NAG Toolbox Chapter Introduction

D03 – Partial Differential Equations

Contents

1 Scope of the Chapter ..................................................... 2
2 Background to the Problems .............................................. 2
3 Recommendations on Choice and Use of Available Functions .............. 3
4 Decision Trees ............................................................ 8
5 Functionality Index .......................................................... 10
6 References ...................................................................... 11
1 Scope of the Chapter

This chapter is concerned with the numerical solution of partial differential equations.

2 Background to the Problems

The definition of a partial differential equation problem includes not only the equation itself but also the domain of interest and appropriate subsidiary conditions. Indeed, partial differential equations are usually classified as elliptic, hyperbolic or parabolic according to the form of the equation and the form of the subsidiary conditions which must be assigned to produce a well-posed problem. The functions in this chapter will often call upon functions from other chapters, such as Chapter F04 (Simultaneous Linear Equations) and Chapter D02 (Ordinary Differential Equations). Other chapters also contain relevant functions, in particular Chapter D06 (Mesh Generation) and Chapter F11 (Large Scale Linear Systems).

The classification of partial differential equations is easily described in the case of linear equations of the second order in two independent variables, i.e., equations of the form

$$au_{xx} + 2bu_{xy} + cu_{yy} + du_x + eu_y + fu + g = 0,$$

where $a$, $b$, $c$, $d$, $e$, $f$ and $g$ are functions of $x$ and $y$ only. Equation (1) is called elliptic, hyperbolic or parabolic according to whether $ac - b^2$ is positive, negative or zero, respectively. Useful definitions of the concepts of elliptic, hyperbolic and parabolic character can also be given for differential equations in more than two independent variables, for systems and for nonlinear differential equations.

For elliptic equations, of which Laplace's equation

$$u_{xx} + u_{yy} = 0$$

is the simplest example of second order, the subsidiary conditions take the form of boundary conditions, i.e., conditions which provide information about the solution at all points of a closed boundary. For example, if equation (2) holds in a plane domain $D$ bounded by a contour $C$, a solution $u$ may be sought subject to the condition

$$u = f \quad \text{on} \quad C,$$

where $f$ is a given function. The condition (3) is known as a Dirichlet boundary condition. Equally common is the Neumann boundary condition

$$u' = g \quad \text{on} \quad C,$$

which is one form of a more general condition

$$u' + fu = g \quad \text{on} \quad C,$$

where $u'$ denotes the derivative of $u$ normal to the contour $C$, and $f$ and $g$ are given functions. Provided that $f$ and $g$ satisfy certain restrictions, condition (5) yields a well-posed boundary value problem for Laplace's equation. In the case of the Neumann problem, one further piece of information, e.g., the value of $u$ at a particular point, is necessary for uniqueness of the solution. Boundary conditions similar to the above are applicable to more general second-order elliptic equations, whilst two such conditions are required for equations of fourth order.

For hyperbolic equations, the wave equation

$$u_{tt} - u_{xx} = 0$$

is the simplest example of second order. It is equivalent to a first-order system

$$u_t - u_x = 0, \quad u_t - u_x = 0.$$ 

The subsidiary conditions may take the form of initial conditions, i.e., conditions which provide information about the solution at points on a suitable open boundary. For example, if equation (6) is satisfied for $t > 0$, a solution $u$ may be sought such that

$$u(x,0) = f(x), \quad u_t(x,0) = g(x),$$
where $f$ and $g$ are given functions. This is an example of an initial value problem, sometimes known as Cauchy's problem.

For parabolic equations, of which the heat conduction equation

$$u_t - u_{xx} = 0$$

(9)

is the simplest example, the subsidiary conditions always include some of initial type and may also include some of boundary type. For example, if equation (9) is satisfied for $t > 0$ and $0 < x < 1$, a solution $u$ may be sought such that

$$u(x, 0) = f(x), \quad 0 < x < 1,$$

(10)

and

$$u(0, t) = 0, \quad u(1, t) = 1, \quad t > 0.$$  

(11)

This is an example of a mixed initial/boundary value problem.

For all types of partial differential equations, finite difference methods (see Mitchell and Griffiths (1980)) and finite element methods (see Wait and Mitchell (1985)) are the most common means of solution. The use of finite difference methods features prominently in this chapter, however, the use of finite element methods is sufficiently large in scope to warrant a separate library of functions for their implementation. NAG no longer provides a companion finite element library, but there are several such specialist libraries available, including ParaFEM. Some of the utility functions in this chapter are concerned with the solution of the large sparse systems of equations which arise from finite difference and finite element methods. Further functions for this purpose are provided in Chapter F11.

Alternative methods of solution are often suitable for special classes of problems. For example, the method of characteristics is the most common for hyperbolic equations involving time and one space dimension (see Smith (1985)). The method of lines (see Mikhlin and Smolitsky (1967)) may be used to reduce a parabolic equation to a (stiff) system of ordinary differential equations, which may be solved by means of functions from Chapter D02 (Ordinary Differential Equations). Similarly, integral equation or boundary element methods (see Jaswon and Symm (1977)) are frequently used for elliptic equations.

Typically, in the latter case, the solution of a boundary value problem is represented in terms of certain boundary functions by an integral expression which satisfies the differential equation throughout the relevant domain. The boundary functions are obtained by applying the given boundary conditions to this representation. Implementation of this method necessitates discretization of only the boundary of the domain, the dimensionality of the problem thus being effectively reduced by one. The boundary conditions yield a full system of simultaneous equations, as opposed to the sparse systems yielded by finite difference and finite element methods, but the full system is usually of much lower order. Solution of this system yields the boundary functions, from which the solution of the problem may be obtained, by quadrature, as and where required.

3 Recommendations on Choice and Use of Available Functions

The choice of function will depend first of all upon the type of partial differential equation to be solved. At present no special allowances are made for problems with boundary singularities such as may arise at corners of domains or at points where boundary conditions change. For such problems results should be treated with caution. The choice of function may also depend on whether or not it is to be used in a multithreaded environment.

You may wish to construct your own partial differential equation solution software for problems not solvable by the functions described in Section 3.1 to Section 3.7 below. In such cases you can employ appropriate functions from the Linear Algebra Chapters to solve the resulting linear systems; see Section 3.9 for further details.

3.1 Elliptic Equations

The function nag_pde_2d_laplace (d03ea) solves Laplace's equation in two dimensions, equation (2), by an integral equation method. This function is applicable to an arbitrary domain bounded internally or externally by one or more closed contours, when the value of either the unknown function $u$ or its normal derivative $u'$ is given at each point of the boundary.
The functions `nag_pde_2d_ellip_fd` and `nag_pde_3d_ellip_fd` solve a system of simultaneous algebraic equations of five-point and seven-point molecule form (see Mikhlin and Smolitsky (1967)) on two-dimensional and three-dimensional topologically-rectangular meshes respectively, using Stone’s Strongly Implicit Procedure (SIP). These functions, which make repeated calls of the utility functions `nag_pde_2d_ellip_fd_iter` and `nag_pde_3d_ellip_fd_iter` respectively, may be used to solve any boundary value problem whose finite difference representation takes the appropriate form.

The function `nag_pde_2d_ellip_mgrid` solves a system of seven-point difference equations in a rectangular grid (in two dimensions), using the multigrid iterative method. The equations are supplied by you, and the seven-point form allows cross-derivative terms to be represented (see Mitchell and Griffiths (1980)). The method is particularly efficient for large systems of equations with diagonal dominance and should be preferred to `nag_pde_2d_ellip_fd` whenever it is appropriate for the solution of the problem.

The function `nag_pde_2d_ellip_discret` discretizes a second-order equation on a two-dimensional rectangular region using finite differences and a seven-point molecule. The function allows for cross-derivative terms, Dirichlet, Neumann or mixed boundary conditions, and either central or upwind differences. The resulting seven-diagonal difference equations are in a form suitable for passing directly to the multigrid function `nag_pde_2d_ellip_mgrid`, although other solution methods could just as easily be used.

The function `nag_pde_3d_ellip_helmholtz`, based on the function HW3CRT from FISHPACK (see Swarztrauber and Sweet (1979)), solves the Helmholtz equation in a three-dimensional cuboidal region, with any combination of Dirichlet, Neumann or periodic boundary conditions. The method used is based on the fast Fourier transform algorithm, and is likely to be particularly efficient on vector-processing machines.

### 3.2 Hyperbolic Equations

See Section 3.7.

### 3.3 Parabolic Equations

There are five functions available for solving general parabolic equations in one space dimension:

- `nag_pde_1d_parab_fd` (d03pc),
- `nag_pde_1d_parab_coll` (d03pd),
- `nag_pde_1d_parab_dae_fd` (d03ph),
- `nag_pde_1d_parab_dae_coll` (d03pj),
- `nag_pde_1d_parab_remesh_fd` (d03pp).

Equations may include nonlinear terms but the true derivative $u_t$ should occur linearly and equations should usually contain a second-order space derivative $u_{xx}$. There are certain restrictions on the coefficients to try to ensure that the problems posed can be solved by the above functions.

The method of solution is to discretize the space derivatives using finite differences or collocation, and to solve the resulting system of ordinary differential equations using a “stiff” solver.

The function `nag_pde_1d_parab_fd` and `nag_pde_1d_parab_coll` can solve a system of parabolic equations of the form

$$
\sum_{j=1}^{n} P_{ij}(x,t,U,U_x) \frac{\partial U_j}{\partial t} + Q_i(x,t,U,U_x) = x^{-m} \frac{\partial}{\partial x}(x^m R_i(x,t,U,U_x)),
$$

where $i = 1, 2, \ldots, n$, $a \leq x \leq b$, $t \geq t_0$.

The parameter $m$ allows the function to handle different coordinate systems easily (Cartesian, cylindrical polars and spherical polars). `nag_pde_1d_parab_fd` uses a finite differences spatial discretization and `nag_pde_1d_parab_coll` uses a collocation spatial discretization.
nag_pde_1d_parab_dae_fd (d03ph) and nag_pde_1d_parab_dae_coll (d03pj) are similar to nag_pde_1d_parab_fd (d03pc) and nag_pde_1d_parab_coll (d03pd) respectively, except that they provide scope for coupled differential-algebraic systems. This extended functionality allows for the solution of more complex and more general problems, e.g., periodic boundary conditions and integro-differential equations.

nag_pde_1d_parab_remesh_fd (d03pp) is similar to nag_pde_1d_parab_dae_fd (d03ph) but allows remeshing to take place in the spatial direction. This facility can be very useful when the nature of the solution in the spatial direction varies considerably over time.

For parabolic systems in two space dimensions see Section 3.6.

### 3.4 Black–Scholes Equations

nag_pde_1d_blackscholes_fd (d03nc) solves the Black–Scholes equation

\[
\frac{\partial f}{\partial t} + (r - q)S \frac{\partial f}{\partial S} + \frac{\sigma^2 S^2}{2} \frac{\partial^2 f}{\partial S^2} = rf
\]

for the value \( f \) of a European or American, put or call stock option. The parameters \( r \), \( q \) and \( \sigma \) may each be either constant or time-dependent. The values of the Greeks are also returned.

In certain cases an analytic solution of the Black–Scholes equation is available. In these cases the solution may be computed by nag_pde_1d_blackscholes_closed (d03nd). More generally, Chapter S contains a set of option pricing functions that evaluate the closed form solutions or approximations to the equations that define mathematical models for the prices of selected financial option contracts, including the Black–Scholes equation (nag_specfun_opt_bsm_price (s30aa)).

### 3.5 First-order Systems in One Space Dimension

There are three functions available for solving systems of first-order partial differential equations:

- nag_pde_1d_parab_keller (d03pe),
- nag_pde_1d_parab_dae_keller (d03pk),
- nag_pde_1d_parab_remesh_keller (d03pr).

Equations may include nonlinear terms but the time derivative should occur linearly. There are certain restrictions on the coefficients to ensure that the problems posed can be solved by the above functions.

The method of solution is to discretize the space derivatives using the Keller box scheme and to solve the resulting system of ordinary differential equations using a ‘stiff’ solver.

nag_pde_1d_parab_keller (d03pe) is designed to solve a system of the form

\[
\sum_{j=1}^{n} P_{ij}(x,t,U,U_z) \frac{\partial U_j}{\partial t} + Q_i(x,t,U,U_z) = 0,
\]

where \( i = 1,2,\ldots,n, \ a \leq x \leq b, \ t \geq t_0. \)

nag_pde_1d_parab_dae_keller (d03pk) is similar to nag_pde_1d_parab_keller (d03pe) except that it provides scope for coupled differential algebraic systems. This extended functionality allows for the solution of more complex problems.

nag_pde_1d_parab_remesh_keller (d03pr) is similar to nag_pde_1d_parab_dae_keller (d03pk) but allows remeshing to take place in the spatial direction. This facility can be very useful when the nature of the solution in the spatial direction varies considerably over time.

nag_pde_1d_parab_keller (d03pe), nag_pde_1d_parab_dae_keller (d03pk) or nag_pde_1d_parab_remesh_keller (d03pr) may also be used to solve systems of higher or mixed order partial differential equations which have been reduced to first-order. Note that in general these functions are unsuitable for hyperbolic first-order equations, for which an appropriate upwind discretization scheme should be used (see Section 3.7 for example).
3.6 Second-order Systems in Two Space Dimensions

There are two functions available for solving nonlinear second-order time-dependent systems in two space dimensions:

- `nag_pde_2d_gen_order2_rectangle (d03ra)`,
- `nag_pde_2d_gen_order2_rectilinear (d03rb)`. 

These functions are formally applicable to the general nonlinear system

\[ F_j(t, x, y, u, u_t, u_x, u_y, u_{xx}, u_{xy}, u_{yy}) = 0, \]

where \( j = 1, 2, \ldots, npde \), \((x, y) \in \Omega, t_0 \leq t \leq t_{out}\). However, they should not be used to solve purely hyperbolic systems, or time-independent problems.

`nag_pde_2d_gen_order2_rectangle (d03ra)` solves the nonlinear system in a rectangular domain, while `nag_pde_2d_gen_order2_rectilinear (d03rb)` solves in a rectilinear region, i.e., a domain bounded by perpendicular straight lines.

Both functions use the method of lines and solve the resulting system of ordinary differential equations using a backward differentiation formula (BDF) method, modified Newton method, and BiCGSTAB iterative linear solver. Local uniform grid refinement is used to improve accuracy.

Utility functions `nag_pde_2d_gen_order2_checkgrid (d03ry)` and `nag_pde_2d_gen_order2_rectilinear_extractgrid (d03rz)` may be used in conjunction with `nag_pde_2d_gen_order2_rectilinear (d03rb)` to check the user-supplied initial mesh, and extract mesh coordinate data.

3.7 Convection-diffusion Systems

There are three functions available for solving systems of convection-diffusion equations with optional source terms:

- `nag_pde_1d_parab_convdiff (d03pf)`,
- `nag_pde_1d_parab_convdiff_dae (d03pl)`,
- `nag_pde_1d_parab_convdiff_remesh (d03ps)`. 

Equations may include nonlinear terms but the time derivative should occur linearly. There are certain restrictions on the coefficients to ensure that the problems posed can be solved by the above functions, in particular the system must be posed in conservative form (see below). The functions may also be used to solve hyperbolic convection-only systems.

Convection terms are discretized using an upwind scheme involving a numerical flux function based on the solution of a Riemann problem at each mesh point (see LeVeque (1990)); and diffusion and source terms are discretized using central differences. The resulting system of ordinary differential equations is solved using a ‘stiff’ solver. In the case of Euler equations for a perfect gas various approximate and exact Riemann solvers are provided in `nag_pde_1d_parab_euler_roe (d03pu)`, `nag_pde_1d_parab_euler_osher (d03pv)`, `nag_pde_1d_parab_euler_hll (d03pw)` and `nag_pde_1d_parab_euler_exact (d03px)`. These functions may be used in conjunction with `nag_pde_1d_parab_convdiff (d03pf)`, `nag_pde_1d_parab_convdiff_dae (d03pl)` and `nag_pde_1d_parab_convdiff_remesh (d03ps)`. 

`nag_pde_1d_parab_convdiff (d03pf)` is designed to solve systems of the form

\[ \sum_{j=1}^{n} P_{ij}(x, t, U) \frac{\partial U_j}{\partial t} + \frac{\partial}{\partial x} F_i(x, t, U) = C_i(x, t, U) \frac{\partial}{\partial x} D_i(x, t, U, U_x) + S_i(x, t, U), \]

or hyperbolic convection-only systems of the form

\[ \sum_{j=1}^{n} P_{ij}(x, t, U) \frac{\partial U_j}{\partial t} + \frac{\partial}{\partial x} F_i(x, t, U) = 0, \]

where \( i = 1, 2, \ldots, n, a \leq x \leq b, t \geq t_0. \)
nag_pde_1d_parab_convdiff_dae (d03pl) is similar to nag_pde_1d_parab_convdiff (d03pf) except that it provides scope for coupled differential algebraic systems. This extended functionality allows for the solution of more complex problems.

nag_pde_1d_parab_convdiff_remesh (d03ps) is similar to nag_pde_1d_parab_convdiff_dae (d03pl) but allows remeshing to take place in the spatial direction. This facility can be very useful when the nature of the solution in the spatial direction varies considerably over time.

3.8 Automatic Mesh Generation

The function nag_pde_2d_triangulate (d03ma) places a triangular mesh over a given two-dimensional region. The region may have any shape and may include holes. It may also be used in conjunction with functions from the NAG Finite Element Library. A wider range of mesh generation functions are available in Chapter D06.

3.9 Utility Functions

nag_pde_2d_ellip_fd_iter (d03ua) (nag_pde_3d_ellip_fd_iter (d03ub)) calculates, by the Strongly Implicit Procedure, an approximate correction to a current estimate of the solution of a system of simultaneous algebraic equations for which the iterative update matrix is of five (seven) point molecule form on a two- (three-) dimensional topologically-rectangular mesh.

Functions are available in the Linear Algebra Chapters for the direct and iterative solution of linear equations. Here we point to some of the functions that may be of use in solving the linear systems that arise from finite difference or finite element approximations to partial differential equation solutions. Chapters F01, F04, F07, F08 and F11 should be consulted for further information and for the appropriate function documents. Decision trees for the solution of linear systems are given in Section 4 in the F04 Chapter Introduction.

The following functions allow the direct solution of symmetric positive definite systems:

Band  
nag_lapack_dpbtrf (f07hd) and nag_lapack_dpbtrs (f07he)

Variable band (skyline)  
nag_matop_real_vband_posdef_fac (f01mc) and nag_linsys_real_posdef_vband_solve (f04mc)

Tridiagonal  
nag_lapack_dptsv (f07ja), nag_lapack_dpttrf (f07jd) and nag_lapack_dpttrs (f07je)

Sparse  
f11ja* and nag_sparse_real_symm_precon_ichol_solve (f11jb)

(‘* the description of nag_sparse_real_symm_precon_ichol_solve (f11jb) explains how nag_sparse_real_symm_precon_ichol (f11ja) should be called to obtain a direct method) and the following functions allow the iterative solution of symmetric positive definite and symmetric-indefinite systems:

Sparse  
nag_sparse_real_symm_basic_setup (f11gd), nag_sparse_real_symm_basic_solver (f11ge), nag_sparse_real_symm_basic_diag (f11gf), nag_sparse_real_symm_precon_ichol (f11ja), nag_sparse_real_symm_solve_ichol (f11jc) and nag_sparse_real_symm_solve_jacssor (f11je).

The latter two functions above are black box functions which include Incomplete Cholesky, SSOR or Jacobi preconditioning.

The following functions allow the direct solution of nonsymmetric systems:

Band  
nag_lapack_dgbtrf (f07bd) and nag_lapack_dgbtrs (f07be)

Almost block-diagonal  
nag_matop_real_gen_blkdiag_lu (f01lh) and nag_linsys_real_blkdiag_fac_solve (f04lh)

Tridiagonal  
nag_matop_real_gen_tridiag_lu (f01le), nag_linsys_real_tridiag_fac_solve (f04le) or nag_lapack_dgtsv (f07ca)

Sparse  
nag_matop_real_gen_sparse_lu (f01br) (and nag_matop_real_gen_sparse_lu_reuse (f01bs)) and nag_linsys_real_sparse_fac_solve (f04ax)

and the following functions allow the iterative solution of nonsymmetric systems:
Sparse & nag_sparse_real_gen_basic_setup (f11bd), nag_sparse_real_gen_basic_solver (f11be), nag_sparse_real_gen_basic_diag (f11bf), nag_sparse_real_gen_precon_ilu (f11da), nag_sparse_real_gen_solve_ilu (f11dc) and nag_sparse_real_gen_solve_jacssor (f11de).

The latter two functions above are black box functions which include incomplete LU, SSOR and Jacobi preconditioning.

The functions nag_pde_1d_parab_coll_interp (d03py) and nag_pde_1d_parab_fd_interp (d03pz) use linear interpolation to compute the solution to a parabolic problem and its first derivative at the user-specified points. nag_pde_1d_parab_fd_interp (d03pz) may be used in conjunction with nag_pde_1d_parab_fd (d03pc), nag_pde_1d_parab_keller (d03pe), nag_pde_1d_parab_dae_fd (d03ph), nag_pde_1d_parab_dae_keller (d03pk), nag_pde_1d_parab_remesh_fd (d03pp) and nag_pde_1d_parab_remesh_keller (d03pr). nag_pde_1d_parab_coll_interp (d03py) may be used in conjunction with nag_pde_1d_parab_coll (d03pd) and nag_pde_1d_parab_dae_coll (d03pj).

nag_pde_2d_gen_order2_checkgrid (d03ry) and nag_pde_2d_gen_order2_rectilinear_extractgrid (d03rz) are utility functions for use in conjunction with nag_pde_2d_gen_order2_rectilinear (d03rb). They can be called to check the user-specified initial mesh and to extract mesh coordinate data.

### 4 Decision Trees

**Tree 1**

- Does PDE have a time derivative?  
  - Yes  
  - Does PDE have second derivatives?  
    - Yes  
      - see Tree 4 Parabolic branch
    - No  
      - Is PDE hyperbolic?  
        - Yes  
          - see Tree 3 Hyperbolic branch
        - No  
          - 1 space dimension?  
            - Yes  
              - Does PDE have coupled ODEs?  
                - Yes  
                  - Is a remeshing process required?  
                    - Yes  
                      - d03pr
                    - No  
                      - d03pk
                - No  
                  - N/A
            - No  
              - d03pe
      - No  
        - see Tree 2 Elliptic branch

- No
5 Functionality Index

Automatic mesh generation, triangles over a plane domain .......................................................... d03ma

Black–Scholes equation, analytic .................................................................................................. d03nd
definite difference .................................................................................................................. d03nc

Convection-diffusion system(s), nonlinear, one space dimension,
using upwind difference scheme based on Riemann solvers .................................................. d03pf
using upwind difference scheme based on Riemann solvers, with coupled differential algebraic system .......................................................... d03pl
with remeshing ....................................................................................................................... d03ps

Elliptic equations,
discretization on rectangular grid (seven-point two-dimensional molecule).................................... d03ee
equations on rectangular grid (seven-point two-dimensional molecule) ........................................ d03ed
finite difference equations (five-point two-dimensional molecule) .............................................. d03eb
finite difference equations (seven-point three-dimensional molecule) ......................................... d03ec
Helmholtz's equation in three dimensions .................................................................................. d03fa
Laplace's equation in two dimensions ........................................................................................ d03ea

First-order system(s), nonlinear, one space dimension,
using Keller box scheme .......................................................................................................... d03pe
using Keller box scheme, with coupled differential algebraic system ........................................ d03pk
with remeshing ....................................................................................................................... d03pr

PDEs, general system, one space variable, method of lines, parabolic,
collocation spatial discretization, coupled DAEs, comprehensive ........................................... d03pj
easy-to-use ................................................................................................................................. d03pd
finite differences spatial discretization, coupled DAEs, comprehensive .................................. d03ph
coupled DAEs, remeshing, comprehensive .............................................................................. d03pp
easy-to-use ................................................................................................................................. d03pc
Second order system(s),
    nonlinear,
    two space dimensions,
    in rectangular domain ................................................................. d03ra
    in rectilinear domain ..................................................................... d03rb
Utility function,
    average values for d03nd.............................................................. d03ne
    basic SIP for five-point two-dimensional molecule.............................. d03ua
    basic SIP for seven-point three-dimensional molecule............................ d03ub
    check initial grid data for d03rb...................................................... d03ry
    exact Riemann solver for Euler equations .......................................... d03px
    HLL Riemann solver for Euler equations ............................................ d03pw
    interpolation function for collocation scheme ..................................... d03py
    interpolation function for finite difference,
        Keller box and upwind scheme..................................................... d03pz
    Osher's Riemann solver for Euler equations ....................................... d03pv
    return coordinates of grid points for d03rb....................................... d03rz
    Roe's Riemann solver for Euler equations .......................................... d03pu

6 References


