm{Q} \neq 0$, 1, 2, 3 or 4.

$\mathrm{IFAIL} = -999$

Internal memory allocation failed.

7 Accuracy

Evaluations of DFT to obtain function or derivative values should be an order n multiple of ***machine precision*** assuming full accuracy to ***machine precision*** in the given Chebyshev series representation.

8 Further Comments

The number of operations is of the order $n\log n$ and the memory requirements are $O(n)$; thus the computation remains efficient and practical for very fine discretizations (very large values of n).

9 Example

See Section 9 in D02UEF.