Solving partial differential equations using the NAG Library

Jeremy Walton

The Numerical Algorithms Group, Ltd.

Wilkinson House, Jordan Hill Road

Oxford OX2 8DR, United Kingdom

1. Introduction

A partial differential equation (PDE) is a mathematical relation which involves functions of multiple variables and their partial derivatives. PDEs are used to formulate (and hence to aid in the solution of) problems involving functions of several variables, and they arise in a variety of important fields. For example, in physics, they are used to describe the propagation of sound or heat, electrostatics, electrodynamics, fluid flow and elasticity, whilst in finance, they have been used in the modelling of the pricing of financial options. Accordingly, the study of their properties and methods of solution has received a great deal of attention [1].

This note describes some applications of the NAG Library [2] to the solution of PDEs. After giving some technical details, including their classification (§2.1), subsidiary conditions (§2.2), and methods of solution (§2.3), we present a few example PDEs and their solutions in §3. These are classified according to the type of equation; some details about the appropriate NAG routine to be used for its solution are also presented. The example results are generated using routines from the NAG Toolbox for *MATLAB* ® [3] and plotted using tools in that environment.

The problems described in §3 are treatable using the numerical method of finite differences (as implemented, for example, in the PDE chapter of the NAG Library [4]) because of the characteristics of the geometry of the domain over which the PDE is defined. Other types of problem (having, for example, domains of irregular geometry) require the application of the so-called finite element method. In §4, we describe the two components of this method – namely, the generation of a mesh over the problem domain (§4.1), and the transformation of the PDE into a set of simultaneous equations (§4.2) – and indicate how routines from the NAG Library [5] [6] [7] can be of assistance in their implementation [8]. A final section (§5) contains our conclusions.

2. Partial Differential Equations – technical background

2.1 Classification

An example of a PDE is the *Laplace equation* in two dimensions:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{1}$$

where $u(x,y)$ is a function of location in 2D space. This equation describes the behaviour of *potential fields* in areas such as gravitation, electrostatics and fluid dynamics. It can be written using a more compact notation as:

$$u_{xx} + u_{yy} = 0. \tag{2}$$

or, even more compactly, as $\nabla^2 u = 0$. Laplace's equation is an example of a so-called *equilibrium* PDE, because it describes a function that doesn't depend on time and thus can be used to model an unchanging physical system (for example, it governs the distribution of heat in a given region in a steady state). By contrast, a PDE such as the *wave equation* in one dimension is a *dynamical* PDE which models a time-varying process:

$$u_{tt} = v^2 u_{xx}. \tag{3}$$

This describes the behaviour of the displacement $u(t,x)$, a function of time t and location x, with v a constant corresponding to the velocity of the wave. It could be used to describe, for example, the displacement of a stretched string from equilibrium, or the magnitude of an electric field in a tube.

Finally, we note that the *order* of a PDE is that of the highest-order derivative which it contains; thus, both (2) and (3) are second order PDEs. The majority of PDEs that appear in scientific applications are of this kind. The general expression for a linear second order PDE can be written as

$$\alpha u_{xx} + 2\beta u_{xy} + \gamma u_{yy} + \delta u_x + \varepsilon u_y + \zeta u + \eta = 0, \tag{4}$$

where the coefficients $\alpha, \beta, \gamma, \delta, \varepsilon, \zeta, \eta$ are functions of the two independent variables x and y only (of course, for a dynamical PDE, one of these will be t). If the discriminant $\alpha\gamma - \beta^2$ is positive, the PDE is called *elliptic*; if it's negative, it's called *hyperbolic*, and if it's zero, it's called *parabolic*. According to this definition, it can be seen that Laplace's equation (2) – in which $\alpha(x, y) = \gamma(x, y) = 1$ and $\beta(x, y) = 0$ – is elliptic, while the wave equation (3) – where $\alpha(t, x) = 1, \gamma(t, x) = -1$ and $\beta(t, x) = 0$ – is hyperbolic. An example of a parabolic PDE is provided by the one-dimensional *heat conduction* equation

$$u_t = \kappa u_{xx}, \tag{5}$$

which governs the evolution of heat – or, equivalently, temperature – $u(t, x)$ as a function of time and location within a given region. Here, κ is a constant related to the thermal diffusivity. Note that a variant of this PDE arises in the solution of the Black-Scholes PDE for financial option pricing.

2.2 Subsidiary conditions

In general, there are many functions which satisfy a given PDE – for example, in the case of (5), two of these are

$$u(t, x) = \kappa t + \frac{1}{2}x^2, \tag{6}$$

and

$$u(t, x) = \frac{e^{-x^2/4\kappa t}}{2\sqrt{\pi\kappa t}}. \tag{7}$$

Determining which solution is appropriate requires extra information, including the details of D, the domain of interest over which u is defined (including a specification of C, the contour that bounds the domain) and subsidiary conditions on u and its derivatives. The latter could take the form of *boundary* conditions – which specify the form of the solution on the boundary of the domain, and/or *initial* conditions – which describe the solution at the starting time.

More specifically, for equilibrium equations such as (2), the subsidiary conditions provide information about the solution at all points on C – thus, a solution may be sought subject to the condition

$$u(x,y) = f(x,y) \qquad \forall\ (x,y) \in C, \tag{8}$$

where $f(x,y)$ is a given function; this is known as a *Dirichlet* boundary condition. A condition that specifies the values of the derivative of the solution on C,

$$\nabla u(x,y) \cdot \boldsymbol{n}(x,y) = g(x,y) \qquad \forall\ (x,y) \in C \tag{9}$$

is called a *Neumann* boundary condition. Here, $g(x,y)$ is a given function, ∇ is the gradient vector operator, the dot denotes the inner product and $\boldsymbol{n}$ is the normal to C. In this case, a further piece of information (such as the value of the solution at a particular location) is necessary to determine a unique solution. Finally, a *Robin* boundary condition can be seen as a weighted combination of the other two:

$$\nabla u \cdot \boldsymbol{n} + u\, f_1 = f_2 \qquad \forall\ (x,y) \in C, \tag{10}$$

where $f_1(x,y)$ and $f_2(x,y)$ are given functions.

For dynamical equations such as (3), the subsidiary conditions provide information about the solution at some initial time. Thus, for example, if (3) is satisfied for $t > 0$, then a solution may be sought subject to

$$u(0,x) = f(x), \qquad u_t(0,x) = g(x), \tag{11}$$

where $f(x)$ and g(x) are given functions. A problem formulated in this way is usually known as an *initial value problem* (sometimes referred to as *Cauchy's problem*).

Other examples of subsidiary conditions might specify both spatial and temporal constraints. Suppose that (5) is satisfied for $t > 0$ and $0 < x < 1$, then a solution may be sought which satisfies

$$u(0,x) = f(x), \qquad 0 < x < 1, \tag{12}$$

and

$$\begin{aligned} u(0,x) &= 0 \\ u_t(t,1) &= 1, \qquad t > 0. \end{aligned} \tag{13}$$

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

2.3 Solution methods

A wide variety of methods have been developed for the solution of PDEs [1] [9]. Analytic methods include [10] the separation of variables, the method of characteristics and integral transformation (such as, for example, Fourier analysis), whilst finite differences and finite elements are among the most useful numerical methods [11]. For linear equilibrium PDEs such as (2), their application results in a system of simultaneous equations that can be treated using linear system solvers. Dynamical PDEs are often tackled by using these methods to discretize the spatial coordinates (leaving t continuous), generating a system of ordinary differential equations (ODEs) which can be solved via appropriate techniques (see §§3.2, 3.3, and 3.4 below). Alternative solution methods are available for special classes of problems.

3. Examples

In this section we present some example PDE problems and their solutions using routines from the ***d03*** chapter [4] of the NAG Library.

3.1 3D Elliptic PDE

We seek a solution of the Laplace equation in three dimensions:

$$u_{xx} + u_{yy} + u_{zz} = 0 \tag{14}$$

in a rectangular box having a non-uniform grid spacing in each direction, subject to the Dirichlet condition

$$u(x,y,z) = e^{\frac{1+x}{Y}} \cos\left(\frac{y\sqrt{2}}{Y}\right) e^{\frac{-1-x}{Y}} \qquad \forall\ (x,y,z) \in C, \tag{15}$$

where Y is the length of the box in the y direction. Problems such as this can be

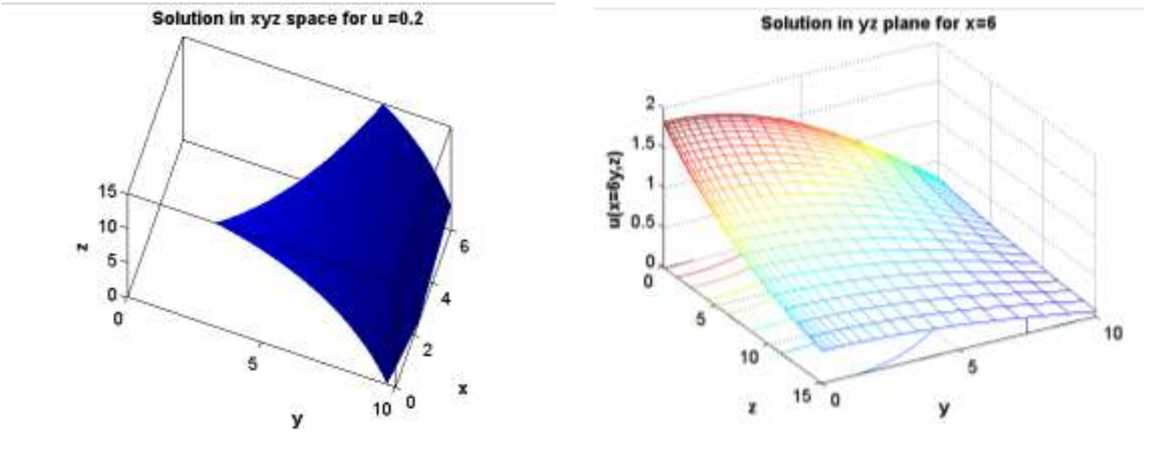

Figure 1. Solution to (14), displayed as an isosurface (left) and a 2D slice at constant x (right).

tackled by the finite difference method: briefly, this involves imposing a mesh on D, and representing the function by its values at the discrete set of mesh points; the PDE can then be approximated in terms of the difference between values at neighbouring points. This transforms the PDE into a system of simultaneous algebraic equations which can then be solved using, for example, the NAG routine ***d03ec*** [12]. This employs the strongly implicit procedure in the solution of the equations. Figure 1 shows the solution, displayed as an isosurface (i.e. a contour in 3D space connecting all points in the box where the function has a specific value) and as a surface plot showing how the function varies across a 2D slice taken through the box. Figure 2 shows how the function varies along the y direction in the box at fixed z, and for a set of x values.

3.2 Advection-Diffusion equation

The simplest form of the so-called advection-diffusion equation in one dimension is

$$u_t = Cu_x + Du_{xx} \qquad (16)$$

which states that the dynamic evolution of – say – a species concentration $u(t, x)$ is given by the sum of an advective term and a diffusion term [compare this PDE to the diffusion equation (5)]. It will be recalled that advection is the transport of the species due to the bulk motion of the underlying fluid, whilst diffusion describes the spread of species along a concentration gradient.

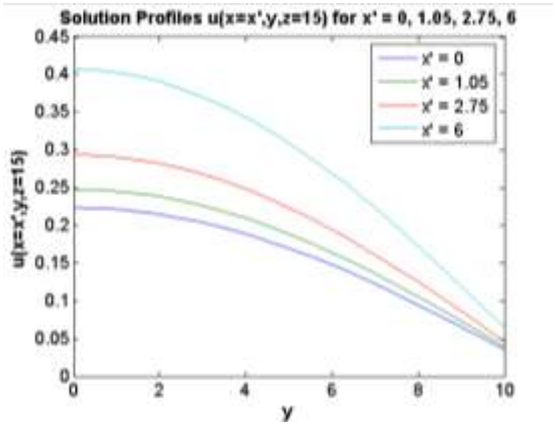

Figure 2. Solution to (14), displayed as a plot of the function at constant z, for a set of x values.

As noted in §2.3, PDEs such as this may be solved by discretizing the space dimension (but not the time), which transforms them into a system of ODEs whose solution may be obtained using a stiff solver. In fact, the NAG routines solve a more general form of (16), specifically

$$\sum_{j=1}^{n} P_{ij}\frac{\partial u_j}{\partial t} + \frac{\partial F_i}{\partial x} = C_i\frac{\partial D_i}{\partial x} + S_i \qquad (17)$$

where n is the number of PDEs in the system, and the number of solutions $u_i(t, x)$. Here, D_i is a function of (amongst other things) $\partial u_i/\partial x$, so the first term on the right contains a term proportional to $\partial^2 u_i/\partial x^2$ – i.e. the diffusion term . Similarly, F_i depends on u_i, so the second term on the left is the advection term, whilst S_i is a source term.

Our example problem is a simple model of the advection and diffusion of a cloud of (single component) material:

$$u_t + \varphi u_x = \delta u_{xx} \tag{18}$$

for $x \in [0,1]$, subject to the boundary conditions

$$u(t,0) = u(t,1) = 0 \tag{19}$$

and the initial conditions

$$u(0,x) = \sin\left(\pi \frac{x-a}{b-a}\right), \qquad a \le x \le b \tag{20}$$

with $a = 0.2, b = 0.4$, $u(0,x) = 0$ elsewhere and φ and δ physical parameters. Comparing (18) to the more general formulation (17) identifies F with φu and D with δu_x.

As noted above, problems like this can be solved by reducing the PDE to a system of ODEs (specifically in this example, by using the so-called method of lines). That system can then be solved using, for example, a Backward Differentiation Formula (BDF). The NAG routine ***d03ps*** [13] uses this technique, and also incorporates automatic adaptive remeshing of the spatial grid. Figure 3 shows the solution to (18), as calculated by ***d03ps***; the motion of the concentration peak towards larger x as time is increased can clearly be seen.

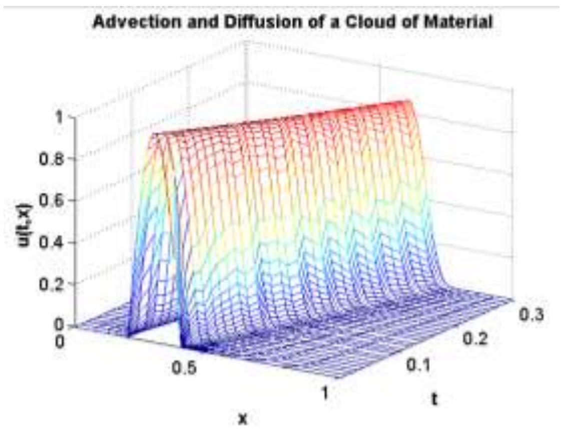

Figure 3. Solution to (18), plotted as an (x,t) surface.

3.3 Parabolic PDE

A general system of parabolic PDEs in one spatial dimension may be written as

$$\sum_{j=1}^{n} P_{ij} \frac{\partial u_j}{\partial t} + Q_i = x^{-m} \frac{\partial (x^m R_i)}{\partial x} \tag{21}$$

where n is the number of PDEs in the system, and the terms P_{ij}, Q_i and R_i are each functions of t, x, u and u_x, in general. The parameter m allows for the treatment of different coordinate systems – specifically: Cartesian ($m = 0$), cylindrical polar ($m = 1$) and spherical polar ($m = 2$).

Such a system can be solved by discretizing the space derivatives and solving the resulting set of ODEs using a stiff solver (such as, for example, a BDF). There are a variety of methods which can be used for discretization: thus, for example, the NAG routine ***d03pc*** [14] uses finite differences, while ***d03pd*** [15] uses a Chebyshev collocation method.

Our example [14] is an elliptic-parabolic pair of PDEs:

$$\begin{aligned} \frac{1}{r}\frac{\partial}{\partial r}\left(r^2\frac{\partial u_1}{\partial r}\right) &= 4\alpha\left(u_2 + r\frac{\partial u_2}{\partial r}\right) \\ (1-r^2)\frac{\partial u_2}{\partial t} &= \frac{1}{r}\frac{\partial}{\partial r}\left(r\left[\frac{\partial u_2}{\partial r} - u_1u_2\right]\right) \end{aligned} \tag{22}$$

for $t, r \in [0,1]$, subject to the boundary conditions

$$\begin{aligned} u_1(t,0) &= \frac{\partial u_2(t,0)}{\partial r} = 0 \\ u_2(t,1) &= \frac{\partial u_1(t,1)}{\partial r} = 0 \end{aligned} \tag{23}$$

and the initial conditions

$$u_1(0,r) = 2\alpha r, \quad u_2(0,r) = 1, \qquad 0 \le r \le 1 \tag{24}$$

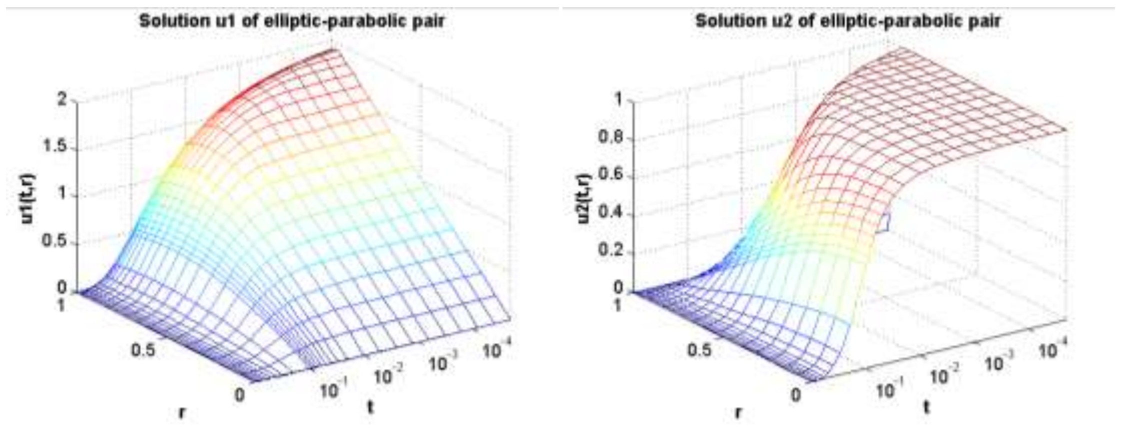

Figure 4. Solution to (22), plotted as a pair of (r,t) surfaces.

A comparison of (22) with (21) gives

$$P_{22} = (1 - r^2), \qquad 0 \text{ otherwise}.$$

$$Q_1 = 4\alpha(u_2 + r\,\partial u_2/\partial r), \qquad Q_2 = 0. \tag{25}$$

$$R_1 = r\,\partial u_1/\partial r, \quad R_2 = \partial u_2/\partial r - u_1 u_2.$$

Figure 4 shows the solution to this system of PDEs, as calculated by ***d03pc***.

This routine has also been used in the solution of [16]:

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(D\frac{\partial u}{\partial x}\right) - \frac{\partial h}{\partial x} + g \tag{26}$$

where D, h and g are functions of u (which is density in the model system). The three functions correspond to diffusion, advection and reaction; a comparison with (21) gives

$$R = Du_x - h, \quad Q = -g. \tag{27}$$

3.4 Time-dependent PDE in 2D

Some PDEs can be classified as time-dependent, with two spatial dimensions. In general, they can be written as

$$F_j\left(t, x, y, u, u_t, u_x, u_y, u_{xx}, u_{xy}, u_{yy}\right) = 0, \qquad j = 1,2 \ldots n \tag{28}$$

where n is the number of PDEs in the system, and the number of solutions $u_j(t, x, y)$. We are interested in systems defined on a rectangular spatial domain, D:

$$x_{min} \le x \le x_{max}; \qquad y_{min} \le y \le y_{max} \tag{29}$$

The initial treatment of problems such as this involves the use of the finite difference method (as before) by imposing a mesh on D. This reduces the system to a set of ODEs, which may be integrated in time using an implicit BDF method with variable step size. The nonlinear equations resulting from the time integration may be solved using a modified Newton method. The NAG

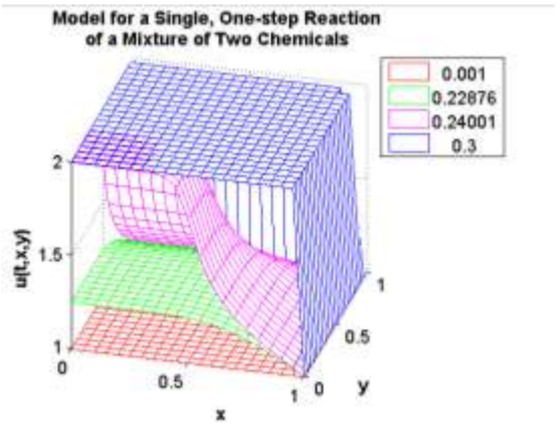

Figure 5. Solution to (23), plotted as an (x,y) surface for various values of t.

routine ***d03ra*** [17] uses these techniques, and also incorporates local uniform grid refinement to improve the accuracy of the solution in regions where this is changing rapidly.

Our first example problem is a model for a single, one-step reaction of a mixture of two chemicals in a two-dimensional region of space [17]. The PDE for $u(t,x,y)$, the dynamic local temperature of the mixture, is

$$u_t = d\left(u_{xx} + u_{yy}\right) + \frac{Re^{\delta}}{\alpha\delta}(1 + \alpha - u)e^{-\delta/u} \tag{30}$$

where d, R, α and δ are physical parameters associated with the reaction mechanism. The domain is defined as

$$x_{min} = y_{min} = 0; \qquad x_{max} = y_{max} = 1 \tag{31}$$

and the initial conditions are $u(0,x,y) = 1$ throughout the domain. The boundary conditions are

$$\begin{aligned} &u_x(t,x_{min},y) = 0,\ \ u(t,x_{max},y) = 1; \qquad && y_{min} \le y \le y_{max} \\ &u_y(t,x,y_{min}) = 0,\ \ u(t,x,y_{max}) = 1; \qquad && x_{min} \le x \le x_{max} \end{aligned} \tag{32}$$

Figure 5 shows the solution of (30), as calculated by ***d03ra***. It can be seen that the temperature gradually increases in a circular region around the origin (assuming that we are modelling a single quadrant of the full domain), before ignition takes place at around $t = 0.24$, when the temperature suddenly jumps to 2.0 and a reaction front forms and propagates outwards.

This has also been applied [16] to the solution of

$$u_t = d\left(u_{xx} + u_{yy}\right) + f(u) \tag{33}$$

where f is a concave function on the interval [0,1] with $f(0) = f(1) = 0$ and $f > 0$ on (0,1).

4. Other approaches

The ***d03*** routines described above use finite differencing techniques, and are applicable to a limited number of specific types of PDE which are to be solved on domains having regular geometries. One approach to handling irregular geometries is the so-called finite element method (FEM), which is a technique [18] for solving

PDEs (or other variational problems) by first discretizing the equations in their spatial dimensions. It has two components: first, the subdivision of the spatial domain into a mesh, and second, the transformation of the PDE into a set of simultaneous equations. We consider each separately.

4.1 Mesh generation

The subdivision of the domain over which a PDE is to be solved is a key requirement of the finite element method. In many cases, the accuracy and validity of a solution is strongly tied to the properties of the underlying mesh.

Generating the mesh over the domain is (usually) a two-stage process [5]: firstly, meshing the domain boundary, and secondly, using the boundary mesh points as starting points in the generation of a mesh within the domain. The NAG routine ***d06ba*** [19] can be used to generate a boundary mesh; we note that the boundary may be of an arbitrary shape, and there may be more than one boundary for a given domain (that is, the domain can contain holes which are specified by *internal* boundaries). A variety of different methods are available for generating the mesh within the domain: three of these are incremental (as used by ***d06aa*** [20]), Delaunay-Voronoi (***d06ab*** [21]) and advancing front (***d06ac*** [22]).

Finally, we note that the ***d03ma*** routine [23] generates a triangular mesh over a 2D region in a single call.

4.2 PDE transformation

Having meshed the domain, the second step in the finite element method is to use the elements of the mesh (typically, triangles) in the transformation of the PDE to a system of simultaneous equations. This is done by first choosing a set of basis functions to represent the equation within each element, which results in a matrix equation that relates the inputs at each node of the element to the outputs at the same points. The number of coefficients in the overall system of equations for the entire domain is commensurate with the size of the mesh – i.e., typically, of the order of thousands. However, the matrix of coefficients is also *banded* because only the diagonal and a few non-diagonal elements are non-zero. The NAG Library contains (in the ***f07*** chapter [6]) routines for the solution of this type of system which take advantage of its structure.

Alternatively, because it could also be described as *sparse*, special methods for solving such a system (which, for example, exploit the sparseness by only storing the non-zero elements) can be used to solve it, and this is the route we follow here, using routines from the ***f11*** chapter of the NAG Library [7]. Thus, for example, ***f11jc*** [24] is a so-called black box routine that calls five other ***f11*** routines to solve a real sparse symmetric linear system using either a conjugate gradient method, or the Lanczos method. The routine ***f11ja*** [25] must be called before using ***f11jc*** in order to compute the incomplete Cholesky factorization of the matrix (this preconditioning step increases the speed and efficiency of the solver); in addition, calling ***f11zb*** [26] prior to ***f11ja*** further increases the efficiency of the solver by reordering the non-zero elements of the matrix.

The ***d06*** and ***f11*** routines have been used in a demo PDE solver [8] which uses the finite element method to solve (1) – and its variant having a non-zero right-hand-side, namely Poisson's equation – over a user-specified 2D domain of arbitrary shape. The solver is distributed as part of the NAG Toolbox for *MATLAB*.

Finally, we note that further technical background on the use of NAG's mesh generators and sparse solver routines for solving PDEs can be found at [27].

5. Conclusions

In this note, we have described the characteristics of PDEs, including their uses, classification, subsidiary conditions and some of the ways in which they may be solved. In this context, we have shown how routines from the NAG Library can be used in their numerical solution. These routines have come not only from the Library's PDE chapter (***d03***), but also from those that deal with mesh generation (***d06***) and the solution of large linear systems that are either banded (***f07***) or sparse (***f11***); these three chapters are applicable in the implementation of the finite element method, which may be used in cases where the complexity of the geometry of the domain over which the PDE is to be solved prevents the application of finite differencing methods (as used, for example, in the ***d03*** chapter).

Finally, it could perhaps be pointed out that, besides the contents that have been mentioned here, the NAG Library [2] contains user-callable routines for treatment of a wide variety of numerical and statistical problems, including – for example –

optimization, curve and surface fitting, quadrature, correlation and regression analysis and random number generation.

6. Acknowledgements

We thank Dr Judith Perez-Velazquez (Institute of Biomathematics and Biometry, Helmholtz Zentrum, Munich) for providing some of the examples, Dr Lawrence Mulholland and Dr David Sayers (NAG) for helpful discussions whilst preparing this note, and David Cassell and Mike Hooper (NAG) for insightful comments on the text.

7. Bibliography

1. **Fox, L, [ed.].** *Numerical Solution of Ordinary and Partial Differential Equations.* Oxford : Pergamon Press, 1962.

2. **Numerical Algorithms Group.** NAG numerical components. [Online] http://www.nag.com/numeric/numerical_libraries.asp.

3. —. The NAG Toolbox for *MATLAB*. [Online] http://www.nag.com/numeric/MB/start.asp.

4. —. D03 - Partial Differential Equations. NAG chapter introduction. [Online] http://www.nag.com/numeric/fl/nagdoc_fl23/pdf/D03/d03intro.pdf.

5. —. D06 - Mesh Generation. NAG chapter introduction. [Online] http://www.nag.com/numeric/fl/nagdoc_fl23/pdf/D06/d06intro.pdf.

6. —. F07 - Linear Equations. NAG chapter introduction. [Online] http://www.nag.com/numeric/fl/nagdoc_fl23/pdf/F07/f07intro.pdf.

7. —. F11 - Large Scale Linear Systems. NAG chapter introduction. [Online] http://www.nag.com/numeric/fl/nagdoc_fl23/pdf/F11/f11intro.pdf.

8. **Esteves, Nicolas, Fenton, Nathaniel and Walton, Jeremy.** Using the NAG Toolbox for *MATLAB* - Part 4. [Online] November 2009. http://www.nag.co.uk/IndustryArticles/usingtoolboxmatlabpart4.asp.

9. **Press, William H., et al.** *Numerical Recipes: The Art of Scientific Computing.* Cambridge : Cambridge University Press, 1986.

10. **Wikipedia.** Analytical methods to solve PDEs. [Online] http://en.wikipedia.org/wiki/Partial_differential_equation#Analytical_methods_to_solve_PDEs.

11. —. Numerical methods to solve PDEs. [Online] http://en.wikipedia.org/wiki/Partial_differential_equation#Numerical_methods_to_solve_PDEs.

12. **Numerical Algorithms Group.** D03ECF NAG routine document. [Online] http://www.nag.com/numeric/fl/nagdoc_fl23/pdf/D03/d03ecf.pdf.

13. —. D03PSF NAG routine document. [Online] http://www.nag.com/numeric/fl/nagdoc_fl23/pdf/D03/d03psf.pdf.

14. —. D03PCF NAG routine document. [Online] http://www.nag.com/numeric/fl/nagdoc_fl23/pdf/D03/d03pcf.pdf.

15. —. D03PDF NAG routine document. [Online] http://www.nag.com/numeric/fl/nagdoc_fl23/pdf/D03/d03pdf.pdf.

16. **Perez-Velazquez, Judith.** 2011. Personal communication.

17. **Numerical Algorithms Group.** D03RAF NAG routine document. [Online] http://www.nag.com/numeric/fl/nagdoc_fl23/pdf/D03/d03raf.pdf.

18. **Strang, Glibert and Fix, George J.** *An Analysis Of The Finite Element Method.* Englewood Cliffs, NJ : Prentice-Hall, 1973.

19. **Numerical Algorithms Group.** D06BAF NAG routine document. [Online] http://www.nag.com/numeric/fl/nagdoc_fl23/pdf/D06/d06baf.pdf.

20. —. D06AAF NAG routine document. [Online] http://www.nag.com/numeric/fl/nagdoc_fl23/pdf/D06/d06aaf.pdf.

21. —. D06ABF NAG routine document. [Online] http://www.nag.com/numeric/fl/nagdoc_fl23/pdf/D06/d06abf.pdf.

22. —. D06ACF NAG routine document. [Online] http://www.nag.com/numeric/fl/nagdoc_fl23/pdf/D06/d06acf.pdf.

23. —. D03MAF NAG routine document. [Online] http://www.nag.com/numeric/fl/nagdoc_fl23/pdf/D03/d03maf.pdf.

24. —. F11JCF NAG routine document. [Online]
http://www.nag.com/numeric/fl/nagdoc_fl23/pdf/F11/f11jcf.pdf.

25. —. F11JAF NAG routine document. [Online]
http://www.nag.com/numeric/fl/nagdoc_fl23/pdf/F11/f11jaf.pdf.

26. —. F11ZBF NAG routine document. [Online]
http://www.nag.com/numeric/fl/nagdoc_fl23/pdf/F11/f11zbf.pdf.

27. **Bouhamou, Nadir.** The use of NAG mesh generation and sparse solver routines for solving partial differential equations. [Online] 2001.
http://www.nag.co.uk/doc/techrep/pdf/Tr1_01.pdf.