

NAG Fortran Library Routine Document

D03FAF

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

D03FAF solves the Helmholtz equation in Cartesian co-ordinates in three dimensions using the standard seven-point finite difference approximation. This routine is designed to be particularly efficient on vector processors.

2 Specification

```

SUBROUTINE D03FAF(XS, XF, L, LBDCND, BDXS, BDXF, YS, YF, M, MBDCND,
1          BDYS, BDYF, ZS, ZF, N, NBDCND, BDZS, BDZF, LAMBDA,
2          LDIMF, MDIMF, F, PERTRB, W, LWRK, IFAIL)
  INTEGER      L, LBDCND, M, MBDCND, N, NBDCND, LDIMF, MDIMF, LWRK,
1          IFAIL
  real        XS, XF, BDXS(MDIMF,N+1), BDXF(MDIMF,N+1), YS, YF,
1          BDYS(LDIMF,N+1), BDYF(LDIMF,N+1), ZS, ZF,
2          BDZS(LDIMF,M+1), BDZF(LDIMF,M+1), LAMBDA,
3          F(LDIMF,MDIMF,N+1), PERTRB, W(LWRK)

```

3 Description

D03FAF solves the three-dimensional Helmholtz equation in Cartesian co-ordinates:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} + \lambda u = f(x, y, z).$$

This subroutine forms the system of linear equations resulting from the standard seven-point finite difference equations, and then solves the system using a method based on the fast Fourier transform (FFT) described by Swarztrauber (1984). This subroutine is based on the routine HW3CRT from FISHPACK (see Swarztrauber and Sweet (1979)).

More precisely, the routine replaces all the second derivatives by second-order central difference approximations, resulting in a block tridiagonal system of linear equations. The equations are modified to allow for the prescribed boundary conditions. Either the solution or the derivative of the solution may be specified on any of the boundaries, or the solution may be specified to be periodic in any of the three dimensions. By taking the discrete Fourier transform in the x - and y -directions, the equations are reduced to sets of tridiagonal systems of equations. The Fourier transforms required are computed using the multiple FFT routines found in Chapter C06 of the NAG Fortran Library.

4 References

Swarztrauber P N (1984) Fast Poisson solvers *Studies in Numerical Analysis* (ed G H Golub) Mathematical Association of America

Swarztrauber P N and Sweet R A (1979) Efficient Fortran subprograms for the solution of separable elliptic partial differential equations *ACM Trans. Math. Software* **5** 352–364

5 Parameters

- 1: ***XS*** – ***real*** *Input*
- On entry: the lower bound of the range of x , i.e., $XS \leq x \leq XF$.
- Constraint: $XS < XF$.

- 2: **XF** – *real* *Input*
On entry: the upper bound of the range of x , i.e., $XS \leq x \leq XF$.
Constraint: $XS < XF$.
- 3: **L** – **INTEGER** *Input*
On entry: the number of panels into which the interval (XS, XF) is subdivided. Hence, there will be $L + 1$ grid points in the x -direction given by $x_i = XS + (i - 1) \times \delta x$, for $i = 1, 2, \dots, L + 1$, where $\delta x = (XF - XS)/L$ is the panel width.
Constraint: $L \geq 5$.
- 4: **LBDCND** – **INTEGER** *Input*
On entry: indicates the type of boundary conditions at $x = XS$ and $x = XF$.
LBDCND = 0
 if the solution is periodic in x , i.e., $u(XS, y, z) = u(XF, y, z)$.
LBDCND = 1
 if the solution is specified at $x = XS$ and $x = XF$.
LBDCND = 2
 if the solution is specified at $x = XS$ and the derivative of the solution with respect to x is specified at $x = XF$.
LBDCND = 3
 if the derivative of the solution with respect to x is specified at $x = XS$ and $x = XF$.
LBDCND = 4
 if the derivative of the solution with respect to x is specified at $x = XS$ and the solution is specified at $x = XF$.
Constraint: $0 \leq \text{LBDCND} \leq 4$.
- 5: **BDXS(MDIMF,N+1)** – *real* array *Input*
On entry: the values of the derivative of the solution with respect to x at $x = XS$. When **LBDCND** = 3 or 4, $\text{BDXS}(j, k) = u_x(XS, y_j, z_k)$, for $j = 1, 2, \dots, M + 1$; $k = 1, 2, \dots, N + 1$.
 When **LBDCND** has any other value, **BDXS** is not referenced.
- 6: **BDXF(MDIMF,N+1)** – *real* array *Input*
On entry: the values of the derivative of the solution with respect to x at $x = XF$. When **LBDCND** = 2 or 3, $\text{BDXF}(j, k) = u_x(XF, y_j, z_k)$, for $j = 1, 2, \dots, M + 1$; $k = 1, 2, \dots, N + 1$.
 When **LBDCND** has any other value, **BDXF** is not referenced.
- 7: **YS** – *real* *Input*
On entry: the lower bound of the range of y , i.e., $YS \leq y \leq YF$.
Constraint: $YS < YF$.
- 8: **YF** – *real* *Input*
On entry: the upper bound of the range of y , i.e., $YS \leq y \leq YF$.
Constraint: $YS < YF$.

- 9: M – INTEGER *Input*
On entry: the number of panels into which the interval (YS,YF) is subdivided. Hence, there will be $M + 1$ grid points in the y -direction given by $y_j = YS + (j - 1) \times \delta y$ for $j = 1, 2, \dots, M + 1$, where $\delta y = (YF - YS)/M$ is the panel width.
Constraint: $M \geq 5$.
- 10: MBDCND – INTEGER *Input*
On entry: indicates the type of boundary conditions at $y = YS$ and $y = YF$.
 MBDCND = 0
 if the solution is periodic in y , i.e., $u(x, YF, z) = u(x, YS, z)$.
 MBDCND = 1
 if the solution is specified at $y = YS$ and $y = YF$.
 MBDCND = 2
 if the solution is specified at $y = YS$ and the derivative of the solution with respect to y is specified at $y = YF$.
 MBDCND = 3
 if the derivative of the solution with respect to y is specified at $y = YS$ and $y = YF$.
 MBDCND = 4
 if the derivative of the solution with respect to y is specified at $y = YS$ and the solution is specified at $y = YF$.
Constraint: $0 \leq MBDCND \leq 4$.
- 11: BDYS(LDIMF,N+1) – *real* array *Input*
On entry: the values of the derivative of the solution with respect to y at $y = YS$. When MBDCND = 3 or 4, $BDYS(i, k) = u_y(x_i, YS, z_k)$, for $i = 1, 2, \dots, L + 1$; $k = 1, 2, \dots, N + 1$.
 When MBDCND has any other value, BDYS is not referenced.
- 12: BDYF(LDIMF,N+1) – *real* array *Input*
On entry: the values of the derivative of the solution with respect to y at $y = YF$. When MBDCND = 2 or 3, $BDYF(i, k) = u_y(x_i, YF, z_k)$, for $i = 1, 2, \dots, L + 1$; $k = 1, 2, \dots, N + 1$.
 When MBDCND has any other value, BDYF is not referenced.
- 13: ZS – *real* *Input*
On entry: the lower bound of the range of z , i.e., $ZS \leq z \leq ZF$.
Constraint: $ZS < ZF$.
- 14: ZF – *real* *Input*
On entry: the upper bound of the range of z , i.e., $ZS \leq z \leq ZF$.
Constraint: $ZS < ZF$.
- 15: N – INTEGER *Input*
On entry: the number of panels into which the interval (ZS,ZF) is subdivided. Hence, there will be $N + 1$ grid points in the z -direction given by $z_k = ZS + (k - 1) \times \delta z$, for $k = 1, 2, \dots, N + 1$, where $\delta z = (ZF - ZS)/N$ is the panel width.
Constraint: $N \geq 5$.

- 16: NBDCND – INTEGER *Input*
- On entry:* specifies the type of boundary conditions at $z = \text{ZS}$ and $z = \text{ZF}$.
- NBDCND = 0
- if the solution is periodic in z , i.e., $u(x, y, \text{ZF}) = u(x, y, \text{ZS})$.
- NBDCND = 1
- if the solution is specified at $z = \text{ZS}$ and $z = \text{ZF}$.
- NBDCND = 2
- if the solution is specified at $z = \text{ZS}$ and the derivative of the solution with respect to z is specified at $z = \text{ZF}$.
- NBDCND = 3
- if the derivative of the solution with respect to z is specified at $z = \text{ZS}$ and $z = \text{ZF}$.
- NBDCND = 4
- if the derivative of the solution with respect to z is specified at $z = \text{ZS}$ and the solution is specified at $z = \text{ZF}$.
- Constraint:* $0 \leq \text{NBDCND} \leq 4$.
- 17: BDZS(LDIMF,M+1) – *real* array *Input*
- On entry:* the values of the derivative of the solution with respect to z at $z = \text{ZS}$. When NBDCND = 3 or 4, $\text{BDZS}(i, j) = u_z(x_i, y_j, \text{ZS})$, for $i = 1, 2, \dots, L + 1$; $j = 1, 2, \dots, M + 1$.
- When NBDCND has any other value, BDZS is not referenced.
- 18: BDZF(LDIMF,M+1) – *real* array *Input*
- On entry:* the values of the derivative of the solution with respect to z at $z = \text{ZF}$. When NBDCND = 2 or 3, $\text{BDZF}(i, j) = u_z(x_i, y_j, \text{ZF})$, for $i = 1, 2, \dots, L + 1$; $j = 1, 2, \dots, M + 1$.
- When NBDCND has any other value, BDZF is not referenced.
- 19: LAMBDA – *real* *Input*
- On entry:* the constant λ in the Helmholtz equation. For certain positive values of λ a solution to the differential equation may not exist, and close to these values the solution of the discretized problem will be extremely ill-conditioned. If $\lambda > 0$, then D03FAF will set IFAIL to 3, but will still attempt to find a solution. However, since in general the values of λ for which no solution exists cannot be predicted a priori, the user is advised to treat any results computed with $\lambda > 0$ with great caution.
- 20: LDIMF – INTEGER *Input*
- On entry:* the first dimension of the arrays F, BDYS, BDYF, BDZS and BDZF as declared in the (sub)program from which D03FAF is called.
- Constraint:* $\text{LDIMF} \geq L + 1$.
- 21: MDIMF – INTEGER *Input*
- On entry:* the second dimension of the array F and the first dimension of the arrays BDXS and BDXF as declared in the (sub)program from which D03FAF is called.
- Constraint:* $\text{MDIMF} \geq M + 1$.

22: F(LDIMF,MDIMF,N+1) – **real** array

Input/Output

On entry: the values of the right-side of the Helmholtz equation and boundary values (if any).

$$F(i, j, k) = f(x_i, y_j, z_k) \quad i = 2, 3, \dots, L, \quad j = 2, 3, \dots, M \text{ and } k = 2, 3, \dots, N.$$

On the boundaries F is defined by

LBDCND	$F(1, j, k)$	$F(L + 1, j, k)$	
0	$f(XS, y_j, z_k)$	$f(XS, y_j, z_k)$	
1	$u(XS, y_j, z_k)$	$u(XF, y_j, z_k)$	
2	$u(XS, y_j, z_k)$	$f(XF, y_j, z_k)$	$j = 1, 2, \dots, M + 1$
3	$f(XS, y_j, z_k)$	$f(XF, y_j, z_k)$	$k = 1, 2, \dots, N + 1$
4	$f(XS, y_j, z_k)$	$u(XF, y_j, z_k)$	

MBDCND	$F(i, 1, k)$	$F(i, M + 1, k)$	
0	$f(x_i, YS, z_k)$	$f(x_i, YS, z_k)$	
1	$u(x_i, YS, z_k)$	$u(x_i, YF, z_k)$	
2	$u(x_i, YS, z_k)$	$f(x_i, YF, z_k)$	$i = 1, 2, \dots, L + 1$
3	$f(x_i, YS, z_k)$	$f(x_i, YF, z_k)$	$k = 1, 2, \dots, N + 1$
4	$f(x_i, YS, z_k)$	$u(x_i, YF, z_k)$	

NBDCND	$F(i, j, 1)$	$F(i, j, N + 1)$	
0	$f(x_i, y_j, ZS)$	$f(x_i, y_j, ZS)$	
1	$u(x_i, y_j, ZS)$	$u(x_i, y_j, ZF)$	
2	$u(x_i, y_j, ZS)$	$f(x_i, y_j, ZF)$	$i = 1, 2, \dots, L + 1$
3	$f(x_i, y_j, ZS)$	$f(x_i, y_j, ZF)$	$j = 1, 2, \dots, M + 1$
4	$f(x_i, y_j, ZS)$	$u(x_i, y_j, ZF)$	

Note: if the table calls for both the solution u and the right-hand side f on a boundary, then the solution must be specified.

On exit: F contains the solution $u(i, j, k)$ of the finite difference approximation for the grid point (x_i, y_j, z_k) for $i = 1, 2, \dots, L + 1$, $j = 1, 2, \dots, M + 1$ and $k = 1, 2, \dots, N + 1$.

23: PERTRB – **real**

Output

On exit: PERTRB = 0, unless a solution to Poisson's equation ($\lambda = 0$) is required with a combination of periodic or derivative boundary conditions (LBDCND, MBDCND and NBDCND = 0 or 3). In this case a solution may not exist. PERTRB is a constant, calculated and subtracted from the array F, which ensures that a solution exists. D03FAF then computes this solution, which is a least-squares solution to the original approximation. This solution is not unique and is unnormalised. The value of PERTRB should be small compared to the right-hand side F, otherwise a solution has been obtained to an essentially different problem. This comparison should always be made to insure that a meaningful solution has been obtained.

24: W(LWRK) – **real** array

Workspace

25: LWRK – INTEGER

Input

On entry: the dimension of the array W as declared in the (sub)program from which D03FAF is called. $2 \times (N + 1) \times \max(L, M) + 3 \times L + 3 \times M + 4 \times N + 6$ is an upper bound on the required size of W. If LWRK is too small, the routine exits with IFAIL = 2, and if on entry IFAIL = 0 or IFAIL = -1, a message is output giving the exact value of LWRK required to solve the current problem.

26: IFAIL – INTEGER

Input/Output

On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

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, because for this routine the values of the output parameters may be useful even if $IFAIL \neq 0$ on exit, the recommended value is -1 . **When the value -1 or 1 is used it is essential to test the value of $IFAIL$ on exit.**

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, $XS \geq XF$,
 or $L < 5$,
 or $LBDCND < 0$,
 or $LBDCND > 4$,
 or $YS \geq YF$,
 or $M < 5$,
 or $MBDCND < 0$,
 or $MBDCND > 4$,
 or $ZS \geq ZF$,
 or $N < 5$,
 or $NBDCND < 0$,
 or $NBDCND > 4$,
 or $LDIMF < L + 1$,
 or $MDIMF < M + 1$.

$IFAIL = 2$

On entry, LWRK is too small.

$IFAIL = 3$

On entry, $\lambda > 0$.

7 Accuracy

Not applicable.

8 Further Comments

The execution time is roughly proportional to $L \times M \times N \times (\log_2 L + \log_2 M + 5)$, but also depends on input parameters LBDCND and MBDCND.

9 Example

The example solves the Helmholtz equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} + \lambda u = f(x, y, z)$$

for $(x, y, z) \in [0, 1] \times [0, 2\pi] \times [0, \frac{\pi}{2}]$ where $\lambda = -2$, and $f(x, y, z)$ is derived from the exact solution

$$u(x, y, z) = x^4 \sin y \cos z.$$

The equation is subject to the following boundary conditions, again derived from the exact solution given above.

$u(0, y, z)$ and $u(1, y, z)$ are prescribed (i.e., LBDCND = 1).

$u(x, 0, z) = u(x, 2\pi, z)$ (i.e., MBDCND = 0).

$u(x, y, 0)$ and $u_x(x, y, \frac{\pi}{2})$ are prescribed (i.e., NBDCND = 2).

9.1 Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*      D03FAF Example Program Text
*      Mark 14 Release.  NAG Copyright 1989.
*      .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          L, M, N, MAXLM, LDIMF, MDIMF, LWRK
      PARAMETER        (L=16, M=32, N=20, MAXLM=32, LDIMF=L+1, MDIMF=M+1,
+                      LWRK=2*(N+1)*MAXLM+3*L+3*M+4*N+6000)
*      .. Local Scalars ..
      real            DX, DY, DZ, ERROR, LAMBDA, PERTRB, PI, T, XF, XS,
+                      YF, YS, ZF, ZS
      INTEGER          I, IFAIL, J, K, LBDCND, MBDCND, NBDCND
*      .. Local Arrays ..
      real            BDXF(MDIMF,N+1), BDXS(MDIMF,N+1),
+                      BDYF(LDIMF,N+1), BDYS(LDIMF,N+1),
+                      BDZF(LDIMF,M+1), BDZS(LDIMF,M+1),
+                      F(LDIMF,MDIMF,N+1), W(LWRK), X(L+1), Y(M+1),
+                      Z(N+1)
*      .. External Functions ..
      real            X01AAF
      EXTERNAL          X01AAF
*      .. External Subroutines ..
      EXTERNAL          D03FAF
*      .. Intrinsic Functions ..
      INTRINSIC          ABS, COS, real, SIN
*      .. Executable Statements ..
      WRITE (NOUT,*) 'D03FAF Example Program Results'
      LAMBDA = -2.0e0
      XS = 0.0e0
      XF = 1.0e0
      LBDCND = 1
      YS = 0.0e0
      PI = X01AAF(PI)
      YF = 2.0e0*PI
      MBDCND = 0
      ZS = 0.0e0
      ZF = PI/2.0e0
      NBDCND = 2

*
*      Define the grid points for later use.
*
      DX = (XF-XS)/real(L)
      DO 20 I = 1, L + 1
          X(I) = XS + real(I-1)*DX
20  CONTINUE
      DY = (YF-YS)/real(M)
      DO 40 J = 1, M + 1
          Y(J) = YS + real(J-1)*DY
40  CONTINUE
      DZ = (ZF-ZS)/real(N)
      DO 60 K = 1, N + 1
          Z(K) = ZS + real(K-1)*DZ
60  CONTINUE

*
*      Define the array of derivative boundary values.
*
      DO 100 J = 1, M + 1
          DO 80 I = 1, L + 1
              BDZF(I,J) = -X(I)**4*SIN(Y(J))
80      CONTINUE
```

```

100 CONTINUE
*
*   Note that for this example all other boundary arrays are
*   dummy variables.
*
*   We define the function boundary values in the F array.
*
      DO 140 K = 1, N + 1
        DO 120 J = 1, M + 1
          F(1,J,K) = 0.0e0
          F(L+1,J,K) = SIN(Y(J))*COS(Z(K))
120      CONTINUE
140 CONTINUE
      DO 180 J = 1, M + 1
        DO 160 I = 1, L + 1
          F(I,J,1) = X(I)**4*SIN(Y(J))
160      CONTINUE
180 CONTINUE
*
*   Define the values of the right hand side of the Helmholtz
*   equation.
*
      DO 240 K = 2, N + 1
        DO 220 J = 1, M + 1
          DO 200 I = 2, L
            F(I,J,K) = 4.0e0*X(I)**2*(3.0e0-X(I)**2)*SIN(Y(J))
            +
            *COS(Z(K))
200      CONTINUE
220      CONTINUE
240 CONTINUE
*
*   Call D03FAF to generate and solve the finite difference equation.
*
      IFAIL = 0
*
      CALL D03FAF(XS,XF,L,LBDCND,BDXS,BDXF,YS,YF,M,MBDCND,BDYS,BDYF,ZS,
+              ZF,N,NBDCND,BDZS,BDZF,LAMBDA,LDIMF,MDIMF,F,PERTRB,W,
+              LWRK,IFAIL)
*
*   Compute discretization error. The exact solution to the
*   problem is
*
      U(X,Y,Z) = X**4*SIN(Y)*COS(Z)
*
      ERROR = 0.0e0
      DO 300 K = 1, N + 1
        DO 280 J = 1, M + 1
          DO 260 I = 1, L + 1
            T = ABS(F(I,J,K)-X(I)**4*SIN(Y(J))*COS(Z(K)))
            IF (T.GT.ERROR) ERROR = T
260      CONTINUE
280      CONTINUE
300 CONTINUE
      WRITE (NOUT,*)
      WRITE (NOUT,99999) 'Maximum component of discretization error =',
+      ERROR
      STOP
*
99999 FORMAT (1X,A,1P,e13.6)
      END

```

9.2 Program Data

None.

9.3 Program Results

D03FAF Example Program Results

Maximum component of discretization error = 5.176553E-04
