12

Partial Differential Equation with a Mixed Partial Derivative

This partial differential equation (PDE) application introduces the following mathematical concepts and computational methods:

1. A PDE with an exact solution that can be used to assess the accuracy of a numerical method of lines (MOL) solution.
2. MOL solution of a PDE with a mixed partial derivative.
3. Implicit ordinary differential equations (ODEs) and their origin in a MOL solution.
4. Matrix formulation of implicit ODE solutions.
5. Differential-algebraic (DAE) systems.

PDEs with mixed partial derivatives are commonplace in the physical sciences. Therefore, we consider the computation of numerical solutions to such PDEs within the MOL framework.

We first consider some examples of PDEs with mixed partials ([1], section 3.5). A two-dimensional (2D) example is

$$\frac{\partial^2 u}{\partial x\,\partial y} = f(u)$$

where x and y are boundary-value (spatial) independent variables. This PDE can therefore be considered *elliptic* since it does not have an initial-value variable (see Chapter 10 for a discussion of this geometric classification). Special cases include ([1], p. 268)

$$\frac{\partial^2 u}{\partial x\,\partial y} = a \sinh u$$

the *sinh-Gordon equation*, and

$$\frac{\partial^2 u}{\partial x\,\partial y} = a \sin u$$

the *sine-Gordon equation*.

Another 2D elliptic PDE with a mixed partial derivative is the Monge-Ampère equation, which has applications in differential geometry, gas dynamics, and meteorology ([1], p. 451):

$$\left(\frac{\partial^2 u}{\partial x\,\partial y}\right)^2 - \frac{\partial^2 u}{\partial x^2}\frac{\partial^2 u}{\partial y^2} = f(x, y)$$

An initial-value variable, t, can also be included in the PDE. For example, a nonlinear form of the 1D *Calogero equation* ([1], chapter 7)

$$\frac{\partial^2 u}{\partial x\,\partial t} = u\frac{\partial^2 u}{\partial x^2} + a\left(\frac{\partial u}{\partial x}\right)^2$$

and the 2D *Khokhlov-Zabolotskaya equation*, which describes the propagation of sound in a nonlinear medium

$$\frac{\partial^2 u}{\partial x\,\partial t} = u\frac{\partial^2 u}{\partial x^2} + \left(\frac{\partial u}{\partial x}\right)^2 + \frac{\partial^2 u}{\partial y^2}$$

have t in the mixed partial derivative.

The origin of PDEs with mixed partial derivatives is illustrated in Appendix 1, where an *anisotropic diffusion equation* is derived in three dimensions in *cylindrical* and *spherical* coordinates.

To illustrate the numerical integration of a PDE with a mixed partial derivative, we consider the following 1D PDE:

$$\frac{\partial u}{\partial t} = \frac{\partial^3 u}{\partial x^2\,\partial t} + u$$

or in subscript notation,

$$u_t = u_{xxt} + u \tag{12.1}$$

where

u	dependent variable
x	boundary-value (spatial) independent variable
t	initial-value independent variable

Equation (12.1) is first order in t and second order in x (through the mixed partial $\partial^3 u/\partial x^2 \partial t$). It therefore requires one *initial condition* (IC) and two *boundary conditions* (BCs). The IC is taken as

$$u(x, t = 0) = \sin(\pi x/L) \tag{12.2}$$

and the two BCs as

$$u(x = 0, t) = u(x = L, t) = 0 \qquad (12.3)(12.4)$$

The analytical solution to Eqs. (12.1)–(12.4) is

$$u(x, t) = \sin(\pi x/L)e^{at}; \quad a = \frac{1}{1 - (\pi/L)^2} \tag{12.5}$$

Equations (12.1)–(12.4) constitute the complete PDE problem. The solution to this problem, Eq. (12.5), is used in the subsequent programming and analysis to evaluate the numerical MOL solution.

We now consider some Matlab routines for a numerical MOL solution of Eqs. (12.1)–(12.4) with the analytical solution, Eq. (12.5), included. A main program, `pde_1_main`, is given in Listing 12.1.

```
%
%Clear previous files
  clear all
  clc
%
% Parameters shared with the ODE routine
  global  a     x    xl    xu    dx    cm     n  ncall
%
% Boundaries, number of grid points
  xl=0.0;
  xu=1.0;
  n=49;
%
% Initial condition
  t0=0.0;
  u0=inital_1(t0);
%
% Independent variable for ODE integration
  tf=20.0;
  tout=[t0:4.0:tf]';
  nout=6;
  ncall=0;
%
% Coefficient matrix
  for i=1:n
  for j=1:n
    if(i==j)cm(i,j)=(1.0-2.0/dx^2);
    elseif(abs(i-j)==1)cm(i,j)=1.0/dx^2;
    else cm(i,j)=0.0;
    end
  end
  end
%
% ODE integration
  reltol=1.0e-06; abstol=1.0e-06;
  options=odeset('RelTol',reltol,'AbsTol',abstol);
%
```

```
% Explicit (nonstiff) integration
  [t,u]=ode45(@pde_1,tout,u0,options);
%
% Analytical solution and difference between the numerical and
% analytical solutions at selected points
  for it=1:nout
    fprintf('\n\n   t     x    u(x,t)    u(x,t)       err\n')
    fprintf('                      num       anal          \n')
    for i=1:n
      u_anal(it,i)=ua(t(it),x(i));
      err(it,i)=u(it,i)-u_anal(it,i);
    end
    for i=1:5:n
      fprintf('%6.2f%8.3f%12.6f%12.6f%12.6f\n',...
              t(it),x(i),u(it,i),u_anal(it,i),err(it,i));
    end
  end
  fprintf('\n  ncall = %4d\n\n',ncall);
%
% Plot numerical and analytical solutions
  figure(1)
  x=[0.0 x 1.0];
  for it=1:nout
    uplot(it,1)=0.0;
    uplot(it,2:n+1)=u(it,1:n);
    uplot(it,n+2)=0.0;
    uaplot(it,1)=0.0;
    uaplot(it,2:n+1)=u_anal(it,1:n);
    uaplot(it,n+2)=0.0;
  end
  plot(x,uplot,'o',x,uaplot,'-')
  xlabel('x')
  ylabel('u(x,t)')
  title('Mixed partial PDE; t = 0, 5,..., 20;
      o - numerical; solid - analytical')
```

Listing 12.1. Main program `pde_1_main`

We can note the following points about this program:

1. After specifying some *global* variables, the program sets the parameters for the spatial grid for the interval $0 \leq x \leq 1$ with 49 *interior points* (grid points at `x=xl,xu` are not included in the grid). Note that these parameters are global so that they can be shared with other routines. IC (12.2) is then defined through a call to `inital_1` (discussed subsequently).

```
%
% Clear previous files
  clear all
  clc
%
% Parameters shared with the ODE routine
  global  a     x    xl    xu    dx    cm     n  ncall
%
% Boundaries, number of grid points
  xl=0.0;
  xu=1.0;
  n=49;
%
% Initial condition
  t0=0.0;
  u0=inital_1(t0);
```

2. The interval in t is defined as $0 \leq t \leq 20$ with an output interval of 4 so that the solution will be displayed six times (counting $t = 0$). The counter for the ODE routine, `ncall`, is also initialized.

```
%
% Independent variable for ODE integration
  tf=20.0;
  tout=[t0:4.0:tf]';
  nout=6;
  ncall=0;
```

3. A coefficient matrix, `cm`, (also termed a *coupling matrix* or a *mass matrix*) is defined that originates with the MOL approximation of the mixed partial derivative $\partial^3 u/\partial x^2\,\partial t$ in Eq. (12.1).

```
%
% Coefficient matrix
  for i=1:n
  for j=1:n
    if(i==j)cm(i,j)=(1.0-2.0/dx^2);
    elseif(abs(i-j)==1)cm(i,j)=1.0/dx^2;
    else cm(i,j)=0.0;
    end
  end
  end
```

The origin and use of this matrix is considered after this discussion of the main program in Listing 12.1 is concluded.

4. The MOL ODEs are integrated by ode45, which, although a nonstiff integrator, is quite efficient for this application, as demonstrated by the output discussed subsequently. The ODE routine called by ode45 is pde_1.

```
%
% ODE integration
  reltol=1.0e-06; abstol=1.0e-06;
  options=odeset('RelTol',reltol,'AbsTol',abstol);
%
% Explicit (nonstiff) integration
  [t,u]=ode45(@pde_1,tout,u0,options);
```

5. The analytical solution of Eq. (12.5) is evaluated by ua (discussed subsequently) and the numerical and analytical solutions and their difference are then displayed. The counter ncall for the calls to pde_1 is then displayed.

```
%
% Analytical solution and difference between the numerical and
% analytical solutions at selected points
  for it=1:nout
    fprintf('\n\n   t     x    u(x,t)    u(x,t)       err\n')
    fprintf('                   num      anal          \n')
    for i=1:n
      u_anal(it,i)=ua(t(it),x(i));
      err(it,i)=u(it,i)-u_anal(it,i);
    end
    for i=1:5:n
      fprintf('%6.2f%8.3f%12.6f%12.6f%12.6f\n', ...
              t(it),x(i),u(it,i),u_anal(it,i),err(it,i));
    end
  end
  fprintf('\n  ncall = %4d\n\n',ncall);
```

6. Finally, the numerical and analytical solutions are plotted together so that they may be compared. The for loop merely fills in BCs (12.3) and (12.4) (at x=xl,xu) since the spatial grid does not include these boundary points; this is done to improve slightly the appearance of the plotted output.

```
%
% Plot numerical and analytical solutions
  figure(1)
  x=[0.0 x 1.0];
```

```
for it=1:nout
  uplot(it,1)=0.0;
  uplot(it,2:n+1)=u(it,1:n);
  uplot(it,n+2)=0.0;
  uaplot(it,1)=0.0;
  uaplot(it,2:n+1)=u_anal(it,1:n);
  uaplot(it,n+2)=0.0;
end
plot(x,uplot,'o',x,uaplot,'-')
xlabel('x')
ylabel('u(x,t)')
title('Mixed partial PDE; t = 0, 5,..., 20;
     o - numerical; solid - analytical')
```

The subordinate routines `inital_1`, `ua`, `pde_1` called by `ode45` and the output from the main program `pde_1_main` are discussed after the following development of the MOL analysis of Eqs. (12.1)–(12.4). Equation (12.1) can be approximated as a system of ODEs if the x differentiation is approximated with the three-point finite difference (FD)

$$u_{xx} \approx \frac{u((i+1)\Delta x) - 2u(\Delta x i) + u((i-1)\Delta x)}{\Delta x^2}, \quad \Delta x = L/(n+1), \; i = 1, 2, \ldots, n \tag{12.6}$$

where $u(0,t) = u((n+1)\Delta x, t) = 0$ from BCs (12.3) and (12.4). Note, again, that the index i spans the *interior* points of the grid and does not include the boundary points at $x = 0, L$.

Application of Eq. (12.6) to Eq. (12.1) gives

$$u_t(i\Delta x, t) = -\frac{u_t((i+1)\Delta x, t) - 2u_t(i\Delta x, t) + u_t((i-1)\Delta x, t)}{\Delta x^2} + u(i,t) \tag{12.7}$$

The distinguishing feature of Eqs. (12.7) is that the n ODEs are coupled through the derivatives in t. In other words, *more than one derivative in t appears in each ODE* and so it is not possible to solve explicitly for individual t derivatives using each equation one at a time. Rather we must *solve the entire ODE system simultaneously* for the three t derivatives in each ODE – a total of n derivatives in t.

Since the t derivatives in Eqs. (12.7) appear implicitly, ODEs of the form of Eqs. (12.7) are termed *implicit* ODEs (in contrast to ODE systems discussed in earlier chapters in which the ODE t derivatives appeared explicitly, i.e., individually, or only one derivative in each ODE, and which are therefore termed *explicit* ODEs). This classification of ODEs should not be confused with explicit and implicit integration algorithms, which relates to a different issue, namely nonstiff and stiff ODEs.

Another important issue is that we have so far used the Matlab integrators for explicit ODE systems. Although some of the Matlab integrators will accept implicit ODEs, we will not address this feature directly, but rather we will still call the Matlab integrators, in the present case `ode45`, as we did previously, but in the ODE routine, for example, pde_1, we will convert the implicit ODEs to the explicit form. The details are given in the discussion of pde_1 to follow, and involve the coefficient matrix cm defined in pde_1_main.

Some additional terminology can be explained at this point. The implicit ODEs of Eqs. (12.7) are coupled in the sense that any given t derivative cannot be evaluated without solving for all n t derivatives. Thus, the ODEs are simultaneous or *coupled*. Further, in the case of Eqs. (12.7), the derivatives are to the first power, so Eqs. (12.7) are termed *linearly implicit* ODEs (which are relatively easy to solve in contrast to ODEs in which the t derivatives appear nonlinearly – in that case, a nonlinear algebraic solver, usually based on a variant of Newton's method, must also be used). The coefficient matrix that couples the ODEs of Eqs. (12.7), cm, is therefore also called a *coupling matrix* or possibly a *mass matrix* reflecting the origin of implicit ODEs from application to mechanical systems with discrete masses.

The uncoupling of the simultaneous ODEs of Eqs. (12.7) can be accomplished by the use of some standard matrix methods of linear algebra. First, Eqs. (12.7) can be written as

$$\begin{aligned}(1/\Delta x^2)u_t((i-1)\Delta x,t) + (1-2/\Delta x^2)u_t(i\Delta x,t) + (1/\Delta x^2)u_t((i+1)\Delta x,t)\\ = u(i\Delta x,t) \qquad (12.8)\end{aligned}$$

or in matrix form,

$$\begin{bmatrix} (1-\frac{2}{\Delta x^2}) & \frac{1}{\Delta x^2} & & & \\ \frac{1}{\Delta x^2} & (1-\frac{2}{\Delta x^2}) & \frac{1}{\Delta x^2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{\Delta x^2} & (1-\frac{2}{\Delta x^2}) & \frac{1}{\Delta x^2} \\ & & & \frac{1}{\Delta x^2} & (1-\frac{2}{\Delta x^2}) \end{bmatrix} \begin{bmatrix} u_t(\Delta x,t) \\ u_t(2\Delta x,t) \\ \vdots \\ u_t((n-1)\Delta x,t) \\ u_t(n\Delta x,t) \end{bmatrix}$$

$$= \begin{bmatrix} u(\Delta x,t) \\ u(2\Delta x,t) \\ \vdots \\ u((n-1)\Delta x,t) \\ u(n\Delta x,t) \end{bmatrix} \qquad (12.9)$$

We can then solve Eq. (12.9) for the vector of derivatives in t

$$\begin{bmatrix} u_t(\Delta x,t) \\ u_t(2\Delta x,t) \\ \vdots \\ u_t((n-1)\Delta x,t) \\ u_t(n\Delta x,t) \end{bmatrix}$$

$$= \begin{bmatrix} (1-\frac{2}{\Delta x^2}) & \frac{1}{\Delta x^2} & & & \\ \frac{1}{\Delta x^2} & (1-\frac{2}{\Delta x^2}) & \frac{1}{\Delta x^2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{\Delta x^2} & (1-\frac{2}{\Delta x^2}) & \frac{1}{\Delta x^2} \\ & & & \frac{1}{\Delta x^2} & (1-\frac{2}{\Delta x^2}) \end{bmatrix}^{-1} \begin{bmatrix} u(\Delta x,t) \\ u(2\Delta x,t) \\ \vdots \\ u((n-1)\Delta x,t) \\ u(n\Delta x,t) \end{bmatrix} \tag{12.10}$$

Equation (12.10) is programmed in the ODE routine pde_1 through the use of the Matlab matrix inverse operator (as applied to the RHS of Eq. (12.10)). Once the vector of (explicit) derivatives $u_t(1\Delta x,t), u_t(2\Delta x,t),\ldots,u_t(n\Delta x,t)$ is computed (from Eq. (12.10)), it can be sent to a Matlab integrator such as ode45 in the usual fashion.

To illustrate how this is done, we now consider pde_1, as given in Listing 12.2.

```
  function ut=pde_1(t,u)
%
% Function yt computes the t derivative vector of the PDE with
% a mixed partial derivative
%
  global  a     x    xl    xu    dx    cm     n  ncall
%
% PDE
  ut=cm\u;
%
% Increment calls to pde_1
  ncall=ncall+1;
```

Listing 12.2. MOL ODE routine pde_1 called by main program pde_1_main

After the function is defined and some parameters and variables are declared global, Eq. (12.10) is programmed as ut=cm\u; (which perhaps qualifies as some of the most compact coding of a PDE to be found anywhere!). Note that the Matlab inverse operator \ is used in conjunction with Eq. (12.10) (you might check that the coefficient matrix of Eq. (12.10) is in fact cm programmed in pde_1_main of Listing 12.1). Even the usual transpose is not required since \ returns a column vector, as required by ode45. cm was not evaluated in pde_1 since it is a constant matrix and is therefore evaluated just once in the main program and passed as a global variable to pde_1. If the elements of this coupling matrix were functions of the dependent variable (so that the ODE system would be nonlinear), cm would then be evaluated in pde_1.

The initialization routine, inital_1, that implements Eq. (12.2) is given in Listing 12.3.

```
  function u0=inital_1(t0)
%
% Function inital_1 sets the initial condition for the PDE
% with a mixed partial derivative
%
  global  a    x    xl    xu    dx    cm    n
%
% Grid spacing, constant a
  dx=(xu-xl)/(n+1);
  a=1.0/(1.0-(pi/(xu-xl))^2);
%
% IC over the spatial grid
  for i=1:n
%
%   Uniform grid
    x(i)=xl+i*dx;
%
%   Initial condition
    u0(i)=sin(pi*x(i)/(xu-xl));
  end
```

Listing 12.3. Initial condition routine inital_1

We can note the following points about inital_1:

1. After the function is defined and certain parameters and variables are declared global, the grid spacing dx is computed based on the n interior points (and thus the use of $n + 1$ in computing the grid spacing dx). Also, the constant a of Eq. (12.5) is computed.

```
  function u0=inital_1(t0)
%
% Function inital_1 sets the initial condition for the PDE
% with a mixed partial derivative
%
  global  a    x    xl    xu    dx    cm    n
%
% Grid spacing, constant a
  dx=(xu-xl)/(n+1);
  a=1.0/(1.0-(pi/(xu-xl))^2);
```

2. IC (2) is then implemented in a `for` loop.

```
%
% IC over the spatial grid
  for i=1:n
%
%   Uniform grid
    x(i)=xl+i*dx;
%
%   Initial condition
    u0(i)=sin(pi*x(i)/(xu-xl));
  end
```

Finally, `ua` is a straightforward coding of the analytical solution of Eq. (12.5).

```
  function uanal=ua(t,x)
%
% Function ua computes the analytical solution to the PDE with
% a mixed partial derivative
%
  global  a    xl    xu
%
% Analytical solution
  uanal=sin(pi*x/(xu-xl))*exp(a*t);
```

This completes the Matlab coding of Eqs. (12.1)–(12.4). We now consider the output from `pde_1_main`. Selected numerical output is given in Table 12.1. Agreement between the numerical and analytical solution is to three figures (with only 67 calls to `pde_1`) so that the three-point FDs of Eq. (12.6) provide satisfactory accuracy. Clearly, the 49 ODEs are not stiff (i.e., `ncall=67`).

The plotted output from `pde_1` is shown in Figure 12.1. The maximum peak corresponds to $t = 0$ and decays according to Eq. (12.5). The long-term solution is $u(x, t = \infty) = 0$ (provided $a < 0$).

We conclude this chapter with the following observations:

1. Mixed partial derivatives like that of Eq. (12.1) (a combination of an initial-value independent variable such as t and a boundary-value independent variable such as x) appear in PDEs that model a spectrum of physical systems.
2. When there is an initial-value independent variable in the mixed partial, the methods presented in this chapter can generally be applied, including the use of an initial-value ODE integrator such as `ode45`.

Table 12.1. Partial output from `pde_1_main` and `pde_1`

```
    t        x        u(x,t)       u(x,t)          err
                       num          anal
  0.00    0.020    0.062791     0.062791     0.000000
  0.00    0.120    0.368125     0.368125     0.000000
  0.00    0.220    0.637424     0.637424     0.000000
  0.00    0.320    0.844328     0.844328     0.000000
  0.00    0.420    0.968583     0.968583     0.000000
  0.00    0.520    0.998027     0.998027     0.000000
  0.00    0.620    0.929776     0.929776     0.000000
  0.00    0.720    0.770513     0.770513     0.000000
  0.00    0.820    0.535827     0.535827     0.000000
  0.00    0.920    0.248690     0.248690     0.000000


    t        x        u(x,t)       u(x,t)          err
                       num          anal
  4.00    0.020    0.039991     0.039998    -0.000007
  4.00    0.120    0.234458     0.234497    -0.000039
  4.00    0.220    0.405975     0.406042    -0.000067
  4.00    0.320    0.537752     0.537841    -0.000089
  4.00    0.420    0.616890     0.616992    -0.000102
  4.00    0.520    0.635643     0.635748    -0.000105
  4.00    0.620    0.592174     0.592272    -0.000098
  4.00    0.720    0.490739     0.490820    -0.000081
  4.00    0.820    0.341268     0.341324    -0.000056
  4.00    0.920    0.158390     0.158417    -0.000026
                 .                       .
                 .                       .
                 .                       .

     Output for t = 8, 12, 16 has been removed
                 .                       .
                 .                       .
                 .                       .

    t        x        u(x,t)       u(x,t)          err
                       num          anal
 20.00    0.020    0.006580     0.006586    -0.000005
 20.00    0.120    0.038579     0.038611    -0.000032
 20.00    0.220    0.066801     0.066856    -0.000055
 20.00    0.320    0.088484     0.088557    -0.000073
 20.00    0.420    0.101506     0.101590    -0.000084
 20.00    0.520    0.104592     0.104678    -0.000086
 20.00    0.620    0.097439     0.097519    -0.000080
 20.00    0.720    0.080749     0.080815    -0.000067
 20.00    0.820    0.056154     0.056200    -0.000046
 20.00    0.920    0.026062     0.026084    -0.000022

  ncall =    67
```

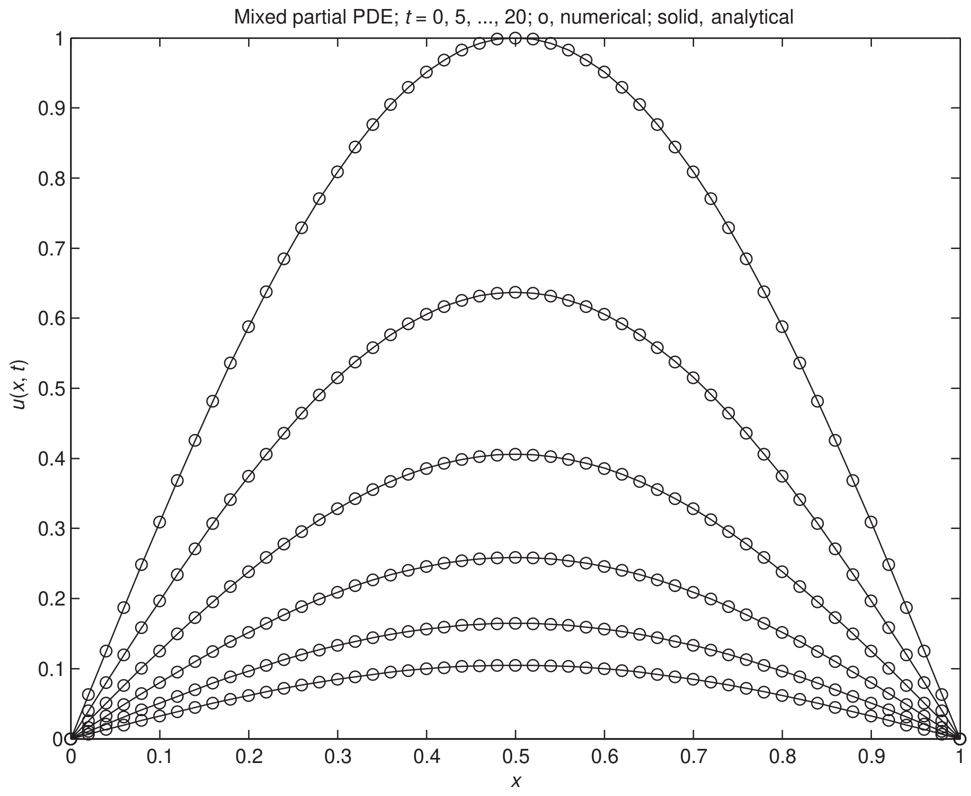

Figure 12.1. Output of main program `pde_1_main`

3. A more difficult problem results from a mixed partial that involves two or more boundary-value independent variables. For example, a PDE in two dimensions, with spatial variables x and y, could have a mixed partial $\partial^2 u/\partial x\, \partial y$. This mixed partial could be approximated by FDs. One approach would be to use *stagewise differentiation*; that is, differentiate u with respect to x to produce $\partial u/\partial x$ and then differentiate this derivative with respect to y to produce $\partial^2 u/\partial x\, \partial y$. The difficulty with this type of mixed partial originates with the stability of the PDE solution, but we will not pursue this matter further.
4. The FD approximation of the PDE with a mixed partial derivative such as Eq. (12.1) generally leads to implicitly coupled ODEs such as Eqs. (12.8). However, coupled ODEs can arise in a variety of applications, particularly in finite element analysis. Again, we will not go into this matter here other than to point out that implicitly coupled ODEs can be handled with the matrix methods illustrated previously or by using ODE integrators particularly designed for this type of ODE system; in fact, some of the Matlab integrators have this feature.
5. The matrix approach considered here in which the ODEs were uncoupled in `pde_1` through the use of the linear algebraic operator (solver) \ is based on full matrix methods. Clearly, however, the coupling matrix `cm` defined numerically in `pde_1_main` is banded (it is tridiagonal from Eq. (12.6)), which is typical in physical applications. Thus, the use of full-matrix linear algebra

can be inefficient when applied to banded matrix systems, particularly as the number of ODEs increases. ($n = 49$ is a very modest problem, so the computational efficiency was quite acceptable.)

6. Another type of differential equation formulation was introduced through BCs (12.3) and (12.4). Specifically, the algebraic equation $u(x = 0, t) = u(x = L, t) = 0$ was added to the ODE system of Eqs. (12.7) (you might check how these BCs were included in Eqs. (12.7)). In other words, algebraic equations were included at the external points $x = 0, x = L$ of the grid, which is a common occurrence in MOL analysis to produce a DAE system.

   However, algebraic equations can also occur at the interior points of a spatial grid. For example, for a reaction–diffusion system, the reactions may be more rapid than the diffusion. The reactions can therefore be considered at equilibrium, so that the equations describing the reactions are algebraic, while the equations describing the diffusion are ODEs (through the MOL analysis of the PDEs). We will not consider DAE systems further here other than to mention that algorithms and associated computer codes for DAE systems are available. Also, depending on a measure of difficulty of a DAE system, the so-called *index*, an algebraic equation solver can possibly be added at the beginning of the MOL ODE routine to solve the algebraic equations first, followed by the coding of the derivative vector of the ODEs.

7. a in Eq. (12.5) can be positive (if $L > \pi$), in which case, the solution of Eq. (12.1) would increase exponentially in t (according to Eq. (12.5)). Thus, this problem can be stable or unstable in its long-time behavior (and this is a property of the PDE problem, not the numerical method used to solve the problem).

   Finally, to further elucidate the numerical solution, we also include a 3D plot produced with the following code:

```
%
% 3D Plot
  figure(2)
  surfl(x,t,uplot, 'light'); shading interp
  axis tight
  title('Mixed partials equation');
  set(get(gca,'XLabel'),'String','space, x')
  set(get(gca,'YLabel'),'String','time, t')
  set(get(gca,'ZLabel'),'String','u(x,t)')
  set(gca,'XLim',[xl xu],'YLim',[t0 tf],'ZLim', [0 1]);
  view(60,40);
  colormap('cool');
  print -dpng -r300 fig2.png;
```

The resulting 3D plot is shown in Figure 12.2.

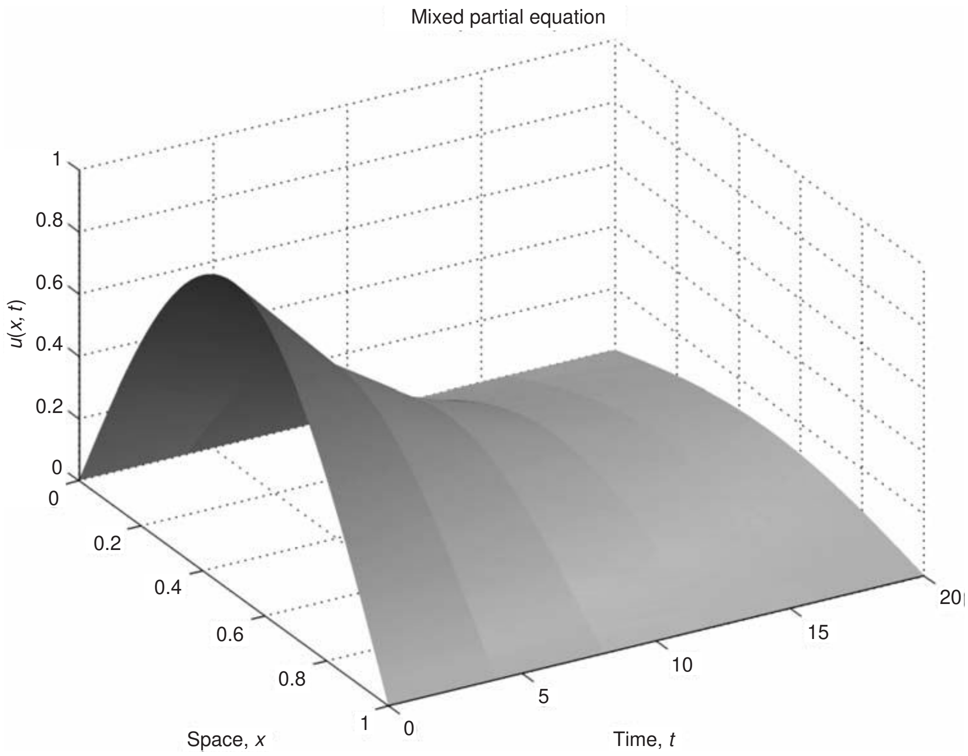

Figure 12.2. Three-dimensional output from main program `pde_1_main`

REFERENCE

[1] Polyanin, A. D. and V. F. Zaitsev (2004), *Handbook of Nonlinear Partial Differential Equations*, Chapman and Hall/CRC, Boca Raton, FL