NAG Toolbox

nag_pde_2d_ellip_discret (d03ee)

1 Purpose

nag_pde_2d_ellip_discret (d03ee) discretizes a second-order elliptic partial differential equation (PDE) on a rectangular region.

2 Syntax

```
[a, rhs, ifail] = nag_pde_2d_ellip_discret(xmin, xmax, ymin, ymax, pdef, bndy, ngx, ngy, scheme)
[a, rhs, ifail] = d03ee(xmin, xmax, ymin, ymax, pdef, bndy, ngx, ngy, scheme)
```

3 Description

nag_pde_2d_ellip_discret (d03ee) discretizes a second-order linear elliptic partial differential equation of the form

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

on a rectangular region

$$\begin{aligned} x_A \le x \le x_B \\ y_A \le y \le y_B \end{aligned}$$

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 nag_pde_2d_ellip_mgrid (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

$$\begin{aligned} h_x &= (x_B - x_A)/(n_x - 1) \\ h_y &= (y_B - y_A)/(n_y - 1) \end{aligned}$$

and the coordinates of the grid points (x_i, y_j) are

$$\begin{aligned} x_i &= x_A + (i-1)h_x, \quad i = 1,2,\ldots,n_x, \\ y_j &= y_A + (j-1)h_y, \quad j = 1,2,\ldots,n_y. \end{aligned}$$

At each grid point (x_i, y_j) 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.

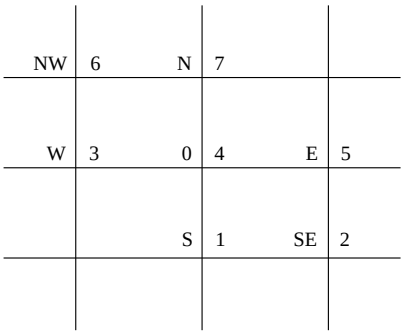

Figure 1

For convenience the approximation u_{ij} to the exact solution $U(x_i, y_j)$ 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:

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

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:

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

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 **pdef**, and applying the approximations. This leads to a seven-diagonal system of linear equations of the form:

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

where the coefficients are given by

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

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 nag_pde_2d_ellip_mgrid (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:

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

If the boundary conditions are either mixed or Neumann (i.e., $B \neq 0$ on return from **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. **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 **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 nag_pde_2d_ellip_mgrid (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,

$$\left|A_{ij}^4\right| > \sum_{k\neq 4}\left|A_{ij}^k\right|, \quad i = 1,2,\ldots,n_x \text{ and } j = 1,2,\ldots,n_y.$$

If this condition is not satisfied then the function returns with **ifail** $= 6$. The multigrid functionnag_ pde_2d_ellip_mgrid (d03ed) 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 nag_pde_2d_ellip_mgrid (d03ed), this document should be read in conjunction with the document for that function.

4 References

Wesseling P (1982) 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** – REAL (KIND=nag_wp)
2: **xmax** – REAL (KIND=nag_wp)

The lower and upper x coordinates of the rectangular region respectively, x_A and x_B.

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

3: **ymin** – REAL (KIND=nag_wp)
4: **ymax** – REAL (KIND=nag_wp)

The lower and upper y coordinates of the rectangular region respectively, y_A and y_B.

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

5: **pdef** – SUBROUTINE, supplied by the user.

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).

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

Input Parameters

1: **x** – REAL (KIND=nag_wp)
2: **y** – REAL (KIND=nag_wp)

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

Output Parameters

1: **alpha** – REAL (KIND=nag_wp)
2: **beta** – REAL (KIND=nag_wp)
3: **gamma** – REAL (KIND=nag_wp)
4: **delta** – REAL (KIND=nag_wp)
5: **epslon** – REAL (KIND=nag_wp)
6: **phi** – REAL (KIND=nag_wp)
7: **psi** – REAL (KIND=nag_wp)

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** – SUBROUTINE, supplied by the user.

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

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

Input Parameters

1: **x** – REAL (KIND=nag_wp)
2: **y** – REAL (KIND=nag_wp)

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

3: **ibnd** – INTEGER

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** – REAL (KIND=nag_wp)
2: **b** – REAL (KIND=nag_wp)
3: **c** – REAL (KIND=nag_wp)

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

7: **ngx** – INTEGER
8: **ngy** – INTEGER

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 nag_pde_2d_ellip_mgrid (d03ed), then **ngx** -1 and **ngy** -1 should preferably be divisible by as high a power of 2 as possible.

Constraints:

ngx ≥ 3;
ngy ≥ 3.

9: **scheme** – CHARACTER(1)

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 **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 Output Parameters

1: **a**$(lda, \mathbf{7})$ – REAL (KIND=nag_wp) array

a(i,j), for $i = 1,2,\ldots,$ **ngx** $\times$ **ngy** and $j = 1,2,\ldots,7$, contains the seven-diagonal linear equations produced by the discretization described above. If $lda >$ **ngx** $\times$ **ngy**, the remaining elements are not referenced by the function, but if $lda \geq (4 \times (\mathbf{ngx}+1) \times (\mathbf{ngy}+1))/3$ then the array **a** can be passed directly to nag_pde_2d_ellip_mgrid (d03ed), where these elements are used as workspace.

2: **rhs**(lda) – REAL (KIND=nag_wp) array

The first **ngx** $\times$ **ngy** elements contain the right-hand sides of the seven-diagonal linear equations produced by the discretization described above. If $lda >$ **ngx** $\times$ **ngy**, the remaining elements are not referenced by the function, but if $lda \geq (4 \times (\mathbf{ngy}+1) \times (\mathbf{ngy}+1))/3$ then the array **rhs** can be passed directly to nag_pde_2d_ellip_mgrid (d03ed), where these elements are used as workspace.

3: **ifail** – INTEGER

ifail = 0 unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Note: nag_pde_2d_ellip_discret (d03ee) may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the function:

ifail = 1

On entry, **xmin** $\geq$ **xmax**,
or **ymin** $\geq$ **ymax**,
or **ngx** < 3,
or **ngy** < 3,
or $lda <$ **ngx** $\times$ **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 **bndy**) and there is a nonzero coefficient of the mixed derivative $\dfrac{\partial^2 U}{\partial x \partial y}$ (**beta** $\neq 0$ on return from **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 **bndy**.

ifail $= 4$ (*warning*)

The equation is not elliptic, i.e., $4 \times$ **alpha** $\times$ **gamma** $<$ **beta**^{2} after a call to **pdef**. The discretization has been completed, but the convergence of nag_pde_2d_ellip_mgrid (d03ed) cannot be guaranteed.

ifail $= 5$ (*warning*)

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

ifail $= 6$ (*warning*)

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

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

7 Accuracy

Not applicable.

8 Further Comments

If this function is used as a preprocessor to the multigrid function nag_pde_2d_ellip_mgrid (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

The program solves the elliptic partial differential equation

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + 50\left\{\frac{\partial U}{\partial x} + \frac{\partial U}{\partial y}\right\} = f(x, y)$$

on the unit square $0 \leq x, \; y \leq 1$, with boundary conditions

$\dfrac{\partial U}{\partial n}$ given on $x = 0$ and $y = 0$,

U given on $x = 1$ and $y = 1$.

The function $f(x, y)$ and the exact form of the boundary conditions are derived from the exact solution $U(x, y) = \sin x \sin y$.

The equation is first solved using central differences. Since the coefficients of the first derivatives are large, the linear equations are not diagonally dominated, and convergence is slow. The equation is solved a second time with upwind differences, showing that convergence is more rapid, but the solution is less accurate.

9.1 Program Text

```
     function d03ee_example

% d03ee example: Solve elliptic equation by discretising using d02ee
%                and solving the disretised system using d03ed.

fprintf('d03ee example results\n\n');

%  u= sin(kx)sin(ly)
global k l;
k = 4; l = 5;

% Discretise on unit square usind d03ee
xmin = 0; xmax = 1;
ymin = 0; ymax = 1;
ngx = nag_int(33); ngy = ngx;
scheme = 'Upwind';
[a, rhs, ifail] = ...
d03ee( ...
       xmin, xmax, ymin, ymax, @pdef, @bndy, ngx, ngy, scheme);

% Solve using d03ed
maxit = nag_int(100); acc = 0.0001; iout = nag_int(0);
ub = zeros(ngx*ngy, 1);
[a, rhs, ub, resid, u, numit, ifail] = ...
d03ed( ...
       ngx, ngy, a, rhs, ub, maxit, acc, iout);

fprintf('Number of iterations to solution = %2d.\n', numit);
fprintf('Maximum residual                 %8.1e\n',resid(numit));

% plot solution
u2 = u(1:ngx*ngy);
umat = reshape(u2,ngx,ngy);
hx = 1/double(ngx+1);
hy = hx;
x(1:ngx) = hx:hx:1-hx;
y(1:ngy) = hy:hy:1-hy;
[xs,ys] = meshgrid(y,x);

fig1 = figure;
meshc(xs,ys,umat);

xlabel('x'); ylabel('y'); zlabel ('u(x,y)');
exact = sprintf('u = sin %dx sin %dy',k,l);
title({'d03ee example:  U_{xx}+U_{yy}+50(U_x+U_y) = f;',exact});
view(-24,28);


function [alpha, beta, gamma, delta, epsilon, phi, psi] = pdef(x,y)
  global k l;
  u = sin(k*x)*cos(l*y);
  ux = k*cos(k*x)*sin(l*y); uy = l*sin(k*x)*cos(l*y);
  uxx = -k*k*u; uyy = -l*l*u;
  uxy = k*l*cos(k*x)*cos(l*y);
  alpha = 1; beta = 0; gamma = 1; delta = 50; epsilon = 50;
  phi = 0;
  psi = alpha*uxx + beta*uxy + gamma*uyy + delta*ux + epsilon*uy + phi*u;

function [a, b, c] = bndy(x, y, ibnd)
  global k l;
  u = sin(k*x)*sin(l*y);
  if (ibnd == 2 || ibnd == 1)
    % Solution prescribed
    a = 1;
    b = 0;
    c = u;
  elseif (ibnd == 0)
    % Derivative prescribed: uy(y=0) = l*sin(k*x)
    a = 0;
```

```
    b = 1;
    c = -l*sin(k*x);
  elseif (ibnd == 3)
    % Derivative prescribed: ux(x=0) = k*sin(l*y)
    a = 0;
    b = 1;
    c = -k*sin(l*y);
  end
```

9.2 Program Results

```
     d03ee example results

Number of iterations to solution =  5.
Maximum residual                    -3.4e-09
```