NAG Toolbox

nag_pde_2d_ellip_mgrid (d03ed)

1 Purpose

nag_pde_2d_ellip_mgrid (d03ed) solves seven-diagonal systems of linear equations which arise from the discretization of an elliptic partial differential equation on a rectangular region. This function uses a multigrid technique.

2 Syntax

```
[a, rhs, ub, us, u, numit, ifail] = nag_pde_2d_ellip_mgrid(ngx, ngy, a, rhs, ub, maxit, acc, iout)
[a, rhs, ub, us, u, numit, ifail] = d03ed(ngx, ngy, a, rhs, ub, maxit, acc, iout)
```

3 Description

nag_pde_2d_ellip_mgrid (d03ed) solves, by multigrid iteration, the seven-point scheme

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

which arises from the discretization of an elliptic partial differential equation of the form

$$\alpha(x,y)U_{xx} + \beta(x,y)U_{xy} + \gamma(x,y)U_{yy} + \delta(x,y)U_x + \epsilon(x,y)U_y + \phi(x,y)U = \psi(x,y)$$

and its boundary conditions, defined on a rectangular region. This we write in matrix form as

$$Au = f.$$

The algorithm is described in separate reports by Wesseling (1982a), Wesseling (1982b) and McCarthy (1983).

Systems of linear equations, matching the seven-point stencil defined above, are solved by a multigrid iteration. An initial estimate of the solution must be provided by you. A zero guess may be supplied if no better approximation is available.

A ‘smoother’ based on incomplete Crout decomposition is used to eliminate the high frequency components of the error. A restriction operator is then used to map the system on to a sequence of coarser grids. The errors are then smoothed and prolongated (mapped onto successively finer grids). When the finest cycle is reached, the approximation to the solution is corrected. The cycle is repeated for **maxit** iterations or until the required accuracy, **acc**, is reached.

nag_pde_2d_ellip_mgrid (d03ed) will automatically determine the number l of possible coarse grids, ‘levels’ of the multigrid scheme, for a particular problem. In other words, nag_pde_2d_ellip_mgrid (d03ed) determines the maximum integer l so that n_x and n_y can be expressed in the form

$$n_x = m2^{l-1} + 1, \quad n_y = n2^{l-1} + 1, \qquad \text{with} \quad m \geq 2 \text{ and } n \geq 2.$$

It should be noted that the rate of convergence improves significantly with the number of levels used (see McCarthy (1983)), so that n_x and n_y should be carefully chosen so that $n_x - 1$ and $n_y - 1$ have factors of the form 2^l, with l as large as possible. For good convergence the integer l should be at least 2.

nag_pde_2d_ellip_mgrid (d03ed) has been found to be robust in application, but being an iterative method the problem of divergence can arise. For a strictly diagonally dominant matrix A

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

no such problem is foreseen. The diagonal dominance of A is not a necessary condition, but should this condition be strongly violated then divergence may occur. The quickest test is to try the function.

4 References

McCarthy G J (1983) Investigation into the multigrid code MGD1 *Report AERE-R 10889* Harwell

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

Wesseling P (1982b) Theoretical aspects of a multigrid method *SIAM J. Sci. Statist. Comput.* **3** 387–407

5 Parameters

5.1 Compulsory Input Parameters

1: **ngx** – INTEGER

The number of interior grid points in the x-direction, n_x. **ngx** -1 should preferably be divisible by as high a power of 2 as possible.

Constraint: **ngx** ≥ 3.

2: **ngy** – INTEGER

The number of interior grid points in the y-direction, n_y. **ngy** -1 should preferably be divisible by as high a power of 2 as possible.

Constraint: **ngy** ≥ 3.

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

a$(i+(j-1)\times$ **ngx**$, k)$ must be set to $\mathbf{a}_{ij}^{k}$, for $i = 1,2,\ldots,$ **ngx**, $j = 1,2,\ldots,$ **ngy** and $k = 1,2,\ldots,7$.

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

rhs$(i+(j-1)\times$ **ngx**$)$ must be set to f_{ij}, for $i = 1,2,\ldots,$ **ngx** and $j = 1,2,\ldots,$ **ngy**.

5: **ub**(**ngx** $\times$ **ngy**) – REAL (KIND=nag_wp) array

ub$(i+(j-1)\times$ **ngx**$)$ must be set to the initial estimate for the solution u_{ij}.

6: **maxit** – INTEGER

The maximum permitted number of multigrid iterations. If **maxit** $= 0$, no multigrid iterations are performed, but the coarse-grid approximations and incomplete Crout decompositions are computed, and may be output if **iout** is set accordingly.

Constraint: **maxit** ≥ 0.

7: **acc** – REAL (KIND=nag_wp)

The required tolerance for convergence of the residual 2-norm:

$$\|r\|_2 = \sqrt{\sum_{k=1}^{\mathbf{ngx}\times\mathbf{ngy}} (r_k)^2}$$

where $r = f - Au$ and u is the computed solution. Note that the norm is not scaled by the

number of equations. The function will stop after fewer than **maxit** iterations if the residual 2-norm is less than the specified tolerance. (If **maxit** > 0, at least one iteration is always performed.)

If on entry **acc** $= 0.0$, then the ***machine precision*** is used as a default value for the tolerance; if **acc** > 0.0, but **acc** is less than the ***machine precision***, then the function will stop when the residual 2-norm is less than the ***machine precision*** and **ifail** will be set to 4.

Constraint: **acc** ≥ 0.0.

8: **iout** – INTEGER

Controls the output of printed information to the advisory message unit as returned by nag_file_set_unit_advisory (x04ab):

iout $= 0$
No output.

iout $= 1$
The solution u_{ij}, for $i = 1, 2, \ldots,$ **ngx** and $j = 1, 2, \ldots,$ **ngy**.

iout $= 2$
The residual 2-norm after each iteration, with the reduction factor over the previous iteration.

iout $= 3$
As for **iout** $= 1$ and **iout** $= 2$.

iout $= 4$
As for **iout** $= 3$, plus the final residual (as returned in **ub**).

iout $= 5$
As for **iout** $= 4$, plus the initial elements of **a** and **rhs**.

iout $= 6$
As for **iout** $= 5$, plus the Galerkin coarse grid approximations.

iout $= 7$
As for **iout** $= 6$, plus the incomplete Crout decompositions.

iout $= 8$
As for **iout** $= 7$, plus the residual after each iteration.

The elements $\mathbf{a}(p, k)$, the Galerkin coarse grid approximations and the incomplete Crout decompositions are output in the format:

Y-index $= j$

X-index $= i\mathbf{a}(p,1)\mathbf{a}(p,2)\mathbf{a}(p,3)\mathbf{a}(p,4)\mathbf{a}(p,5)\mathbf{a}(p,6)\mathbf{a}(p,7)$

where $p = i + (j-1) \times$ **ngx**, for $i = 1, 2, \ldots,$ **ngx** and $j = 1, 2, \ldots,$ **ngy**.

The vectors $\mathbf{u}(p)$, $\mathbf{ub}(p)$, $\mathbf{rhs}(p)$ are output in matrix form with **ngy** rows and **ngx** columns. Where **ngx** > 10, the **ngx** values for a given j value are produced in rows of 10. Values of **iout** > 4 may yield considerable amounts of output.

Constraint: $0 \leq$ **iout** ≤ 8.

5.2 Optional Input Parameters

None.

5.3 Output Parameters

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

Is overwritten.

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

The first **ngx** $\times$ **ngy** elements are unchanged and the rest of the array is used as workspace.

3: **ub**(**ngx** $\times$ **ngy**) – REAL (KIND=nag_wp) array

The corresponding component of the residual $r = f - \mathbf{a}u$.

4: **us**(lda) – REAL (KIND=nag_wp) array

The residual 2-norm, stored in element **us**(1).

5: **u**(lda) – REAL (KIND=nag_wp) array

The computed solution u_{ij} is returned in **u**$(i + (j-1) \times$ **ngx**$)$, for $i = 1, 2, \ldots,$ **ngx** and $j = 1, 2, \ldots,$ **ngy**.

6: **numit** – INTEGER

The number of iterations performed.

7: **ifail** – INTEGER

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

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, **ngx** < 3,
or **ngy** < 3,
or lda is too small,
or **acc** < 0.0,
or **maxit** < 0,
or **iout** < 0,
or **iout** > 8.

ifail $= 2$

maxit iterations have been performed with the residual 2-norm decreasing at each iteration but the residual 2-norm has not been reduced to less than the specified tolerance (see **acc**). Examine the progress of the iteration by setting **iout** ≥ 2.

ifail $= 3$

As for **ifail** $= 2$, except that at one or more iterations the residual 2-norm did not decrease. It is likely that the method fails to converge for the given matrix A.

ifail $= 4$ (*warning*)

On entry, **acc** is less than the ***machine precision***. The function terminated because the residual norm is less than the ***machine precision***.

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

See **acc** (Section 5).

8 Further Comments

The rate of convergence of this function is strongly dependent upon the number of levels, l, in the multigrid scheme, and thus the choice of **ngx** and **ngy** is very important. You are advised to experiment with different values of **ngx** and **ngy** to see the effect they have on the rate of convergence; for example, using a value such as **ngx** $= 65$ $(= 2^6 + 1)$ followed by **ngx** $= 64$ (for which $l = 1$).

9 Example

The program solves the elliptic partial differential equation

$$U_{xx} - \alpha U_{xy} + U_{yy} = -4, \quad \alpha = 1.7$$

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

$$U = 0 \text{on} \begin{cases} x = 0, & (0 \leq y \leq 1) \\ y = 0, & (0 \leq x \leq 1) \\ y = 1, & (0 \leq x \leq 1) \end{cases} U = 1 \text{on } x = 1, \quad 0 \leq y \leq 1.$$

For the equation to be elliptic, α must be less than 2.

The equation is discretized on a square grid with mesh spacing h in both directions using the following approximations:

$$\begin{aligned} U_{xx} &\simeq \tfrac{1}{h^2}(U_{\mathrm{E}} - 2U_{\mathrm{O}} + U_{\mathrm{W}}) \\ U_{yy} &\simeq \tfrac{1}{h^2}(U_{\mathrm{N}} - 2U_{\mathrm{O}} + U_{\mathrm{S}}) \\ U_{xy} &\simeq \tfrac{1}{2h^2}(U_{\mathrm{N}} - U_{\mathrm{NW}} + U_{\mathrm{E}} - 2U_{\mathrm{O}} + U_{\mathrm{W}} - U_{\mathrm{SE}} + U_{\mathrm{S}}). \end{aligned}$$

Thus the following equations are solved:

$$\begin{array}{ccccccc} & \frac{1}{2}\alpha u_{i-1,j+1} & + & \left(1 - \frac{1}{2}\alpha\right) u_{i,j+1} & & & \\ + & \left(1 - \frac{1}{2}\alpha\right) u_{i+1,j} & + & (-4 + \alpha) u_{ij} & + & \left(1 - \frac{1}{2}\alpha\right) u_{i+1,j} & \\ & & + & \left(1 - \frac{1}{2}\alpha\right) u_{i,j-1} & + & \frac{1}{2}\alpha u_{i+1,j-1} = -4h^2 \end{array}$$

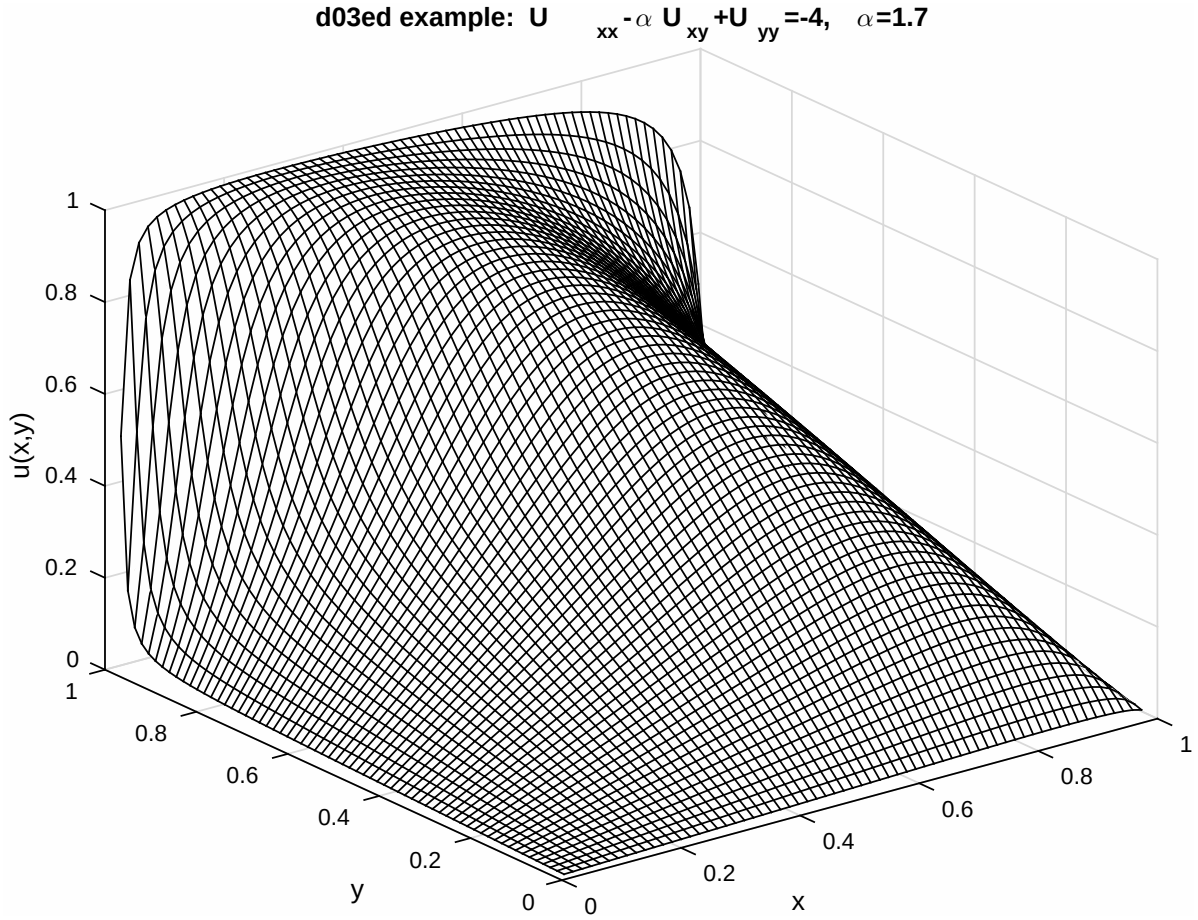
9.1 Program Text

```
function d03ed_example

fprintf('d03ed example results\n\n');

ngx = nag_int(65);
ngy = ngx;
nn  = ngx*ngy;
lda = 4*(ngx+1)*(ngy+1)/3;
a   = zeros(lda,7);
rhs = zeros(lda, 1);
ub  = zeros(nn, 1);

maxit = nag_int(15); acc = 0.0001; iout = nag_int(0);
hx = 1/double(ngx+1);
hy = 1/double(ngy+1);
alpha = 1.7;

% Set up operator, right-hand side and initial guess for
% step-lengths hx and hy
a(1:nn,[1 3 5 7]) = 1 - 0.5*alpha;
a(1:nn,[2 6])     = 0.5*alpha;
```

```
a(1:nn,4)         = -4 + alpha;
rhs(1:nn)         = -4*hx*hy;

% Boundary conditions u(x,y=0,1) = 0
a(2:ngx-1,1:2)       = 0.0;
a(nn-ngx+2:nn-1,6:7) = 0.0;
for j = 2:ngy-1
  % Boundary condition u(x=0,y) = 0
  iy = (j-1)*ngx + 1;
  a(iy,[3 6]) = 0.0;
  % Boundary condition u(x=1,y) = 1
  iy = j*ngx;
  rhs(iy) = rhs(iy) - a(iy,5) - a(iy,2);
  a(iy,[2 5]) = 0.0;
end

% Bottom left corner
a(1,[1:3 6]) = 0.0;
% Top left corner
k = nn - ngx + 1;
a(k,[3 6:7]) = 0.0;
% Bottom right corner
% Use average value at discontinuity ( = 0.5 )
k = ngx;
rhs(k) = rhs(k) - a(k,2)*0.5 - a(k,5);
a(k,[1:2 5]) = 0.0;
% Top right corner
k = ngx*ngy;
rhs(k) = rhs(k) - a(k,2) - a(k,5);
a(k,[2 5:7]) = 0.0;
u = zeros(ngx*ngy,1);

[a, rhs, ub, resid, u, numit, ifail] = ...
d03ed(...
       ngx, ngy, a, rhs, ub, maxit, acc, iout);

 % Output solution statistics
fprintf('Number of iterations  %4d\n',numit);
fprintf('Maximum residual       %8.3e\n',resid(numit));

% Plot the solution u
fig1 = figure;
u2 = u(1:nn);
umat = reshape(u2,ngx,ngy);
x(1:ngx) = hx:hx:1-hx; y(1:ngy) = hy:hy:1-hy;
[xs,ys] = meshgrid(y,x);
mesh(xs,ys,umat);
xlabel('x'); ylabel('y'); zlabel ('u(x,y)');
title('d03ed example:  U_{xx}-\alpha U_{xy}+U_{yy}=-4, \alpha=1.7');
```

9.2 Program Results

```
     d03ed example results

Number of iterations     4
Maximum residual       -1.060e-08
```


d03ed example: U xx - α U xy +U yy =-4, α=1.7
u(x,y)
1
0.8
0.6
0.4
0.2
0
y
x