NAG Toolbox for MATLAB

d03ee

1 Purpose

d03ee discretizes a second-order elliptic partial differential equation (PDE) on a rectangular region.

2 Syntax

```
[a, rhs, ifail] = d03ee(xmin, xmax, ymin, ymax, pdef, bndy, ngx, ngy, scheme)
```

3 Description

d03ee discretizes a second-order linear elliptic partial differential equation of the form

$$\alpha(x,y)\frac{\partial^2 U}{\partial x^2}+\beta(x,y)\frac{\partial^2 U}{\partial x\partial y}+\gamma(x,y)\frac{\partial^2 U}{\partial y^2}+\delta(x,y)\frac{\partial U}{\partial x}+\epsilon(x,y)\frac{\partial U}{\partial y}+\phi(x,y)U=\psi(x,y) \tag{1}$$

on a rectangular region

$$x_A \le x \le x_B$$
$$y_A \le y \le y_B$$

subject to boundary conditions of the form

$$a(x,y)U+b(x,y)\frac{\partial U}{\partial n}=c(x,y)$$

where $\dfrac{\partial U}{\partial n}$ denotes the outward pointing normal derivative on the boundary. Equation (1) is said to be elliptic if

$$4\alpha(x,y)\gamma(x,y) \ge (\beta(x,y))^2$$

for all points in the rectangular region. The linear equations produced are in a form suitable for passing directly to the multigrid function d03ed.

The equation is discretized on a rectangular grid, with n_x grid points in the x-direction and n_y grid points in the y-direction. The grid spacing used is therefore

$$h_x=(x_B-x_A)/(n_x-1)$$
$$h_y=(y_B-y_A)/(n_y-1)$$

and the co-ordinates of the grid points $\left(x_i,y_j\right)$ are

$$x_i=x_A+(i-1)h_x, \qquad i=1,2,\ldots,n_x,$$
$$y_j=y_A+(j-1)h_y, \qquad j=1,2,\ldots,n_y.$$

At each grid point $\left(x_i,y_j\right)$ six neighbouring grid points are used to approximate the partial differential equation, so that the equation is discretized on the seven-point stencil shown in Figure 1.

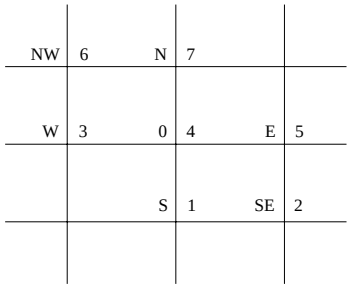

Figure 1

For convenience the approximation u_{ij} to the exact solution $U\left(x_i, y_j\right)$ is denoted by u_{O}, and the neighbouring approximations are labelled according to points of the compass as shown. Where numerical labels for the seven points are required, these are also shown.

The following approximations are used for the second derivatives:

$$\begin{aligned}
\frac{\partial^2 U}{\partial x^2} &\simeq \tfrac{1}{h_x^2}(u_{\mathrm{E}} - 2u_{\mathrm{O}} + u_{\mathrm{W}}) \\
\frac{\partial^2 U}{\partial y^2} &\simeq \tfrac{1}{h_y^2}(u_{\mathrm{N}} - 2u_{\mathrm{O}} + u_{\mathrm{S}}) \\
\frac{\partial^2 U}{\partial x \partial y} &\simeq \tfrac{1}{2h_x h_y}(u_{\mathrm{N}} - u_{\mathrm{NW}} + u_{\mathrm{E}} - 2u_{\mathrm{O}} + u_{\mathrm{W}} - u_{\mathrm{SE}} + u_{\mathrm{S}}).
\end{aligned}$$

Two possible schemes may be used to approximate the first derivatives:

Central Differences

$$\begin{aligned}
\frac{\partial U}{\partial x} &\simeq \tfrac{1}{2h_x}(u_{\mathrm{E}} - u_{\mathrm{W}}) \\
\frac{\partial U}{\partial y} &\simeq \tfrac{1}{2h_y}(u_{\mathrm{N}} - u_{\mathrm{S}})
\end{aligned}$$

Upwind Differences

$$\begin{aligned}
\frac{\partial U}{\partial x} &\simeq \tfrac{1}{h_x}(u_{\mathrm{O}} - u_{\mathrm{W}}) \quad \text{if} \quad \delta(x,y) > 0 \\
\frac{\partial U}{\partial x} &\simeq \tfrac{1}{h_x}(u_{\mathrm{E}} - u_{\mathrm{O}}) \quad \text{if} \quad \delta(x,y) < 0 \\
\frac{\partial U}{\partial y} &\simeq \tfrac{1}{h_y}(u_{\mathrm{N}} - u_{\mathrm{O}}) \quad \text{if} \quad \epsilon(x,y) > 0 \\
\frac{\partial U}{\partial y} &\simeq \tfrac{1}{h_y}(u_{\mathrm{O}} - u_{\mathrm{S}}) \quad \text{if} \quad \epsilon(x,y) < 0.
\end{aligned}$$

Central differences are more accurate than upwind differences, but upwind differences may lead to a more diagonally dominant matrix for those problems where the coefficients of the first derivatives are significantly larger than the coefficients of the second derivatives.

The approximations used for the first derivatives may be written in a more compact form as follows:

$$\begin{aligned}
\frac{\partial U}{\partial x} &\simeq \tfrac{1}{2h_x}((k_x - 1)u_{\mathrm{W}} - 2k_x u_{\mathrm{O}} + (k_x + 1)u_{\mathrm{E}}) \\
\frac{\partial U}{\partial y} &\simeq \tfrac{1}{2h_y}\left(\left(k_y - 1\right)u_{\mathrm{S}} - 2k_y u_{\mathrm{O}} + \left(k_y + 1\right)u_{\mathrm{N}}\right)
\end{aligned}$$

where $k_x = \operatorname{sign}\delta$ and $k_y = \operatorname{sign}\epsilon$ for upwind differences, and $k_x = k_y = 0$ for central differences.

At all points in the rectangular domain, including the boundary, the coefficients in the partial differential equation are evaluated by calling the user-supplied (sub)program **pdef**, and applying the approximations. This leads to a seven-diagonal system of linear equations of the form:

$$\begin{array}{ccccccc} & A_{ij}^6 u_{i-1,j+1} & + & A_{ij}^7 u_{i,j+1} & & & \\ + & A_{ij}^3 u_{i-1,j} & + & A_{ij}^4 u_{ij} & + & A_{ij}^5 u_{i+1,j} & \\ & & + & A_{ij}^1 u_{i,j-1} & + & A_{ij}^2 u_{i+1,j-1} = f_{ij}, & i = 1,2,\ldots,n_x; j = 1,2,\ldots,n_y, \end{array}$$

where the coefficients are given by

$$\begin{aligned} A_{ij}^1 &= \beta\left(x_i,y_j\right)\tfrac{1}{2h_xh_y} + \gamma\left(x_i,y_j\right)\tfrac{1}{h_y^2} + \epsilon\left(x_i,y_j\right)\tfrac{1}{2h_y}(k_y-1) \\ A_{ij}^2 &= -\beta\left(x_i,y_j\right)\tfrac{1}{2h_xh_y} \\ A_{ij}^3 &= \alpha\left(x_i,y_j\right)\tfrac{1}{h_x^2} + \beta\left(x_i,y_j\right)\tfrac{1}{2h_xh_y} + \delta\left(x_i,y_j\right)\tfrac{1}{2h_x}(k_x-1) \\ A_{ij}^4 &= -\alpha\left(x_i,y_j\right)\tfrac{2}{h_x^2} - \beta\left(x_i,y_j\right)\tfrac{1}{h_xh_y} - \gamma\left(x_i,y_j\right)\tfrac{2}{h_y^2} - \delta\left(x_i,y_j\right)\tfrac{k_y}{h_x} - \epsilon\left(x_i,y_j\right)\tfrac{k_y}{h_y} - \phi\left(x_i,y_j\right) \\ A_{ij}^5 &= \alpha\left(x_i,y_j\right)\tfrac{1}{h_x^2} + \beta\left(x_i,y_j\right)\tfrac{1}{2h_xh_y} + \delta\left(x_i,y_j\right)\tfrac{1}{2h_x}(k_x+1) \\ A_{ij}^6 &= -\beta\left(x_i,y_j\right)\tfrac{1}{2h_xh_y} \\ A_{ij}^7 &= \beta\left(x_i,y_j\right)\tfrac{1}{2h_xh_y} + \gamma\left(x_i,y_j\right)\tfrac{1}{h_y^2} + \epsilon\left(x_i,y_j\right)\tfrac{1}{2h_y}(k_y+1) \\ f_{ij} &= \psi\left(x_i,y_j\right) \end{aligned}$$

These equations then have to be modified to take account of the boundary conditions. These may be Dirichlet (where the solution is given), Neumann (where the derivative of the solution is given), or mixed (where a linear combination of solution and derivative is given).

If the boundary conditions are Dirichlet, there are an infinity of possible equations which may be applied:

$$\mu u_{ij} = \mu f_{ij}, \mu \neq 0. \tag{2}$$

If d03ed is used to solve the discretized equations, it turns out that the choice of μ can have a dramatic effect on the rate of convergence, and the obvious choice $\mu = 1$ is not the best. Some choices may even cause the multigrid method to fail altogether. In practice it has been found that a value of the same order as the other diagonal elements of the matrix is best, and the following value has been found to work well in practice:

$$\mu = \min_{ij}\left(-\left\{\frac{2}{h_x^2}+\frac{2}{h_y^2}\right\}, A_{ij}^4\right).$$

If the boundary conditions are either mixed or Neumann (i.e., $B \neq 0$ on return from the user-supplied (sub)program **bndy**), then one of the points in the seven-point stencil lies outside the domain. In this case the normal derivative in the boundary conditions is used to eliminate the 'fictitious' point, u_{outside}:

$$\frac{\partial U}{\partial n} \simeq \frac{1}{2h}(u_{\text{outside}} - u_{\text{inside}}). \tag{3}$$

It should be noted that if the boundary conditions are Neumann and $\phi(x,y) \equiv 0$, then there is no unique solution. The function returns with **ifail** $= 5$ in this case, and the seven-diagonal matrix is singular.

The four corners are treated separately. The user-supplied (sub)program **bndy** is called twice, once along each of the edges meeting at the corner. If both boundary conditions at this point are Dirichlet and the prescribed solution values agree, then this value is used in an equation of the form (2). If the prescribed solution is discontinuous at the corner, then the average of the two values is used. If one boundary condition is Dirichlet and the other is mixed, then the value prescribed by the Dirichlet condition is used in an equation of the form given above. Finally, if both conditions are mixed or Neumann, then two 'fictitious' points are eliminated using two equations of the form (3).

It is possible that equations for which the solution is known at all points on the boundary, have coefficients which are not defined on the boundary. Since this function calls user-supplied (sub)program **pdef** at **all** points in the domain, including boundary points, arithmetic errors may occur in **pdef** which this function cannot trap. If you have an equation with Dirichlet boundary conditions (i.e., $B = 0$ at all points on the boundary), but with PDE coefficients which are singular on the boundary, then d03ed could be called directly only using interior grid points at your discretization.

After the equations have been set up as described above, they are checked for diagonal dominance. That is to say,

$$|A_{ij}^4| > \sum_{k \neq 4} |A_{ij}^k|, \qquad i = 1,2,\ldots,n_x; j = 1,2,\ldots,n_y.$$

If this condition is not satisfied then the function returns with **ifail** $= 6$. The multigrid functiond03ed may still converge in this case, but if the coefficients of the first derivatives in the partial differential equation are large compared with the coefficients of the second derivative, you should consider using upwind differences (**scheme** = 'U').

Since this function is designed primarily for use with d03ed, this document should be read in conjunction with the document for that function.

4 References

Wesseling P 1982a MGD1 – A robust and efficient multigrid method *Multigrid Methods. Lecture Notes in Mathematics* **960** 614–630 Springer–Verlag

5 Parameters

5.1 Compulsory Input Parameters

1: **xmin – double scalar**

2: **xmax – double scalar**

The lower and upper x co-ordinates of the rectangular region respectively, x_A and x_B.

Constraint: **xmin** $<$ **xmax**.

3: **ymin – double scalar**

4: **ymax – double scalar**

The lower and upper y co-ordinates of the rectangular region respectively, y_A and y_B.

Constraint: **ymin** $<$ **ymax**.

5: **pdef – string containing name of m-file**

pdef must evaluate the functions $\alpha(x,y)$, $\beta(x,y)$, $\gamma(x,y)$, $\delta(x,y)$, $\epsilon(x,y)$, $\phi(x,y)$ and $\psi(x,y)$ which define the equation at a general point (x,y).

Its specification is:

```
[alpha, beta, gamma, delta, epslon, phi, psi] = pdef(x, y)
```

Input Parameters

1: **x – double scalar**

2: **y – double scalar**

The x and y co-ordinates of the point at which the coefficients of the partial differential equation are to be evaluated.

Output Parameters

1: **alpha – double scalar**
2: **beta – double scalar**
3: **gamma – double scalar**
4: **delta – double scalar**
5: **epslon – double scalar**
6: **phi – double scalar**
7: **psi – double scalar**

alpha, **beta**, **gamma**, **delta**, **epslon**, **phi** and **psi** must be set to the values of $\alpha(x,y)$, $\beta(x,y)$, $\gamma(x,y)$, $\delta(x,y)$, $\epsilon(x,y)$, $\phi(x,y)$ and $\psi(x,y)$ respectively at the point specified by **x** and **y**.

6: **bndy – string containing name of m-file**

bndy must evaluate the functions $a(x,y)$, $b(x,y)$, and $c(x,y)$ involved in the boundary conditions.

Its specification is:

```
[a, b, c] = bndy(x, y, ibnd)
```

Input Parameters

1: **x – double scalar**
2: **y – double scalar**

The x and y co-ordinates of the point at which the boundary conditions are to be evaluated.

3: **ibnd – int32 scalar**

Specifies on which boundary the point (**x**,**y**) lies. **ibnd** $= 0$, 1, 2 or 3 according as the point lies on the bottom, right, top or left boundary.

Output Parameters

1: **a – double scalar**
2: **b – double scalar**
3: **c – double scalar**

a, **b** and **c** must be set to the values of the functions appearing in the boundary conditions.

7: **ngx – int32 scalar**
8: **ngy – int32 scalar**

The number of interior grid points in the x- and y-directions respectively, n_x and n_y. If the seven-diagonal equations are to be solved by d03ed, then **ngx** $- 1$ and **ngy** $- 1$ should preferably be divisible by as high a power of 2 as possible.

Constraint: **ngx** ≥ 3, **ngy** ≥ 3.

9: **scheme – string**

The type of approximation to be used for the first derivatives which occur in the partial differential equation.

scheme = 'C'

Central differences are used.

scheme = 'U'

Upwind differences are used.

Constraint: **scheme** = 'C' or 'U'.

Note: generally speaking, if at least one of the coefficients multiplying the first derivatives (**delta** or **epslon** as returned by user-supplied (sub)program **pdef**) are large compared with the coefficients multiplying the second derivatives, then upwind differences may be more appropriate. Upwind differences are less accurate than central differences, but may result in more rapid convergence for strongly convective equations. The easiest test is to try both schemes

5.2 Optional Input Parameters

None.

5.3 Input Parameters Omitted from the MATLAB Interface

lda

5.4 Output Parameters

1: **a**(**lda,7**) – **double array**

$\mathbf{a}(i,j)$, for $i = 1,2,\ldots,\mathbf{ngx} \times \mathbf{ngy}$ and $j = 1,2,\ldots,7$, contains the seven-diagonal linear equations produced by the discretization described above. If $\mathbf{lda} > \mathbf{ngx} \times \mathbf{ngy}$, the remaining elements are not referenced by the function, but if $\mathbf{lda} \geq (4 \times (\mathbf{ngx}+1) \times (\mathbf{ngy}+1))/3$ then the array **a** can be passed directly to d03ed, where these elements are used as workspace.

2: **rhs**(**lda**) – **double array**

The first $\mathbf{ngx} \times \mathbf{ngy}$ elements contain the right-hand sides of the seven-diagonal linear equations produced by the discretization described above. If $\mathbf{lda} > \mathbf{ngx} \times \mathbf{ngy}$, the remaining elements are not referenced by the function, but if $\mathbf{lda} \geq (4 \times (\mathbf{ngy}+1) \times (\mathbf{ngy}+1))/3$ then the array **rhs** can be passed directly to d03ed, where these elements are used as workspace.

3: **ifail** – **int32 scalar**

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

6 Error Indicators and Warnings

Note: d03ee may return useful information for one or more of the following detected errors or warnings.

ifail = 1

On entry, $\mathbf{xmin} \geq \mathbf{xmax}$,
or $\mathbf{ymin} \geq \mathbf{ymax}$,
or $\mathbf{ngx} < 3$,
or $\mathbf{ngy} < 3$,
or $\mathbf{lda} < \mathbf{ngx} \times \mathbf{ngy}$,
or **scheme** is not one of 'C' or 'U'.

ifail = 2

At some point on the boundary there is a derivative in the boundary conditions (**b** $\neq 0$ on return from user-supplied (sub)program **bndy**) and there is a nonzero coefficient of the mixed derivative $\frac{\partial^2 U}{\partial x \partial y}$ (**beta** $\neq 0$ on return from the user-supplied (sub)program **pdef**).

ifail = 3

A null boundary has been specified, i.e., at some point both **a** and **b** are zero on return from a call to user-supplied (sub)program **bndy**.

ifail = 4

The equation is not elliptic, i.e., $4 \times \textbf{alpha} \times \textbf{gamma} < \textbf{beta}^2$ after a call to user-supplied (sub)program **pdef**. The discretization has been completed, but the convergence of d03ed cannot be guaranteed.

ifail = 5

The boundary conditions are purely Neumann (only the derivative is specified) and there is, in general, no unique solution.

ifail = 6

The equations were not diagonally dominant. (See Section 3.)

7 Accuracy

Not applicable.

8 Further Comments

If this function is used as a pre-processor to the multigrid function d03ed it should be noted that the rate of convergence of that function is strongly dependent upon the number of levels in the multigrid scheme, and thus the choice of **ngx** and **ngy** is very important.

9 Example

d03ee_bndy.m

```
function [a, b, c] = bndy(x, y, ibnd)
      if (ibnd == 2 || ibnd == 1)
%        Solution prescribed
         a = 1.0d0;
         b = 0.0d0;
         c = sin(x)*sin(y);
      elseif (ibnd == 0)
%        Derivative prescribed
         a = 0.0d0;
         b = 1.0d0;
         c = -sin(x);
      elseif (ibnd == 3)
%        Derivative prescribed
         a = 0.0d0;
         b = 1.0d0;
         c = -sin(y);
      end
```

d03ee_pdef.m

```
function [alpha, beta, gamma, delta, epsilon, phi, psi] = pdef(x,y)

      alpha = 1;
      beta = 0;
      gamma = 1;
      delta = 50;
      epsilon = 50;
      phi = 0;
```

```
      psi = (-alpha-gamma+phi)*sin(x)*sin(y) + beta*cos(x)*cos(y) + ...
                                             delta*cos(x)*sin(y)  +
epsilon*sin(x)*cos(y);
```

```
xmin = 0;
xmax = 1;
ymin = 0;
ymax = 1;
ngx = int32(9);
ngy = int32(9);
scheme = 'Central';
[a, rhs, ifail] = ...
     d03ee(xmin, xmax, ymin, ymax, 'd03ee_pdef', 'd03ee_bndy', ngx, ngy,
scheme);
 a(1:81,:)
 rhs(1:81)
```

```
Warning: d03ee returned a non-zero warning or error indicator (6)
ans =
     0     0     0  -256   128     0   128
     0     0  -136  -256   264     0   128
     0     0  -136  -256   264     0   128
     0     0  -136  -256   264     0   128
     0     0  -136  -256   264     0   128
     0     0  -136  -256   264     0   128
     0     0  -136  -256   264     0   128
     0     0  -136  -256   264     0   128
     0     0     0  -256     0     0     0
  -136     0     0  -256   128     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
     0     0     0  -256     0     0     0
  -136     0     0  -256   128     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
     0     0     0  -256     0     0     0
  -136     0     0  -256   128     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
     0     0     0  -256     0     0     0
  -136     0     0  -256   128     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
     0     0     0  -256     0     0     0
  -136     0     0  -256   128     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
```

```
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
     0     0     0  -256     0     0     0
  -136     0     0  -256   128     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
     0     0     0  -256     0     0     0
  -136     0     0  -256   128     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
  -136     0  -136  -256   264     0   264
     0     0     0  -256     0     0     0
     0     0     0  -256     0     0     0
     0     0     0  -256     0     0     0
     0     0     0  -256     0     0     0
     0     0     0  -256     0     0     0
     0     0     0  -256     0     0     0
     0     0     0  -256     0     0     0
     0     0     0  -256     0     0     0
     0     0     0  -256     0     0     0
     0     0     0  -256     0     0     0
ans =
         0
    1.9948
    3.9585
    5.8604
    7.6708
    9.3616
   10.9062
   12.2807
         0
    1.9948
   12.3391
   18.2519
   23.8799
   29.1353
   33.9360
   38.2072
   41.8822
  -26.8570
    3.9585
   18.2519
   23.8489
   29.0736
   33.8447
   38.0877
   41.7363
   44.7336
  -53.2949
    5.8604
   23.8799
   29.0736
   33.8136
   38.0260
   41.6449
   44.6140
   46.8870
  -78.9012
    7.6708
```

```
   29.1353
   33.8447
   38.0260
   41.6139
   44.5524
   46.7956
   48.3087
 -103.2762
    9.3616
   33.9360
   38.0877
   41.6449
   44.5524
   46.7646
   48.2470
   48.9766
 -126.0396
   10.9062
   38.2072
   41.7363
   44.6140
   46.7956
   48.2470
   48.9455
   48.8802
 -146.8363
   12.2807
   41.8822
   44.7336
   46.8870
   48.3087
   48.9766
   48.8802
   48.0211
 -165.3416
         0
  -26.8570
  -53.2949
  -78.9012
 -103.2762
 -126.0396
 -146.8363
 -165.3416
 -181.2668
```