

NAG Library Routine Document

D02UEF

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

D02UEF finds the solution of a linear constant coefficient boundary value problem by using the Chebyshev integration formulation on a Chebyshev Gauss–Lobatto grid.

2 Specification

```
SUBROUTINE D02UEF (N, A, B, M, C, BMAT, Y, BVEC, F, UC, RESID, IFAIL)
INTEGER          N, M, IFAIL
REAL (KIND=nag_wp) A, B, C(N+1), BMAT(M,M+1), Y(M), BVEC(M), F(M+1),      &
UC(N+1,M+1), RESID
```

3 Description

D02UEF solves the constant linear coefficient ordinary differential problem

$$\sum_{j=0}^m f_{j+1} \frac{d^j u}{dx^j} = f(x), \quad x \in [a, b]$$

subject to a set of m linear constraints at points $y_i \in [a, b]$, for $i = 1, 2, \dots, m$:

$$\sum_{j=0}^m B_{i,j+1} \left(\frac{d^j u}{dx^j} \right)_{(x=y_i)} = \beta_i,$$

where $1 \leq m \leq 4$, B is an $m \times (m+1)$ matrix of constant coefficients and β_i are constants. The points y_i are usually either a or b .

The function $f(x)$ is supplied as an array of Chebyshev coefficients c_j , $j = 0, 1, \dots, n$ for the function discretized on $n+1$ Chebyshev Gauss–Lobatto points (as returned by D02UCF); the coefficients are normally obtained by a previous call to D02UAF. The solution and its derivatives (up to order m) are returned, in the form of their Chebyshev series representation, as arrays of Chebyshev coefficients; subsequent calls to D02UBF will return the corresponding function and derivative values at the Chebyshev Gauss–Lobatto discretization points on $[a, b]$. Function and derivative values can be obtained on any uniform grid over the same range $[a, b]$ by calling the interpolation routine D02UWF.

4 References

Clenshaw C W (1957) The numerical solution of linear differential equations in Chebyshev series *Proc. Camb. Phil. Soc.* **53** 134–149

Coutsias E A, Hagstrom T and Torres D (1996) An efficient spectral method for ordinary differential equations with rational function coefficients *Mathematics of Computation* **65(214)** 611–635

Greengard L (1991) Spectral integration and two-point boundary value problems *SIAM J. Numer. Anal.* **28(4)** 1071–80

Lundbladh A, Hennigson D S and Johansson A V (1992) An efficient spectral integration method for the solution of the Navier–Stokes equations *Technical report FFA–TN 1992–28* Aeronautical Research Institute of Sweden

Muite B K (2010) A numerical comparison of Chebyshev methods for solving fourth-order semilinear initial boundary value problems *Journal of Computational and Applied Mathematics* **234(2)** 317–342

5 Parameters

- 1: N – INTEGER *Input*
On entry: n , where the number of grid points is $n + 1$.
Constraint: $N \geq 8$ and N is even.
- 2: A – REAL (KIND=nag_wp) *Input*
On entry: a , the lower bound of domain $[a, b]$.
Constraint: $A < B$.
- 3: B – REAL (KIND=nag_wp) *Input*
On entry: b , the upper bound of domain $[a, b]$.
Constraint: $B > A$.
- 4: M – INTEGER *Input*
On entry: the order, m , of the boundary value problem to be solved.
Constraint: $1 \leq M \leq 4$.
- 5: C(N + 1) – REAL (KIND=nag_wp) array *Input*
On entry: the Chebyshev coefficients c_j , $j = 0, 1, \dots, n$, for the right hand side of the boundary value problem. Usually these are obtained by a previous call of D02UAF.
- 6: BMAT(M, M + 1) – REAL (KIND=nag_wp) array *Input/Output*
On entry: $\text{BMAT}(i, j + 1)$ must contain the coefficients $B_{i,j+1}$, for $i = 1, 2, \dots, m$ and $j = 0, 1, \dots, m$, in the problem formulation of Section 3.
On exit: the coefficients have been scaled to form an equivalent problem defined on the domain $[-1, 1]$.
- 7: Y(M) – REAL (KIND=nag_wp) array *Input*
On entry: the points, y_i , for $i = 1, 2, \dots, m$, where the boundary conditions are discretized.
- 8: BVEC(M) – REAL (KIND=nag_wp) array *Input*
On entry: the values, β_i , for $i = 1, 2, \dots, m$, in the formulation of the boundary conditions given in Section 3.
- 9: F(M + 1) – REAL (KIND=nag_wp) array *Input/Output*
On entry: the coefficients, f_j , for $j = 1, 2, \dots, m + 1$, in the formulation of the linear boundary value problem given in Section 3. The highest order term, $F(M + 1)$, needs to be nonzero to have a well posed problem.
On exit: the coefficients have been scaled to form an equivalent problem defined on the domain $[-1, 1]$.
- 10: UC(N + 1, M + 1) – REAL (KIND=nag_wp) array *Output*
On exit: the Chebyshev coefficients in the Chebyshev series representations of the solution and derivatives of the solution to the boundary value problem. The $n + 1$ elements $\text{UC}(1 : N + 1, 1)$ contain the coefficients representing the solution $U(x_i)$, for $i = 0, 1, \dots, n$. $\text{UC}(1 : N + 1, j + 1)$ contains the coefficients representing the j th derivative of U , for $j = 1, 2, \dots, m$.

11: RESID – REAL (KIND=nag_wp)

Output

On exit: the maximum residual resulting from substituting the solution vectors returned in UC into both linear equations of Section 3 representing the linear boundary value problem and associated boundary conditions. That is

$$\max \left\{ \max_{i=1,m} \left(\left| \sum_{j=0}^m B_{i,j+1} \left(\frac{d^j u}{dx^j} \right)_{(x=y_i)} - \beta_i \right| \right), \max_{i=1,n+1} \left(\left| \sum_{j=0}^m f_{j+1} \left(\frac{d^j u}{dx^j} \right)_{(x=x_i)} - f(x) \right| \right) \right\}.$$

12: 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: 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 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:

IFAIL = 1

On entry, N = *value*.

Constraint: N is even.

On entry, N = *value*.

Constraint: N ≥ 8.

IFAIL = 2

On entry, A = *value* and B = *value*.

Constraint: A < B.

IFAIL = 3

On entry, F(M + 1) = 0.0.

IFAIL = 6

On entry, M = *value*.

Constraint: 1 ≤ M ≤ 4.

IFAIL = 7

Internal error while unpacking matrix during iterative refinement.

Please contact NAG.

IFAIL = 8

Singular matrix encountered during iterative refinement.

Please check that your system is well posed.

IFAIL = 9

During iterative refinement, the maximum number of iterations was reached.
Number of iterations = $\langle value \rangle$ and residual achieved = $\langle value \rangle$.

IFAIL = 10

During iterative refinement, convergence was achieved, but the residual is more than $100 \times$ *machine precision*. Residual achieved on convergence = $\langle value \rangle$.

IFAIL = -99

An unexpected error has been triggered by this routine. Please contact NAG.
See Section 3.8 in the Essential Introduction for further information.

IFAIL = -399

Your licence key may have expired or may not have been installed correctly.
See Section 3.7 in the Essential Introduction for further information.

IFAIL = -999

Dynamic memory allocation failed.
See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

The accuracy should be close to *machine precision* for well conditioned boundary value problems.

8 Parallelism and Performance

D02UEF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

D02UEF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 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). Collocation methods will be faster for small problems, but the method of D02UEF should be faster for larger discretizations.

10 Example

This example solves the third-order problem $4U_{xxx} + 3U_{xx} + 2U_x + U = 2 \sin x - 2 \cos x$ on $[-\pi/2, \pi/2]$ subject to the boundary conditions $U[-\pi/2] = 0$, $3U_{xx}[-\pi/2] + 2U_x[-\pi/2] + U[-\pi/2] = 2$, and $3U_{xx}[\pi/2] + 2U_x[\pi/2] + U[\pi/2] = -2$ using the Chebyshev integration formulation on a Chebyshev Gauss-Lobatto grid of order 16.

10.1 Program Text

```

!   D02UEF Example Program Text
!   Mark 25 Release. NAG Copyright 2014.

Module d02uefe_mod

!   D02UEF Example Program Module:
!       Parameters and User-defined Routines

!   .. Use Statements ..
Use nag_library, Only: nag_wp
!   .. Implicit None Statement ..
Implicit None
!   .. Accessibility Statements ..
Private
Public                                     :: bndary, exact, pdedef
!   .. Parameters ..
Real (Kind=nag_wp), Parameter             :: four = 4.0_nag_wp
Real (Kind=nag_wp), Parameter             :: one = 1.0_nag_wp
Real (Kind=nag_wp), Parameter             :: three = 3.0_nag_wp
Real (Kind=nag_wp), Parameter, Public     :: two = 2.0_nag_wp
Real (Kind=nag_wp), Parameter, Public     :: zero = 0.0_nag_wp
Integer, Parameter, Public                :: m = 3, nin = 5, nout = 6
Logical, Parameter, Public                :: reqerr = .False.
!   .. Local Scalars ..
Real (Kind=nag_wp), Public, Save          :: a, b
Contains
Function exact(x,q)

!   .. Function Return Value ..
Real (Kind=nag_wp)                        :: exact
!   .. Scalar Arguments ..
Real (Kind=nag_wp), Intent (In)           :: x
Integer, Intent (In)                      :: q
!   .. Intrinsic Procedures ..
Intrinsic                                  :: cos, sin
!   .. Executable Statements ..
Select Case (q)
Case (0)
    exact = cos(x)
Case (1)
    exact = -sin(x)
Case (2)
    exact = -cos(x)
Case (3)
    exact = sin(x)
End Select
End Function exact
Subroutine bndary(m,y,bmat,bvec)

!   .. Scalar Arguments ..
Integer, Intent (In)                      :: m
!   .. Array Arguments ..
Real (Kind=nag_wp), Intent (Out)          :: bmat(m,m+1), bvec(m), y(m)
!   .. Executable Statements ..
!   Boundary condition on left side of domain
y(1:2) = a
y(3) = b
!   Set up Dirichlet condition using exact solution
bmat(1:m,1:m+1) = zero
bmat(1:3,1) = one
bmat(2:3,2) = two
bmat(2:3,3) = three
bvec(1) = zero
bvec(2) = two
bvec(3) = -two
Return
End Subroutine bndary
Subroutine pdedef(m,f)

```

```

!      .. Scalar Arguments ..
      Integer, Intent (In)           :: m
!      .. Array Arguments ..
      Real (Kind=nag_wp), Intent (Out) :: f(m+1)
!      .. Executable Statements ..
      f(1) = one
      f(2) = two
      f(3) = three
      f(4) = four
      Return
      End Subroutine pdedef
End Module d02uefe_mod
Program d02uefe

!      D02UEF Example Main Program

!      .. Use Statements ..
      Use nag_library, Only: d02uaf, d02ubf, d02ucf, d02uef, nag_wp, x01aaf, &
                           x02ajf
      Use d02uefe_mod, Only: a, b, bndary, exact, m, nin, nout, pdedef, &
                           reqerr, two, zero
!      .. Implicit None Statement ..
      Implicit None
!      .. Local Scalars ..
      Real (Kind=nag_wp)           :: pi, resid, teneps
      Integer                      :: i, ifail, n, q, ql
!      .. Local Arrays ..
      Real (Kind=nag_wp)           :: bmat(m,m+1), bvec(m), f(m+1), &
                                   uerr(m+1), y(m)
      Real (Kind=nag_wp), Allocatable :: c(:), f0(:), u(:,,:), uc(:,,:), x(:)
!      .. Intrinsic Procedures ..
      Intrinsic                    :: abs, cos, int, max, sin
!      .. Executable Statements ..
      Write (nout,*) ' D02UEF Example Program Results '
      Write (nout,*)

      Read (nin,*)
      Read (nin,*) n

      Allocate (u(n+1,m+1),f0(n+1),c(n+1),uc(n+1,m+1),x(n+1))

!      Set up domain, boundary conditions and definition
      pi = x01aaf(zero)
      a = -pi/two
      b = pi/two
      Call bndary(m,y,bmat,bvec)
      Call pdedef(m,f)

!      Set up solution grid.
      ifail = 0
      Call d02ucf(n,a,b,x,ifail)

!      Set up problem right hand sides for grid and transform.
      f0(1:n+1) = two*sin(x(1:n+1)) - two*cos(x(1:n+1))
      ifail = 0
      Call d02uaf(n,f0,c,ifail)

!      Solve in coefficient space.
      ifail = 0
      Call d02uef(n,a,b,m,c,bmat,y,bvec,f,uc,resid,ifail)
!      Evaluate solution and derivatives on Chebyshev grid.
      Do q = 0, m
         ifail = 0
         Call d02ubf(n,a,b,q,uc(1,q+1),u(1,q+1),ifail)
      End Do
!      Print solution
      Write (nout,*) ' Numerical Solution U and its first three derivatives'
      Write (nout,*)
      Write (nout,99999)
      Write (nout,99998)(x(i),u(i,1:m+1),i=1,n+1)

```

```

If (reqerr) Then
  uerr(1:m+1) = zero
  Do i = 1, n + 1
    Do q = 0, m
      q1 = q + 1
      uerr(q1) = max(uerr(q1),abs(u(i,q1)-exact(x(i),q)))
    End Do
  End Do
  teneps = 10.0_nag_wp*x02ajf()
  Write (nout,'(//)')
  Write (nout,99997)(q,10*(int(uerr(q+1)/teneps)+1),q=0,m)
End If

99999 Format (1X,T8,'X',T18,'U',T28,'Ux',T37,'Uxx',T47,'Uxxx')
99998 Format (1X,5F10.4)
99997 Format (1X,'Error in the order ',I1,' derivative of U is < ',I8, &
  ' * machine precision.')

End Program d02uefe

```

10.2 Program Data

D02UEF Example Program Data
 16 : N

10.3 Program Results

D02UEF Example Program Results

Numerical Solution U and its first three derivatives

X	U	Ux	Uxx	Uxxx
-1.5708	-0.0000	1.0000	0.0000	-1.0000
-1.5406	0.0302	0.9995	-0.0302	-0.9995
-1.4512	0.1193	0.9929	-0.1193	-0.9929
-1.3061	0.2616	0.9652	-0.2616	-0.9652
-1.1107	0.4440	0.8960	-0.4440	-0.8960
-0.8727	0.6428	0.7661	-0.6428	-0.7661
-0.6011	0.8247	0.5656	-0.8247	-0.5656
-0.3064	0.9534	0.3017	-0.9534	-0.3017
-0.0000	1.0000	0.0000	-1.0000	-0.0000
0.3064	0.9534	-0.3017	-0.9534	0.3017
0.6011	0.8247	-0.5656	-0.8247	0.5656
0.8727	0.6428	-0.7661	-0.6428	0.7661
1.1107	0.4440	-0.8960	-0.4440	0.8960
1.3061	0.2616	-0.9652	-0.2616	0.9652
1.4512	0.1193	-0.9929	-0.1193	0.9929
1.5406	0.0302	-0.9995	-0.0302	0.9995
1.5708	-0.0000	-1.0000	0.0000	1.0000
