NAG Toolbox for MATLAB

d03ea

1 Purpose

d03ea solves Laplace's equation in two dimensions for an arbitrary domain bounded internally or externally by one or more closed contours, given the value of either the unknown function or its normal derivative (into the domain) at each point of the boundary.

2 Syntax

```
[phi, phid, alpha, ifail] = d03ea(stage1, ext, dorm, p, q, x, y, phi,
phid, alpha, 'n', n, 'n1p1', n1p1)
```

3 Description

The function uses an integral equation method, based upon Green's formula, which yields the solution, ϕ, within the domain, given its value or that of its normal derivative at each point of the boundary (except possibly at a finite number of discrete points). The solution is obtained in two stages. The first, which is executed once only, determines the complementary boundary-values, i.e., ϕ, where its normal derivative is known and vice versa. The second stage is entered once for each point at which the solution is required.

The boundary is divided into a number of intervals in each of which ϕ and its normal derivative are approximated by constants. Of these half are evaluated by applying the given boundary conditions at one 'nodal' point within each interval while the remainder are determined (in stage 1) by solving a set of simultaneous linear equations. Here this is achieved by means of auxiliary functions f02wd and f04jg, which will yield the least-squares solution of an overdetermined system of equations as well as the unique solution of a square nonsingular system.

In exterior domains the solution behaves as $c + s \log r + O(1/r)$ as r tends to infinity, where c is a constant, s is the total integral of the normal derivative around the boundary and r is the radial distance from the origin of co-ordinates. For the Neumann problem (when the normal derivative is given along the whole boundary) s is fixed by the boundary conditions whilst c is chosen by you. However, for a Dirichlet problem (when ϕ is given along the whole boundary) or for a mixed problem, stage 1 produces a value of c for which $s = 0$; then as r tends to infinity the solution tends to the constant c.

4 References

Symm G T and Pitfield R A 1974 Solution of Laplace's equation in two dimensions *NPL Report NAC 44* National Physical Laboratory

5 Parameters

5.1 Compulsory Input Parameters

1: **stage1 – logical scalar**

Indicates whether the function call is for stage 1 of the computation as defined in Section 3.

stage1 = **true**

The call is for stage 1.

stage1 = **false**

The call is for stage 2.

2: **ext – logical scalar**

The form of the domain. If **ext** = **true**, the domain is unbounded. Otherwise the domain is an interior one.

3: **dorm – logical scalar**

The form of the boundary conditions. If **dorm** = **true**, then the problem is a Dirichlet or mixed boundary-value problem. Otherwise it is a Neumann problem.

4: **p – double scalar**
5: **q – double scalar**

To stage 2, **p** and **q** must specify the x and y co-ordinates respectively of a point at which the solution is required.

When **stage1** is **true**, **p** and **q** are ignored.

6: **x**(**n1p1**) **– double array**
7: **y**(**n1p1**) **– double array**

The x and y co-ordinates respectively of points on the one or more closed contours which define the domain of the problem.

Note: each contour is described in such a manner that the subscripts of the co-ordinates increase when the domain is kept on the left. The final point on each contour coincides with the first and, if a further contour is to be described, the co-ordinates of this point are repeated in the arrays. In this way each interval is defined by three points, the second of which (the nodal point) always has an even subscript. In the case of the interior Neumann problem, the outermost boundary contour must be given first, if there is more than one

8: **phi**(**n**) **– double array**

For stage 1, **phi** must contain the nodal values of ϕ or its normal derivative (into the domain) as prescribed in each interval. For stage 2 it must retain its output values from stage 1.

9: **phid**(**n**) **– double array**

For stage 1, **phid**(i) must hold the value 0.0 or 1.0 according as **phi**(i) contains a value of ϕ or its normal derivative, for $i = 1, 2, \ldots,$ **n**. For stage 2 it must retain its output values from stage 1.

10: **alpha – double scalar**

For stage 1, the use of **alpha** depends on the nature of the problem:

if **dorm** = **true**, **alpha** need not be set;
if **dorm** = **false** and **ext** = **true**, **alpha** must contain the prescribed constant c (see Section 3);
if **dorm** = **false** and **ext** = **false**, **alpha** must contain an appropriate value (often zero) for the integral of ϕ around the outermost boundary.

For stage 2, on every call **alpha** must contain the value returned at stage 1.

5.2 Optional Input Parameters

1: **n – int32 scalar**

Default: The dimension of the arrays **phi**, **phid**. (An error is raised if these dimensions are not equal.)

the number of intervals into which the boundary is divided (see Sections 7 and 8).

2: **n1p1 – int32 scalar**

Default: The dimension of the arrays **x**, **y**. (An error is raised if these dimensions are not equal.)

the value $2(\mathbf{n} + M) - 1$, where M denotes the number of closed contours which make up the boundary.

5.3 Input Parameters Omitted from the MATLAB Interface

c, ldc, np4, icint, np1

5.4 Output Parameters

1: **phi(n) – double array**

From stage 1, it contains the constants which approximate ϕ in each interval. It remains unchanged on exit from stage 2.

2: **phid(n) – double array**

From stage 1, **phid** contains the constants which approximate the normal derivative of ϕ in each interval. It remains unchanged on exit from stage 2.

3: **alpha – double scalar**

From stage 1:

if **ext** = **false**, **alpha** contains 0.0;
if **ext** = **true** and **dorm** = **false** **alpha** is unchanged;
if **ext** = **true** and **dorm** = **true** **alpha** contains a computed estimate for *c*.

From stage 2:

alpha contains the computed value of ϕ at the point (**p**,**q**).

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

Invalid tolerance used in an internal call to an auxiliary function:

icint(1) = 0

Indicates too large a tolerance.

icint(1) > 0

Indicates too small a tolerance.

Note: this error is only possible in stage 1, and the circumstances under which it may occur cannot be foreseen. In the event of a failure, it is suggested that you change the scale of the domain of the problem and apply the function again.

ifail $= 2$

Incorrect rank obtained by an auxiliary function; **icint**(1) contains the computed rank.

7 Accuracy

The accuracy of the computed solution depends upon how closely ϕ and its normal derivative may be approximated by constants in each interval of the boundary and upon how well the boundary contours are represented by polygons with vertices at the selected points $(\mathbf{x}(i), \mathbf{y}(i))$, $i = 1, 2, \ldots, 2(\mathbf{n} + \mathrm{M}) - 1$.

Consequently, in general, the accuracy increases as the boundary is subdivided into smaller and smaller intervals and by comparing solutions for successive subdivisions one may obtain an indication of the error in these solutions.

Alternatively, since the point of maximum error always lies on the boundary of the domain, an estimate of the error may be obtained by computing ϕ at a sufficient number of points on the boundary where the true solution is known. The latter method (not applicable to the Neumann problem) is most useful in the case where ϕ alone is prescribed on the boundary (the Dirichlet problem).

8 Further Comments

The time taken for stage 1, which is executed once only, is roughly proportional to $\mathbf{n}^2$, being dominated by the time taken to compute the coefficients. The time for each stage 2 application is proportional to $\mathbf{n}$.

The intervals into which the boundary is divided need not be of equal lengths.

For many practical problems useful results may be obtained with 20 to 40 intervals per boundary contour.

9 Example

```
stage1 = true;
ext = false;
dorm = false;
p = 0;
q = 0;
x = [3;
     1.5;
     0;
     -1.5;
     -3;
     0;
     3;
     3;
     2;
     0;
     -2;
     -1;
     0;
     1;
     2];
y = [0;
     2;
     4;
     2;
     0;
     0;
     0;
     0;
     1;
     1;
     1;
     2;
     3;
     2;
     1];
phi = [0;
     0;
     0;
     0;
     0;
     0];
phid = [1;
     1;
     1;
     1;
     1;
     1];
alpha = 16;
[phiOut, phidOut, alphaOut, ifail] = ...
    d03ea(stage1, ext, dorm, p, q, x, y, phi, phid, alpha)
```

```
phiOut =
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
phidOut =
     0
     0
     0
     0
     0
     0
alphaOut =
     0
ifail =
           0
```