NAG Toolbox for MATLAB

d03rz

1 Purpose

d03rz is designed to be used in conjunction with d03rb. It can be called from the user-supplied (sub)program **monitr** to obtain the number of grid points and their (x, y) co-ordinates on a solution grid.

2 Syntax

```
[npts, x, y, ifail] = d03rz(level, nlev, xmin, ymin, dxb, dyb, lgrid, istruc, lenxy)
```

3 Description

d03rz extracts the number of grid points and their (x, y) co-ordinates on a specific solution grid produced by d03rb. It must be called only from within the user-supplied (sub)program **monitr**. The parameters **nlev**, **xmin**, **ymin**, **dxb**, **dyb**, **lgrid** and **istruc** to **monitr** must be passed unchanged to d03rz.

4 References

None.

5 Parameters

5.1 Compulsory Input Parameters

1: **level – int32 scalar**

The grid level at which the co-ordinates are required.

Constraint: $1 \leq$ **level** $\leq$ **nlev**.

2: **nlev – int32 scalar**

3: **xmin – double scalar**

4: **ymin – double scalar**

5: **dxb – double scalar**

6: **dyb – double scalar**

7: **lgrid**$(*)$ **– int32 array**

8: **istruc**$(*)$ **– int32 array**

Note: the dimension of the array **lgrid** must be at least **nlev**.

The dimension of the array **istruc** must be at least **lgrid**(**nlev**) $+ 2 \times nrows +$ **npts** $+ 1$ where $nrows$ is stored in **istruc**(**lgrid**(**nlev**)) and is the number of rows in the grid at level **nlev**.

nlev, **xmin**, **ymin**, **dxb**, **dyb**, **lgrid** and **istruc** as supplied to user-supplied (sub)program **monitr** must be passed unchanged to d03rz.

9: **lenxy – int32 scalar**

Constraint: **lenxy** $\geq$ **npts**.

5.2 Optional Input Parameters

None.

5.3 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **npts – int32 scalar**

The number of grid points in the grid level **level**.

2: **x**(**lenxy**) **– double array**

3: **y**(**lenxy**) **– double array**

$\mathbf{x}(i)$ and $\mathbf{y}(i)$ contain the (x,y) co-ordinates respectively of the ith grid point, for $i = 1, 2, \ldots,$ **npts**.

4: **ifail – int32 scalar**

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

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, **level** < 1,
or **level** $>$ **nlev**.

ifail $= 2$

The dimension of the arrays **x** and **y** is too small for the requested grid level, i.e., **lenxy** $<$ **npts**.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

d03rb_bndary.m

```
function [res] = ...
     bndary(npts, npde, t, x, y, u, ut, ux, uy, nbnds, nbpts, llbnd,
ilbnd, lbnd, res)

      eps = 1e-3;

      for k = llbnd(1):nbpts
         i = lbnd(k);
         a = (-4.0d0*x(i)+4.0d0*y(i)-t)/(32.0d0*eps);
         if (a <= 0.0d0)
            res(i,1) = u(i,1) - (0.75d0-0.25d0/(1.0d0+exp(a)));
            res(i,2) = u(i,2) - (0.75d0+0.25d0/(1.0d0+exp(a)));
         else
                     res(i,1)  =  u(i,1)  -  (0.75d0-0.25d0*exp(-a)/(exp(-
a)+1.0d0));
                     res(i,2)  =  u(i,2)  -  (0.75d0+0.25d0*exp(-a)/(exp(-
a)+1.0d0));
         end
```

```
      end
```

d03rb_inidom.m

```
function [xmin, xmax, ymin, ymax, nx, ny, npts, nrows, nbnds, nbpts,
...
          lrow, irow, icol, llbnd, ilbnd, lbnd, ierr] = inidom(maxpts,
ierr)

  nrows = int32(11);
  npts = int32(105);
  nbnds = int32(28);
  nbpts = int32(72);

  lrow = zeros(nrows, 1, 'int32');
  irow = zeros(nrows, 1, 'int32');
  icol = zeros(npts, 1, 'int32');
  llbnd = zeros(nbnds, 1, 'int32');
  ilbnd = zeros(nbnds, 1, 'int32');
  lbnd = zeros(nbpts, 1, 'int32');



  icold = [int32(0),
           int32(1),
           int32(2),
           int32(0),
           int32(1),
           int32(2),
           int32(3),
           int32(4),
           int32(5),
           int32(6),
           int32(7),
           int32(8),
           int32(9),
           int32(10),
           int32(0),
           int32(1),
           int32(2),
           int32(3),
           int32(4),
           int32(5),
           int32(6),
           int32(7),
           int32(8),
           int32(9),
           int32(10),
           int32(0),
           int32(1),
           int32(2),
           int32(3),
           int32(4),
           int32(5),
           int32(6),
           int32(7),
           int32(8),
           int32(9),
           int32(10),
           int32(0),
           int32(1),
           int32(2),
           int32(3),
           int32(4),
           int32(5),
           int32(8),
           int32(9),
           int32(10),
```

```
           int32(0),
           int32(1),
           int32(2),
           int32(3),
           int32(4),
           int32(5),
           int32(6),
           int32(7),
           int32(8),
           int32(9),
           int32(10),
           int32(0),
           int32(1),
           int32(2),
           int32(3),
           int32(4),
           int32(5),
           int32(6),
           int32(7),
           int32(8),
           int32(9),
           int32(10),
           int32(0),
           int32(1),
           int32(2),
           int32(3),
           int32(4),
           int32(5),
           int32(6),
           int32(7),
           int32(8),
           int32(9),
           int32(10),
           int32(0),
           int32(1),
           int32(2),
           int32(3),
           int32(4),
           int32(5),
           int32(6),
           int32(7),
           int32(8),
           int32(0),
           int32(1),
           int32(2),
           int32(3),
           int32(4),
           int32(5),
           int32(6),
           int32(7),
           int32(8),
           int32(0),
           int32(1),
           int32(2),
           int32(3),
           int32(4),
           int32(5),
           int32(6),
           int32(7),
           int32(8)];

  ilbndd = [int32(1),
            int32(2),
            int32(3),
            int32(4),
            int32(1),
            int32(4),
            int32(1),
            int32(2),
            int32(3),
```

```
            int32(4),
            int32(3),
            int32(4),
            int32(1),
            int32(2),
            int32(12),
            int32(23),
            int32(34),
            int32(41),
            int32(14),
            int32(41),
            int32(12),
            int32(23),
            int32(34),
            int32(41),
            int32(43),
            int32(14),
            int32(21),
            int32(32)];

  irowd = [int32(0),
           int32(1),
           int32(2),
           int32(3),
           int32(4),
           int32(5),
           int32(6),
           int32(7),
           int32(8),
           int32(9),
           int32(10)];

  lbndd = [int32(2),
           int32(4),
           int32(15),
           int32(26),
           int32(37),
           int32(46),
           int32(57),
           int32(68),
           int32(79),
           int32(88),
           int32(98),
           int32(99),
           int32(100),
           int32(101),
           int32(102),
           int32(103),
           int32(104),
           int32(96),
           int32(86),
           int32(85),
           int32(84),
           int32(83),
           int32(82),
           int32(70),
           int32(59),
           int32(48),
           int32(39),
           int32(28),
           int32(17),
           int32(6),
           int32(8),
           int32(9),
           int32(10),
           int32(11),
           int32(12),
           int32(13),
           int32(18),
           int32(29),
```

```
           int32(40),
           int32(49),
           int32(60),
           int32(72),
           int32(73),
           int32(74),
           int32(75),
           int32(76),
           int32(77),
           int32(67),
           int32(56),
           int32(45),
           int32(36),
           int32(25),
           int32(33),
           int32(32),
           int32(42),
           int32(52),
           int32(53),
           int32(43),
           int32(1),
           int32(97),
           int32(105),
           int32(87),
           int32(81),
           int32(3),
           int32(7),
           int32(71),
           int32(78),
           int32(14),
           int32(31),
           int32(51),
           int32(54),
           int32(34)];

  llbndd = [int32(1),
            int32(2),
            int32(11),
            int32(18),
            int32(19),
            int32(24),
            int32(31),
            int32(37),
            int32(42),
            int32(48),
            int32(53),
            int32(55),
            int32(56),
            int32(58),
            int32(59),
            int32(60),
            int32(61),
            int32(62),
            int32(63),
            int32(64),
            int32(65),
            int32(66),
            int32(67),
            int32(68),
            int32(69),
            int32(70),
            int32(71),
            int32(72)];

  lrowd = [int32(1),
           int32(4),
           int32(15),
           int32(26),
           int32(37),
           int32(46),
```

```
           int32(57),
           int32(68),
           int32(79),
           int32(88),
           int32(97)];

  nx = int32(11);
  ny = int32(11);

% check maxpts against rough estimate of npts
  if (maxpts < nx*ny)
     ierr = -1;
     return;
  end

  xmin = 0.0d0;
  ymin = 0.0d0;
  xmax = 1.0d0;
  ymax = 1.0d0;

  for i = 1:nrows
    lrow(i) = lrowd(i);
    irow(i) = irowd(i);
  end

  for i = 1:nbnds
    llbnd(i) = llbndd(i);
    ilbnd(i) = ilbndd(i);
  end

  for i = 1:nbpts
    lbnd(i) = lbndd(i);
  end

  for i = 1:npts
    icol(i) = icold(i);
  end
```

d03rb_pdedef.m

```
function [res] = pdedef(npts, npde, t, x, y, u, ut, ux, uy, uxx, uxy,
uyy)
  res = zeros(npts, npde);

  eps = 1e-3;
  for i = 1:npts
                          res(i,1)      =      ut(i,1)-(-u(i,1)*ux(i,1)-
u(i,2)*uy(i,1)+eps*(uxx(i,1)+uyy(i,1)));
                          res(i,2)      =      ut(i,2)-(-u(i,1)*ux(i,2)-
u(i,2)*uy(i,2)+eps*(uxx(i,2)+uyy(i,2)));
  end
```

d03rb_pdeiv.m

```
function [u] = pdeiv(npts, npde, t, x, y)
  u = zeros(npts, npde);

  eps = 1e-3;

  for i = 1:npts
    a = (-4.0d0*x(i)+4.0d0*y(i)-t)/(32.0d0*eps);
    if (a <= 0.0d0)
       u(i,1) = 0.75d0 - 0.25d0/(1.0d0+exp(a));
       u(i,2) = 0.75d0 + 0.25d0/(1.0d0+exp(a));
    else
       u(i,1) = 0.75d0 - 0.25d0*exp(-a)/(exp(-a)+1.0d0);
       u(i,2) = 0.75d0 + 0.25d0*exp(-a)/(exp(-a)+1.0d0);
```

```
    end
  end
```

d03rz_monitr.m

```
function [ierr] = ...
      monitr(npde, t, dt, dtnew, tlast, nlev, xmin, ymin, dxb, dyb,
lgrid, ...
           istruc, lsol, sol, ierr)

  lenxy=int32(2500);

  global iout;

  if (tlast && iout == 2)
    for level = 1:nlev
      ipsol = lsol(level);

      % Get grid information
      [npts, x, y, ifail] = ...
    d03rz(level, nlev, xmin, ymin, dxb, dyb, lgrid, istruc, lenxy);
      if (ifail ~= 0)
        ierr = 1;
      elseif (level == 1)
        % Get exact solution
        [uex] = d03rb_pdeiv(npts, npde, t, x, y);

        fprintf('\n Solution at every 2nd grid point in level 1 at time
%8.4f\n', t);
         fprintf('        x            y         approx u      exact u
approx v     exact v\n');
        for ipt = 1:2:npts
              fprintf('%11.4E  %11.4e  %11.4e  %11.4e  %11.4e  %11.4e\n',
x(ipt), ...
              y(ipt), sol(ipsol+ipt), uex(ipt,1), sol(ipsol+npts+ipt),
uex(ipt,2));
        end
        fprintf('\n');
      end
    end
  end
```

```
ts = 0;
twant = [0.25; 1];
dt = [0.001;
     1e-07;
     0];
tols = 0.1;
tolt = 0.05;
opti = [int32(5);
     int32(0);
     int32(0);
     int32(0)];
optr = [1, 1;
     1, 1;
     1, 1];
rwk = zeros(426000, 1);
iwk = zeros(117037, 1, 'int32');
lenlwk = int32(6000);
itrace = int32(0);
ind = int32(0);
global iout;
for iout = 1:2
  tout = twant(iout);
  [ts, dt, rwk, iwk, ind, ifail] = ...
    d03rb(ts, tout, dt, tols, tolt, 'd03rb_inidom', 'd03rb_pdedef', ...
```

```
    'd03rb_bndary', 'd03rb_pdeiv', 'd03rz_monitr', opti, optr, ...
    rwk, iwk, lenlwk, itrace, ind);
  fprintf('\nStatistics\n');
  fprintf('Time = %8.4f\n', ts);
  fprintf('Total number of accepted timesteps = %d\n', iwk(1));
  fprintf('Total number of rejected timesteps = %d\n', iwk(2));
  fprintf('\n            T o t a l  n u m b e r  o f    \n');
  fprintf('          Residual  Jacobian    Newton  Lin sys\n');
  fprintf('            evals     evals     iters    iters\n');
  fprintf(' At level \n');
  maxlev = opti(1);
  for j = 1:maxlev
    if (iwk(j+2) ~= 0)
       fprintf('%6d %10d %10d %10d %10d\n', j, iwk(j+2), iwk(j+2+maxlev),
...
                                iwk(j+2+2*maxlev), iwk(j+2+3*maxlev) );
    end
  end
  fprintf('\n            M a x i m u m  n u m b e r  o f\n');
  fprintf('             Newton iters    Lin sys iters \n');
  fprintf(' At level \n');
  for j = 1:maxlev
    if (iwk(j+2) ~= 0)
      fprintf('%6d%14d%14d\n', j, iwk(j+2+4*maxlev), iwk(j+2+5*maxlev) );
    end
  end

end
```

```
Statistics
Time =   0.2500
Total number of accepted timesteps = 14
Total number of rejected timesteps = 0

            T o t a l  n u m b e r  o f
          Residual  Jacobian    Newton  Lin sys
            evals     evals     iters    iters
 At level
     1        196         14         28         14
     2        196         14         28         22
     3        196         14         28         25
     4        196         14         28         31
     5        141         10         21         29

            M a x i m u m  n u m b e r  o f
             Newton iters    Lin sys iters
 At level
     1             2             1
     2             2             1
     3             2             1
     4             2             2
     5             3             2

 Solution at every 2nd grid point in level 1 at time   1.0000
       x          y        approx u     exact u    approx v     exact v
 0.0000E+00  0.0000e+00  5.0000e-01  5.0000e-01  1.0000e-00  1.0000e-00
 2.0000E-01  0.0000e+00  5.0000e-01  5.0000e-01  1.0000e+00  1.0000e+00
 1.0000E-01  1.0000e-01  5.0022e-01  5.0000e-01  9.9978e-01  1.0000e-00
 3.0000E-01  1.0000e-01  5.0000e-01  5.0000e-01  1.0000e+00  1.0000e+00
 5.0000E-01  1.0000e-01  5.0000e-01  5.0000e-01  1.0000e+00  1.0000e+00
 7.0000E-01  1.0000e-01  5.0000e-01  5.0000e-01  1.0000e+00  1.0000e+00
 9.0000E-01  1.0000e-01  5.0000e-01  5.0000e-01  1.0000e+00  1.0000e+00
 0.0000E+00  2.0000e-01  5.0048e-01  5.0048e-01  9.9952e-01  9.9952e-01
 2.0000E-01  2.0000e-01  5.0000e-01  5.0000e-01  1.0000e-00  1.0000e-00
 4.0000E-01  2.0000e-01  5.0011e-01  5.0000e-01  9.9989e-01  1.0000e+00
 6.0000E-01  2.0000e-01  4.9994e-01  5.0000e-01  1.0001e+00  1.0000e+00
 8.0000E-01  2.0000e-01  4.9999e-01  5.0000e-01  1.0000e+00  1.0000e+00
 1.0000E+00  2.0000e-01  5.0000e-01  5.0000e-01  1.0000e+00  1.0000e+00
 1.0000E-01  3.0000e-01  4.9997e-01  5.0048e-01  1.0000e+00  9.9952e-01
 3.0000E-01  3.0000e-01  5.0000e-01  5.0000e-01  1.0000e-00  1.0000e-00
```

```
 5.0000E-01  3.0000e-01  5.0000e-01  5.0000e-01  1.0000e+00  1.0000e+00
 7.0000E-01  3.0000e-01  5.0000e-01  5.0000e-01  1.0000e+00  1.0000e+00
 9.0000E-01  3.0000e-01  5.0003e-01  5.0000e-01  9.9997e-01  1.0000e+00
 0.0000E+00  4.0000e-01  7.5000e-01  7.5000e-01  7.5000e-01  7.5000e-01
 2.0000E-01  4.0000e-01  5.0048e-01  5.0048e-01  9.9952e-01  9.9952e-01
 4.0000E-01  4.0000e-01  5.0018e-01  5.0000e-01  9.9982e-01  1.0000e-00
 8.0000E-01  4.0000e-01  5.0000e-01  5.0000e-01  1.0000e+00  1.0000e+00
 1.0000E+00  4.0000e-01  5.0000e-01  5.0000e-01  1.0000e+00  1.0000e+00
 1.0000E-01  5.0000e-01  7.4998e-01  7.5000e-01  7.5002e-01  7.5000e-01
 3.0000E-01  5.0000e-01  5.0048e-01  5.0048e-01  9.9952e-01  9.9952e-01
 5.0000E-01  5.0000e-01  5.0000e-01  5.0000e-01  1.0000e-00  1.0000e-00
 7.0000E-01  5.0000e-01  5.0000e-01  5.0000e-01  1.0000e+00  1.0000e+00
 9.0000E-01  5.0000e-01  5.0005e-01  5.0000e-01  9.9995e-01  1.0000e+00
 0.0000E+00  6.0000e-01  7.5000e-01  7.5000e-01  7.5000e-01  7.5000e-01
 2.0000E-01  6.0000e-01  7.5000e-01  7.5000e-01  7.5000e-01  7.5000e-01
 4.0000E-01  6.0000e-01  4.9998e-01  5.0048e-01  1.0000e+00  9.9952e-01
 6.0000E-01  6.0000e-01  4.9991e-01  5.0000e-01  1.0001e+00  1.0000e-00
 8.0000E-01  6.0000e-01  4.9983e-01  5.0000e-01  1.0002e+00  1.0000e+00
 1.0000E+00  6.0000e-01  5.0000e-01  5.0000e-01  1.0000e+00  1.0000e+00
 1.0000E-01  7.0000e-01  7.5002e-01  7.5000e-01  7.4998e-01  7.5000e-01
 3.0000E-01  7.0000e-01  7.5000e-01  7.5000e-01  7.5000e-01  7.5000e-01
 5.0000E-01  7.0000e-01  5.0048e-01  5.0048e-01  9.9952e-01  9.9952e-01
 7.0000E-01  7.0000e-01  5.0000e-01  5.0000e-01  1.0000e-00  1.0000e-00
 9.0000E-01  7.0000e-01  5.0000e-01  5.0000e-01  1.0000e+00  1.0000e+00
 0.0000E+00  8.0000e-01  7.5000e-01  7.5000e-01  7.5000e-01  7.5000e-01
 2.0000E-01  8.0000e-01  7.5000e-01  7.5000e-01  7.5000e-01  7.5000e-01
 4.0000E-01  8.0000e-01  7.5000e-01  7.5000e-01  7.5000e-01  7.5000e-01
 6.0000E-01  8.0000e-01  5.0048e-01  5.0048e-01  9.9952e-01  9.9952e-01
 8.0000E-01  8.0000e-01  5.0000e-01  5.0000e-01  1.0000e-00  1.0000e-00
 1.0000E-01  9.0000e-01  7.5001e-01  7.5000e-01  7.4999e-01  7.5000e-01
 3.0000E-01  9.0000e-01  7.5002e-01  7.5000e-01  7.4998e-01  7.5000e-01
 5.0000E-01  9.0000e-01  7.5002e-01  7.5000e-01  7.4998e-01  7.5000e-01
 7.0000E-01  9.0000e-01  4.9994e-01  5.0048e-01  1.0001e+00  9.9952e-01
 0.0000E+00  1.0000e+00  7.5000e-01  7.5000e-01  7.5000e-01  7.5000e-01
 2.0000E-01  1.0000e+00  7.5000e-01  7.5000e-01  7.5000e-01  7.5000e-01
 4.0000E-01  1.0000e+00  7.5000e-01  7.5000e-01  7.5000e-01  7.5000e-01
 6.0000E-01  1.0000e+00  7.5000e-01  7.5000e-01  7.5000e-01  7.5000e-01
 8.0000E-01  1.0000e+00  5.0048e-01  5.0048e-01  9.9952e-01  9.9952e-01


Statistics
Time =   1.0000
Total number of accepted timesteps = 45
Total number of rejected timesteps = 0

            T o t a l  n u m b e r  o f
          Residual  Jacobian    Newton  Lin sys
             evals     evals     iters    iters
 At level
     1        630        45        90        45
     2        630        45        90        78
     3        630        45        90        87
     4        630        45        90       124
     5        575        41        83       122

            M a x i m u m  n u m b e r  o f
             Newton iters    Lin sys iters
 At level
     1            2            1
     2            2            1
     3            2            1
     4            2            2
     5            3            2
```