6

The Cubic Schrödinger Equation

The following partial differential equation (PDE) problem introduces the following mathematical concepts and computational methods:

1. A nonlinear PDE with an exact solution that can be used to assess the accuracy of a numerical method of lines (MOL) solution.
2. A complex PDE. Specifically, the complex PDE is expressed as two real PDEs, so this problem also illustrates the numerical integration of simultaneous PDEs.
3. Some of the basic properties of *traveling waves* that have broad application in many areas of physics, engineering, and the biological sciences.
4. Illustrative programming of a traveling wave solution.
5. Sparse matrix integration of the ordinary differential equations (ODEs) for the MOL approximation of PDEs.
6. Computer analysis of simultaneous PDEs over an essentially infinite spatial domain.

The *one-dimensional* (1D) *cubic Schrödinger equation* (CSE) *equation* is [1]

$$i\frac{\partial u}{\partial t} + \frac{\partial^2 u}{\partial x^2} + q|u|^2 u = 0$$

or in subscript notation,

$$iu_t + u_{xx} + q|u|^2 u = 0 \tag{6.1}$$

where

u complex dependent variable
x boundary-value (spatial) independent variable
t initial-value independent variable
q arbitrary parameter (constant)
i $\sqrt{-1}$

The nonlinear term $q|u|^2u = 0$ is the origin of the term "cubic" in the name *cubic Schrödinger equation* ($|u|$ denotes the absolute value of the complex variable u).

Some additional background information to the Schrödinger equation is given in Appendix A. As we will observe, including this nonlinearity in the MOL solution of Eq. (6.1) is quite straightforward.

Equation (6.1) is first order in t and second order in x. It therefore requires one *initial condition* (IC) and two *boundary conditions* (BCs). The IC is taken from the analytical solution (with $q = 1$) [1]

$$u(x,t) = \sqrt{2}\, e^{i(0.5x+0.75t)}\, \mathrm{sech}(x-t) \tag{6.2a}$$

Since $u(x,t)$ is complex, we analyze it as $u(x,t) = v(x,t) + iw(x,t)$, where $v(x,t)$ and $w(x,t)$ are two real functions that are to be computed numerically.

It then follows from Eq. (6.2a) that

$$|u(x,t)| = \sqrt{2}\, \mathrm{sech}(x-t) \tag{6.2b}$$

Equation (6.2b) will be used subsequently for comparison with the numerical solution.

Then with $t = 0$ in Eq. (6.2a), the IC is

$$u(x,t=0) = \sqrt{2}\, e^{i(0.5x)}\, \mathrm{sech}(x) \tag{6.3}$$

To complete the specification of the problem, we would naturally consider the two required BCs for Eq. (6.1). However, if the PDE is analyzed over an essentially infinite domain, $-\infty \le x \le \infty$, and if changes in the solution occur only over a finite interval in x, then *BCs at infinity have no effect*; in other words, we do not have to actually specify BCs (since they have no effect). This situation will be clarified through the PDE solution.

Consequently, Eqs. (6.1) and (6.3) constitute the complete PDE problem. The solution to this problem, Eqs. (6.2a) and (6.2b), is used in the subsequent programming and analysis to evaluate the numerical MOL solution. Note that Eqs. (6.2a) and (6.2b) are a *traveling wave* solution since the argument of the sech function is $x - t$.

We now consider some Matlab routines for a numerical MOL solution of Eqs. (6.1) and (6.3) with the analytical solution, Eqs. (6.2a) and (6.2b), included. A main program, `pde_1_main`, is given in Listing 6.1.

```
%
% Clear previous files
  clear all
  clc %
%
% Parameters shared with the ODE routine
  global  ncall     x     xl     xu      n
%
% Boundaries, number of grid points
  xl=-10.0;
  xu= 40.0;
  n=251;
%
% Initial condition
```

```
  t0=0.0;
  u0=inital_1(t0);
%
% Independent variable for ODE integration
  tf=30.0;
  tout=[t0:5.0:tf]';
  nout=7;
  ncall=0;
%
% ODE integration
  mf=1;
  reltol=1.0e-06; abstol=1.0e-06;
  options=odeset('RelTol',reltol,'AbsTol',abstol);
%
% Explicit (nonstiff) integration
  if(mf==1)[t,u]=ode45(@pde_1,tout,u0,options); end
%
% Implicit (sparse stiff) integration
  if(mf==2)
    S=jpattern_num;
    pause
    options=odeset(options,'JPattern',S)
    [t,u]=ode15s(@pde_1,tout,u0,options);
  end
%
% One vector to two vectors
  for it=1:nout
  for i=1:n
    v(it,i)=u(it,i);
    w(it,i)=u(it,i+n);
  end
  end
%
% Analytical solution and difference between the numerical and
% analytical solutions
  for it=1:nout
    fprintf('     t     x  x - t      v^2+w^2      v^2+w^2
                  err\n');
    fprintf('                              num       anal
              \n')
    for i=1:n
      v2w2_anal(it,i)=ua(t(it),x(i));
      v2w2(it,i)=u(it,i)^2+u(it,i+n)^2;
      err(it,i)=v2w2(it,i)-v2w2_anal(it,i);
%
%     Output in the neighborhood of the soliton peak
      if(abs(x(i)-t(it))<=1.0)
```

```
        fprintf('%6.2f%8.1f%8.2f%15.6f%15.6f%15.6f\n', ...
                t(it),x(i),x(i)-t(it),v2w2(it,i), ...
                v2w2_anal(it,i),err(it,i));
      end
    end
%
%   Calculate and display three invariants
    ui=u(it,:);
    uint=simp(xl,xu,n,ui);
    fprintf('\n Invariants at t = %5.2f',t(it));
    fprintf('\n    I1 = %10.4f',    uint(1));
    fprintf('\n    I2 = %10.4f',    uint(2));
    fprintf('\n    I3 = %10.4f\n\n',uint(3));
    fprintf('\n');
  end
  fprintf('  ncall = %4d\n\n',ncall);
%
% Plot numerical and analytical solutions
  figure(1)
  plot(x,v2w2,'-',x,v2w2_anal,'o')
  xlabel('x')
  ylabel('u(x,t)')
  title('CSE; t = 0, 5,..., 30; o - numerical;
         solid - analytical')
  print -deps -r300 pde.eps; print -dps -r300 pde.ps; ...
  print -dpng -r300 pde.png
```

Listing 6.1. Main program `pde_1_main`

We can note the following points about this program:

1. After specifying some *global* variables, the spatial domain is defined as $-10 \leq x \leq 40$, and the MOL grid has 251 points.

```
%
% Clear previous files
  clear all
  clc
%
% Parameters shared with the ODE routine
  global  ncall     x     xl     xu      n
%
% Boundaries, number of grid points
  xl=-10.0;
  xu= 40.0;
  n=251;
```

Since $u(x,t) = v(x,t) + iw(x,t)$, pde_1_main of Listing 6.1 will actually integrate $2 \times 251 = 502$ ODEs (251 for $v(x,t)$ and 251 for $w(x,t)$).

2. Routine inital_1 (discussed subsequently) is called to define IC (6.3) over $-10 \le x \le 40$. The t interval is then defined as $0 \le t \le 30$ with solution outputs at $t = 0, 5, \ldots, 30$.

```
%
% Initial condition
  t0=0.0;
  u0=inital_1(t0);
%
% Independent variable for ODE integration
  tf=30.0;
  tout=[t0:5.0:tf]';
  nout=7;
  ncall=0;
```

3. The $n = 2(251)$ ODEs are integrated by a call to either a nonstiff integrator ode45 (for mf=1) or a stiff integrator ode15s (for mf=2). For this problem, ode45 requires more calls to the ODE routine pde_1 than ode15s, which is not unusual. However, the additional computation required by ode15s, that is, the sparse matrix integrator, requires additional time, so the two integrators are not very different in their overall computational efficiency. In other words, a stiff integrator *does not always lead to greater computational efficiency* (depending on the characteristics of the problem). In the subsequent discussion, we consider the output from ode45.

```
%
% ODE integration
  mf=1;
  reltol=1.0e-06; abstol=1.0e-06;
  options=odeset('RelTol',reltol,'AbsTol',abstol);
%
% Explicit (nonstiff) integration
  if(mf==1)[t,u]=ode45(@pde_1,tout,u0,options); end
%
% Implicit (sparse stiff) integration
  if(mf==2)
    S=jpattern_num;
    pause
    options=odeset(options,'JPattern',S)
    [t,u]=ode15s(@pde_1,tout,u0,options);
  end
```

4. Since the Matlab integrators such as `ode45` integrate a single vector of ODEs, the solution vector `u` returned from `ode45` is then divided into two parts, $v(x, t)$ and $w(x, t)$, of $u(x, t) = v(x, t) + iw(x, t)$.

```
%
% One vector to two vectors
  for it=1:nout
  for i=1:n
    v(it,i)=u(it,i);
    w(it,i)=u(it,i+n);
  end
  end
```

5. The analytical solution of Eq. (6.2b) is then computed by function `ua` (discussed subsequently) and the difference between the analytical and numerical solutions is computed. Selected portions of the analytical and numerical solutions and their differences are then displayed.

```
%
% Analytical solution and difference between the numerical and
% analytical solutions
  for it=1:nout
    fprintf('   t     x    x - t       v^2+w^2       v^2+w^2
                err\n');
    fprintf('                                num          anal
                 \n')
    for i=1:n
      v2w2_anal(it,i)=ua(t(it),x(i));
      v2w2(it,i)=u(it,i)^2+u(it,i+n)^2;
      err(it,i)=v2w2(it,i)-v2w2_anal(it,i);
%
%     Output in the neighborhood of the soliton peak
      if(abs(x(i)-t(it))<=1.0)
        fprintf('%6.2f%8.1f%8.2f%15.6f%15.6f%15.6f\n', ...
                t(it),x(i),x(i)-t(it),v2w2(it,i), ...
                v2w2_anal(it,i),err(it,i));
      end
    end
```

Since `v` and `w` are computed over 251 points in x, the output when fully displayed would be quite lengthy. We therefore display the solution only in the neighborhood of the peak of the solution. This is accomplished by monitoring the argument $x - t$ in Eqs. (6.2a) and (6.2b). At $x = t$, the solution has a peak

(maximum) value, and drops off in the neighborhood of this peak, which in the preceding code is the interval defined by $|x - t| \leq 1$. The traveling wave of Eqs. (6.2a) and (6.2b) is clear from the numerical output discussed subsequently.

6. Three invariants are evaluated by a call to `simp` that implements a numerical quadrature (integration) based on Simpson's rule (discussed subsequently).

```
%
%   Calculate and display three invariants
    ui=u(it,:);
    uint=simp(xl,xu,n,ui);
    fprintf('\n Invariants at t = %5.2f',t(it));
    fprintf('\n    I1 = %10.4f',    uint(1));
    fprintf('\n    I2 = %10.4f',    uint(2));
    fprintf('\n    I3 = %10.4f\n\n',uint(3));
    fprintf('\n');
  end
  fprintf('  ncall = %4d\n\n',ncall);
```

The integral invariants for the single soliton are [1]

$$u_1(t) = \int_{-\infty}^{\infty} |u(x,t)|^2 \, dx \tag{6.4a}$$

$$\begin{aligned} u_2(t) &= \int_{-\infty}^{\infty} i(u\bar{u}_x - \bar{u}u_x)dx = \int_{-\infty}^{\infty} i[(v+iw)(v_x - iw_x) - (v-iw)(v_x + iw_x)]dx \\ &= \int_{-\infty}^{\infty} i[(vv_x + iwv_x - ivw_x + ww_x) - (vv_x - iwv_x + ivw_x + ww_x)]dx \\ &= 2\int_{-\infty}^{\infty} (vw_x - wv_x)dx \end{aligned} \tag{6.4b}$$

$$u_3(t) = \int_{-\infty}^{\infty} [|u_x|^2 - \frac{1}{2}q|u|^4]dx \tag{6.4c}$$

The computed values of these invariants are discussed subsequently.

7. The numerical and analytical solutions are then plotted.

```
%
% Plot numerical and analytical solutions
  figure(1)
  plot(x,v2w2,'-',x,v2w2_anal,'o')
  xlabel('x')
  ylabel('u(x,t)')
  title('CSE; t = 0, 5,..., 30; o - numerical;
```

```
        solid - analytical')
    print -deps -r300 pde.eps; print -dps -r300 pde.ps;
    print -dpng -r300 pde.png
```

The selected numerical output and the plotted output are given in Table 6.1 and Figure 6.1.

Table 6.1. Partial output from `pde_1_main` and `pde_1`

```
    t       x    x - t        v^2+w^2        v^2+w^2          err
                                 num            anal
 0.00    -1.0   -1.00       0.839949       0.839949       0.000000
 0.00    -0.8   -0.80       1.118110       1.118110       0.000000
 0.00    -0.6   -0.60       1.423156       1.423156       0.000000
 0.00    -0.4   -0.40       1.711278       1.711278      -0.000000
 0.00    -0.2   -0.20       1.922086       1.922086       0.000000
 0.00     0.0    0.00       2.000000       2.000000       0.000000
 0.00     0.2    0.20       1.922086       1.922086       0.000000
 0.00     0.4    0.40       1.711278       1.711278       0.000000
 0.00     0.6    0.60       1.423156       1.423156       0.000000
 0.00     0.8    0.80       1.118110       1.118110       0.000000
 0.00     1.0    1.00       0.839949       0.839949       0.000000

Invariants at t =  0.00
   I1 =     6.7303
   I2 =     1.9997
   I3 =   -13.8880

    t       x    x - t        v^2+w^2        v^2+w^2          err
                                 num            anal
 5.00     4.0   -1.00       0.840692       0.839949       0.000743
 5.00     4.2   -0.80       1.119083       1.118110       0.000973
 5.00     4.4   -0.60       1.424426       1.423156       0.001271
 5.00     4.6   -0.40       1.712740       1.711278       0.001463
 5.00     4.8   -0.20       1.923553       1.922086       0.001467
 5.00     5.0    0.00       2.000991       2.000000       0.000991
 5.00     5.2    0.20       1.922272       1.922086       0.000186
 5.00     5.4    0.40       1.710552       1.711278      -0.000725
 5.00     5.6    0.60       1.421850       1.423156      -0.001305
 5.00     5.8    0.80       1.116591       1.118110      -0.001520
 5.00     6.0    1.00       0.838547       0.839949      -0.001402
Invariants at t =  5.00
   I1 =     6.7252
   I2 =     1.9997
   I3 =   -13.8635
```

(*continued*)

Table 6.1 (*continued*)

```
    t       x    x - t        v^2+w^2        v^2+w^2          err
                                 num            anal
10.00     9.0   -1.00       0.841776       0.839949       0.001827
10.00     9.2   -0.80       1.120379       1.118110       0.002269
10.00     9.4   -0.60       1.425802       1.423156       0.002647
10.00     9.6   -0.40       1.714002       1.711278       0.002724
10.00     9.8   -0.20       1.924345       1.922086       0.002260
10.00    10.0    0.00       2.001183       2.000000       0.001183
10.00    10.2    0.20       1.921711       1.922086      -0.000375
10.00    10.4    0.40       1.709649       1.711278      -0.001628
10.00    10.6    0.60       1.420661       1.423156      -0.002494
10.00    10.8    0.80       1.115437       1.118110      -0.002674
10.00    11.0    1.00       0.837441       0.839949      -0.002508

Invariants at t = 10.00
   I1 =     6.7089
   I2 =     1.9997
   I3 =   -13.7750
    t       x    x - t        v^2+w^2        v^2+w^2          err
                                 num            anal
15.00    14.0   -1.00       0.842866       0.839949       0.002917
15.00    14.2   -0.80       1.121625       1.118110       0.003515
15.00    14.4   -0.60       1.427007       1.423156       0.003851
15.00    14.6   -0.40       1.715080       1.711278       0.003803
15.00    14.8   -0.20       1.924948       1.922086       0.002862
15.00    15.0    0.00       2.001288       2.000000       0.001288
15.00    15.2    0.20       1.921148       1.922086      -0.000938
15.00    15.4    0.40       1.708711       1.711278      -0.002567
15.00    15.6    0.60       1.419372       1.423156      -0.003784
15.00    15.8    0.80       1.114223       1.118110      -0.003888
15.00    16.0    1.00       0.836325       0.839949      -0.003624

Invariants at t = 15.00
   I1 =     6.6814
   I2 =     1.9997
   I3 =   -13.6262

    t       x    x - t        v^2+w^2        v^2+w^2          err
                                 num            anal
20.00    19.0   -1.00       0.843951       0.839949       0.004002
20.00    19.2   -0.80       1.123048       1.118110       0.004938
20.00    19.4   -0.60       1.428368       1.423156       0.005213
20.00    19.6   -0.40       1.716169       1.711278       0.004892
20.00    19.8   -0.20       1.925484       1.922086       0.003398
20.00    20.0    0.00       2.000912       2.000000       0.000912
20.00    20.2    0.20       1.920351       1.922086      -0.001735
20.00    20.4    0.40       1.707293       1.711278      -0.003985
20.00    20.6    0.60       1.418163       1.423156      -0.004992
```

```
20.00    20.8    0.80       1.112913       1.118110      -0.005197
20.00    21.0    1.00       0.835423       0.839949      -0.004525

Invariants at t = 20.00
   I1 =     6.6427
   I2 =     1.9997
   I3 =   -13.4157

    t        x    x - t        v^2+w^2        v^2+w^2         err
                                  num            anal
25.00    24.0   -1.00       0.844906       0.839949       0.004957
25.00    24.2   -0.80       1.124041       1.118110       0.005931
25.00    24.4   -0.60       1.429421       1.423156       0.006265
25.00    24.6   -0.40       1.717158       1.711278       0.005881
25.00    24.8   -0.20       1.926096       1.922086       0.004010
25.00    25.0    0.00       2.001210       2.000000       0.001210
25.00    25.2    0.20       1.919751       1.922086      -0.002335
25.00    25.4    0.40       1.706370       1.711278      -0.004907
25.00    25.6    0.60       1.416704       1.423156      -0.006451
25.00    25.8    0.80       1.111702       1.118110      -0.006408
25.00    26.0    1.00       0.834255       0.839949      -0.005694

Invariants at t = 25.00
   I1 =     6.5935
   I2 =     1.9997
   I3 =   -13.1521

    t        x    x - t        v^2+w^2        v^2+w^2         err
                                  num            anal
30.00    29.0   -1.00       0.846284       0.839949       0.006335
30.00    29.2   -0.80       1.125482       1.118110       0.007371
30.00    29.4   -0.60       1.431189       1.423156       0.008033
30.00    29.6   -0.40       1.718323       1.711278       0.007046
30.00    29.8   -0.20       1.926872       1.922086       0.004786
30.00    30.0    0.00       2.000883       2.000000       0.000883
30.00    30.2    0.20       1.919042       1.922086      -0.003044
30.00    30.4    0.40       1.704914       1.711278      -0.006364
30.00    30.6    0.60       1.415336       1.423156      -0.007820
30.00    30.8    0.80       1.110307       1.118110      -0.007803
30.00    31.0    1.00       0.833140       0.839949      -0.006808

Invariants at t = 30.00
   I1 =     6.5337
   I2 =     1.9997
   I3 =   -12.8329

 ncall = 18523
```

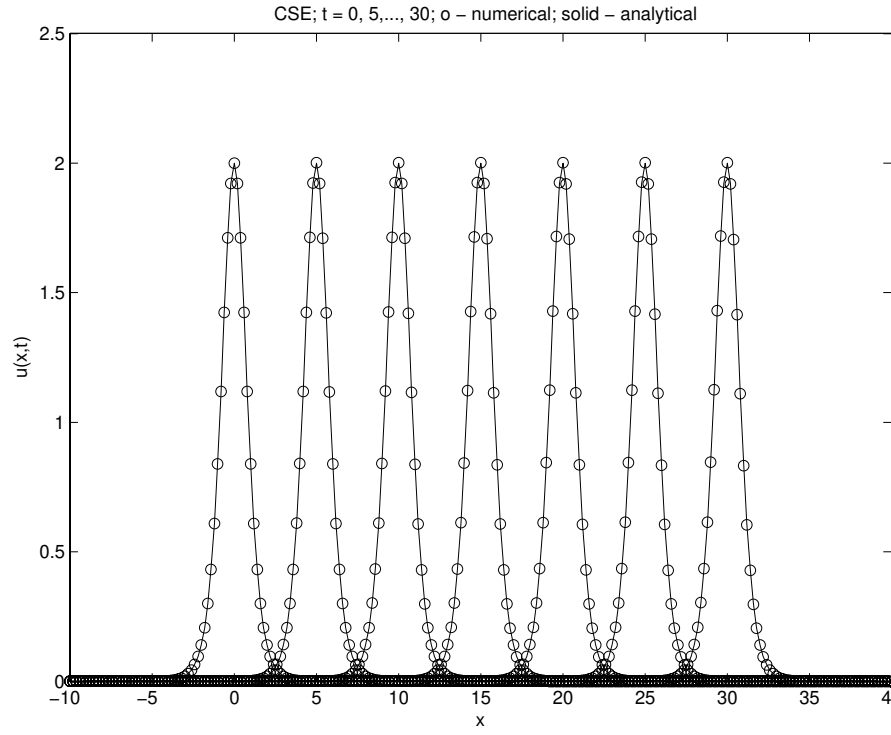

Figure 6.1. Output of main program `pde_1_main`

We can note the following points about this output:

1. The soliton moves left to right in x, as indicated by the movement of the peak. See Table 6.2 for a summary of the peak movement at $t = 0, 5, \ldots, 30$ corresponding to $x = 0, 5, \ldots, 30$.

Table 6.2. Partial Output from `pde_1_main` and `pde_1`, illustrating the movement of the solution with unit velocity

t	x	x - t	v^2+w^2 num	v^2+w^2 anal	err
0.00	0.0	0.00	2.000000	2.000000	0.000000
5.00	5.0	0.00	2.000991	2.000000	0.000991
10.00	10.0	0.00	2.001183	2.000000	0.001183
15.00	15.0	0.00	2.001288	2.000000	0.001288
20.00	20.0	0.00	2.000912	2.000000	0.000912
25.00	25.0	0.00	2.001210	2.000000	0.001210
30.00	30.0	0.00	2.000883	2.000000	0.000883

In this case, the maximum error in the peak amplitude is `0.001288`, which is 0.064% of the exact value of 2.

2. A similar summary of the three invariants is given in Table 6.3. `I3` has a maximum variation of $[-13.8880-(-12.8329)]/(-13.8880)\times 100 = 7.6\%$. This variation probably reflects the accuracy of the MOL solution due to the limited number of grid points ($n = 251$) and possibly the associated limited accuracy of the Simpson's rule integration in `simp`. But increasing the number of grid points also increases the computer run time that is already rather substantial as reflected in the value `ncall = 18523`. The run time would increase with n because (a) more ODEs would be integrated, and (b) the ODE stiffness generally increases with more grid points; if the latter is the case, then a switch from `ode45` to `ode15s` would be logical.

Table 6.3. Partial output from `pde_1_main` and `pde_1`, illustrating the invariants `I1,I2,I3` of Eqs. (6.4a)–(6.4c)

```
( t = 0)  I1 =     6.7303
          I2 =     1.9997
          I3 =   -13.8880

( t = 5)  I1 =     6.7252
          I2 =     1.9997
          I3 =   -13.8635

(t = 10)  I1 =     6.7089
          I2 =     1.9997
          I3 =   -13.7750

(t = 15)  I1 =     6.6814
          I2 =     1.9997
          I3 =   -13.6262

(t = 20)  I1 =     6.6427
          I2 =     1.9997
          I3 =   -13.4157

(t = 25)  I1 =     6.5935
          I2 =     1.9997
          I3 =   -13.1521

(t = 30)  I1 =     6.5337
          I2 =     1.9997
          I3 =   -12.8329
```

3. However, the agreement between the numerical and analytical solutions is quite good as reflected in the differences `err` of Tables 6.1 and 6.2, and the plotted output shown in Figure 6.1.

The solution starts with the IC pulse (defined by Eq. (6.3)) centered at $x = 0$ (for $t = 0$). The movement of the traveling wave left to right according to Eq. (6.2b) for $t = 0, 5, \ldots, 30$ is evident in Figure 6.1.

We now consider the subordinate routines called by the main program, `pde_1_main` of Listing 6.1, `inital_1, ua, simp and pde_1`. `inital_1` is listed first (see Listing 6.2).

```
  function u0=inital_1(t0)
%
% Function inital_1 sets the initial condition for the CSE
%
  global ncall     q     x     xl     xu   rt2     n
%
% q in cubic Schrodinger equation (CSE)
  q=1.0;
%
% Precompute 2^0.5
  rt2=sqrt(2.0);
%
% Grid spacing
  dx=(xu-xl)/(n-1);
%
% IC over the spatial grid
  for i=1:n
%
%   Uniform grid
    x(i)=xl+(i-1)*dx;
%
%   Initial real, imaginary parts (v(x,0), w(x,0)), absolute
%   value of u(x,0) = u0(x) = (2^0.5)exp(0.5*i*x)sech(x),
%   i = (-1)^0.5 over xl <= x <= xu
%
%   sech(x)
    sch=2.0/(exp(x(i))+exp(-x(i)));
%
%   Real part
    v(i)=rt2*cos(0.5*x(i))*sch;
%
%   Imaginary part
    w(i)=rt2*sin(0.5*x(i))*sch;
  end
```

```
%
% Two vectors to one vector
  for i=1:n
    u0(i)  =v(i);
    u0(i+n)=w(i);
  end
```

Listing 6.2. Initialization routine `inital_1`

We can note the following details about `inital_1`:

1. After the routine is defined and some parameters are declared as global, a spatial grid is defined for $-10 \leq x \leq 40$ (recall that the number of grid points, n, and the left and right boundary values of x, `xl` and `xu`, are set in main program `pde_1_main` as global parameters).

```
  function u0=inital_1(t0)
%
% Function inital_1 sets the initial condition for the CSE
%
  global ncall     q     x     xl     xu   rt2     n
%
% q in cubic Schrodinger equation (CSE)
  q=1.0;
%
% Precompute 2^0.5
  rt2=sqrt(2.0);
%
% Grid spacing
  dx=(xu-xl)/(n-1);
%
% IC over the spatial grid
  for i=1:n
%
%   Uniform grid
    x(i)=xl+(i-1)*dx;
```

2. Finally, the real and imaginary parts of the IC of Eq. (6.3) are computed.

```
%
%   Initial real, imaginary parts (v(x,0), w(x,0)), absolute
%   value of u(x,0) = u0(x) = (2^0.5)exp(0.5*i*x)sech(x),
```

```
%   i = (-1)^0.5 over xl <= x <= xu
%
%   sech(x)
    sch=2.0/(exp(x(i))+exp(-x(i)));
%
%   Real part
    v(i)=rt2*cos(0.5*x(i))*sch;
%
%   Imaginary part
    w(i)=rt2*sin(0.5*x(i))*sch;
  end
%
% Two vectors to one vector
  for i=1:n
    u0(i)  =v(i);
    u0(i+n)=w(i);
  end
```

Note that a single vector of ODE dependent variables `u0` is defined as required by the calls to `ode45` and `ode15s` in `pde_1_main`.

The analytical solution of Eq. (6.2b) is programmed in `ua` (see Listing 6.3).

```
  function uanal=ua(t,x)
%
% Function ua computes the analytical solution to the CSE
%
% The following exact solution is |u(x,t)|^2 which is compared
% with the numerical solution
%
  uanal=2.0*(2.0/(exp(x-t)+exp(-(x-t))))^2;
```

Listing 6.3. Analytical solution routine `ua` for Eq. (6.2b)

The code in `ua` is essentially self-explanatory when compared with Eq. (6.2b) (recall $\text{sech}(x) = 1/\cosh(x) = 2/(e^x + e^{-x})$ and note the argument for the traveling wave solution, $x - t$).

The routine for calculating the integrals of Eqs. (6.4a)–(6.4c) by Simpson's rule, `simp`, is given in Listing 6.4.

```
  function uint=simp(xl,xu,n,u)
%
% Function simp computes three integral invariants by
```

```
% Simpson's rule
%
  global q
%
% One vector to two vectors
  for i=1:n
    v(i)=u(i);
    w(i)=u(i+n);
  end
%
% Three invariants
  for int=1:3
    h=(xu-xl)/(n-1);
%
%   I1
    if(int==1)
       uabs(1)=u(1)^2+v(1)^2;
       uabs(n)=u(n)^2+v(n)^2;
       uint(1)=uabs(1)-uabs(n);
       for i=3:2:n
         uabs(i-1)=u(i-1)^2+v(i-1)^2;
         uabs(i  )=u(i  )^2+v(i  )^2;
         uint(1)=uint(1)+4.0*uabs(i-1)+2.0*uabs(i);
       end
       uint(1)=h/3.0*uint(1);
    end
%
%   I2
    if(int==2)
       vx=dss004(xl,xu,n,v);
       wx=dss004(xl,xu,n,w);
       int(1)=v(1)*wx(1)-w(1)*vx(1);
       int(n)=v(n)*wx(n)-w(n)*vx(n);
       uint(2)=int(1)-int(n);
       for i=3:2:n
         int(i-1)=v(i-1)*wx(i-1)-w(i-1)*vx(i-1);
         int(i  )=v(i  )*wx(i  )-w(i  )*vx(i  );
         uint(2)=uint(2)+4.0*int(i-1)+2.0*int(i);
       end
       uint(2)=h/3.0*uint(2);
%
%   I3
    if(int==3)
       uxabs(1)=vx(1)^2+wx(1)^2;
       uxabs(n)=vx(n)^2+wx(n)^2;
       uint(3)=uxabs(1)-0.5*q*uabs(1)^2 ...
                       -(uxabs(n)-0.5*q*uabs(n)^2);
```

```
      for i=3:2:n
        uxabs(i-1)=vx(i-1)^2+wx(i-1)^2;
        uxabs(i)  =vx(i  )^2+wx(i  )^2;
        uint(3)=uint(3)+4.0*(uxabs(i-1)-0.5*q*uabs(i-1)^2) ...
                       +2.0*(uxabs(i  )-0.5*q*uabs(i  )^2);
      end
      uint(3)=2.0*h/3.0*uint(3);
    end
  end
```

Listing 6.4. Numerical quadrature routine `simp` applied to Eqs. (6.4a)–(6.4c)

`simp` has three parts corresponding to the integrals `I1,I2,I3` of Eqs. (6.4a)–(6.4c).

1. The coding for `I1` is

```
  function uint=simp(xl,xu,n,u)
%
% Function simp computes three integral invariants by
% Simpson's rule
%
  global q
%
% One vector to two vectors
  for i=1:n
    v(i)=u(i);
    w(i)=u(i+n);
  end
%
% Three invariants
  for int=1:3
    h=(xu-xl)/(n-1);
%
%   I1
    if(int==1)
      uabs(1)=u(1)^2+v(1)^2;
      uabs(n)=u(n)^2+v(n)^2;
      uint(1)=uabs(1)-uabs(n);
      for i=3:2:n
        uabs(i-1)=u(i-1)^2+v(i-1)^2;
        uabs(i  )=u(i  )^2+v(i  )^2;
        uint(1)=uint(1)+4.0*uabs(i-1)+2.0*uabs(i);
      end
      uint(1)=h/3.0*uint(1);
    end
```

After defining the function, vector `u` is divided into two vectors, `v` and `w`, according to $u(x,t) = v(x,t) + iw(x,t)$. The integration interval `h=(xl-xu)/(n-1)` is then computed, where `xl=-10` and `xu=40` are the lower and upper limits of the integral (set in `inital_1`). The `for` loop for `I1` is an implementation of the weighted sum for Simpson's rule applied to the function $|u(x,t)|^2$ (the integrand in Eq. (6.4a)).

$$\int_{-\infty}^{\infty} u(x,t)dx \approx \frac{h}{3}\left[|u_1|^2 + \sum_{i=2}^{n-2}\left(4|u_i|^2 + 2|u_{i+1}|^2\right) + |u_n|^2\right]$$

2. Similarly, the coding for `I2` of Eq. (6.4b) is

```
%
%   I2
    if(int==2)
       vx=dss004(xl,xu,n,v);
       wx=dss004(xl,xu,n,w);
       int(1)=v(1)*wx(1)-w(1)*vx(1);
       int(n)=v(n)*wx(n)-w(n)*vx(n);
       uint(2)=int(1)-int(n);
       for i=3:2:n
         int(i-1)=v(i-1)*wx(i-1)-w(i-1)*vx(i-1);
         int(i  )=v(i  )*wx(i  )-w(i  )*vx(i  );
         uint(2)=uint(2)+4.0*int(i-1)+2.0*int(i);
       end
       uint(2)=h/3.0*uint(2);
    end
```

The only difference is the use of the integrand $v(x,t)w_x(x,t) - w(x,t)v_x(x,t)$ according to Eq. (6.4b). The derivatives $v_x(x,t)$ and $w_x(x,t)$ are computed by calls to the differentiation routine `dss004`.

3. Finally, the coding for `I3` of Eq. (6.4c) is

```
%
%   I3
    if(int==3)
       uxabs(1)=vx(1)^2+wx(1)^2;
       uxabs(n)=vx(n)^2+wx(n)^2;
       uint(3)=uxabs(1)-0.5*q*uabs(1)^2 ...
                       -(uxabs(n)-0.5*q*uabs(n)^2);
       for i=3:2:n
         uxabs(i-1)=vx(i-1)^2+wx(i-1)^2;
         uxabs(i)  =vx(i  )^2+wx(i  )^2;
         uint(3)=uint(3)+4.0*(uxabs(i-1)-0.5*q*uabs(i-1)^2) ...
                         +2.0*(uxabs(i  )-0.5*q*uabs(i  )^2);
```

```
      end
      uint(3)=2.0*h/3.0*uint(3);
    end
  end
```

The integrand is $|u_x(x,t)|^2 - 0.5q|u(x,t)|^4$ according to Eq. (6.4c).

The MOL ODE routine, `pde_1`, called by `ode45` or `ode15s` in `pde_1_main` of Listing 6.1, is given in Listing 6.5

```
  function ut=pde_1(t,u)
%
% Function yt computes the t derivative vector of the CSE
%
  global  ncall      q     xl     xu      n
%
% One vector to two vectors
  for i=1:n
    v(i)=u(i);
    w(i)=u(i+n);
  end
%
% PDEs
%
%   v
%    xx
%
  vx=zeros(n,1);
  v(1)=0.0;
  v(n)=0.0;
  nl=1;
  nu=1;
  vxx=dss044(xl,xu,n,v,vx,nl,nu);
%
%   w
%    xx
  wx=zeros(n,1);
  w(1)=0.0;
  w(n)=0.0;
  nl=1;
  nu=1;
  wxx=dss044(xl,xu,n,w,wx,nl,nu);
%
% ODEs at the boundaries
  vt(1)=0.0;
  wt(1)=0.0;
```

```
  vt(n)=0.0;
  wt(n)=0.0;
%
% ODEs at the interior points
  for i=2:n-1
%
%   v**2 + w**2
    v2w2(i)=v(i)^2+w(i)^2;
%
%   vt
    vt(i)=-wxx(i)-q*v2w2(i)*w(i);
%
%   wt
    wt(i)= vxx(i)+q*v2w2(i)*v(i);
  end
%
% Two vectors to one vector
  for i=1:n
    ut(i)  =vt(i);
    ut(i+n)=wt(i);
  end
  ut=ut';
%
% Increment calls to pde_1
  ncall=ncall+1;
```

Listing 6.5. MOL ODE routine `pde_1` called by main program `pde_1_main`

We can note the following points about `pde_1`:

1. After the definition of the function and the global declaration of some parameters and variables, the dependent variable vector `u` is stored as two vectors, `v` and `w`, according to $u(x,t) = v(x,t) + iw(x,t)$.

```
  function ut=pde_1(t,u)
%
% Function yt computes the t derivative vector of the CSE
%
  global  ncall      q     xl     xu      n
%
% One vector to two vectors
  for i=1:n
    v(i)=u(i);
    w(i)=u(i+n);
  end
```

2. The second derivatives v_{xx} and w_{xx} in Eq. (6.1) are then computed by a call to `dss044` that uses a five-point finite-difference (FD) approximation for these second derivatives.

```
%
% PDEs
%
%   v
%    xx
%
  vx=zeros(n,1);
  v(1)=0.0;
  v(n)=0.0;
  nl=1;
  nu=1;
  vxx=dss044(xl,xu,n,v,vx,nl,nu);
%
%   w
%    xx
  wx=zeros(n,1);
  w(1)=0.0;
  w(n)=0.0;
  nl=1;
  nu=1;
  wxx=dss044(xl,xu,n,w,wx,nl,nu);
```

There are a few details about this calculation of second derivatives that require explanation:

(a) The first derivative arrays, `vx` and `ux`, which are input arguments to `dss044`, are not actually used in the calculation of `uxx` and `wxx`. However, Matlab requires that at least one element of an array that is an input to a function has a numerical value. This is accomplished by setting all of the elements of `v` and `w` to zero through the Matlab `zeros` utility.

(b) Again, BCs for Eq. (6.1) are not required since $-10 \leq x \leq 40$ is equivalent to $-\infty \leq x \leq \infty$. However, the calculation of v_{xx} and w_{xx} by `dss044` requires BCs (since this is a general purpose routine that is typically applied to a finite spatial domain that requires BCs), so the Dirichlet BCs $v(x = -10, t) = v(x = 40, t) = 0$ and $w(x = -10, t) = w(x = 40, t) = 0$ are used (and programmed as `v(1)=0.0, v(n)=0.0, w(1)=0.0, w(n)=0.0`). Also, these Dirichlet BCs must be identified before calling `dss044` by `nl=1, nu=1` for the lower ($x = -10$) and upper ($x = 40$) boundary values of x.

3. Now that all of the spatial derivatives in Eq. (6.1) have been computed, the MOL programming of Eq. (6.1) can be added.

```
%
% ODEs at the boundaries
  vt(1)=0.0;
  wt(1)=0.0;
  vt(n)=0.0;
  wt(n)=0.0;
%
% ODEs at the interior points
  for i=2:n-1
%
%   v**2 + w**2
    v2w2(i)=v(i)^2+w(i)^2;
%
%   vt
    vt(i)=-wxx(i)-q*v2w2(i)*w(i);
%
%   wt
    wt(i)= vxx(i)+q*v2w2(i)*v(i);
  end
%
% Two vectors to one vector
  for i=1:n
    ut(i)  =vt(i);
    ut(i+n)=wt(i);
  end
  ut=ut';
%
% Increment calls to pde_1
  ncall=ncall+1;
```

Note that the derivatives in t at the boundaries are set to zero (to reflect the Dirichlet BCs). Then the real and imaginary parts of u are programmed for the interior points $2 \le i \le n-1$ according to Eq. (6.1). In particular, the programming of the nonlinear term $-q|u|^2u$ is easily accomplished. Then the two derivative vectors, `vt` and `wt`, are returned from `pde_1` as a single vector, `ut`, for integration by `ode45` or `ode15s`. The usual transpose of `ut` is included, as well as the updating of the counter for the calls to `pde_1`.

We conclude this chapter by mentioning the following concepts:

1. The complex CSE can be integrated numerically as a 2×2 system of real PDEs.
2. The traveling wave solution of Eqs. (6.2a) and (6.2b) is demonstrated by the numerical MOL solution.
3. The invariants of Eqs. (6.4a)–(6.4c) can be computed numerically from the MOL solution.

4. This calculation of the invariants demonstrates another important aspect of the numerical solution, namely the availability of all of the solution derivatives in x and t, for example, u_t, u_{xx}, as well as functions of the dependent variable, for example, $|u|^2u$. Thus, in addition to displaying the solution, u, the analysis can also be extended to a display of all of the terms in the problem PDE(s), Eq. (6.1) in this case. We have found that displaying these terms along with the solution often gives enhanced visualization and understanding of the solution (at essentially no cost since these terms are readily available from the MOL solution because they are computed as part of the MOL solution, generally in the ODE routine, e.g., `pde_1`).

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

```
% 3D Plots
  figure(2)
  surfl(x,t,v2w2, 'light');
  shading interp
  title('Cubic Schrodinger Equation');
  set(get(gca,'XLabel'),'String','space, x')
  set(get(gca,'YLabel'),'String','time, t')
  set(get(gca,'ZLabel'),'String','dependent variable, u(t,x)')
  set(gca,'XLim',[-10 40],'YLim',[0 30],'ZLim', [0 2]);
  axis tight
  view(-10,45)
  colormap('cool');   % set color map
  print -dpng -r300 fig2.png;
```

The resulting 3D plot is shown in Figure 6.2.

Additional enhanced plotting of $|u|^2$ for the Eq. (6.1) solution can be accomplished by animation (a movie). The details for producing an animation are given in Appendix 6.

APPENDIX A: SOME BACKGROUND TO SCHRÖDINGER'S EQUATION

A.1. Introduction

De Broglie[1] was the first to develop a wave theory for the behavior of an electron, postulating in his Ph.D. thesis that the wavelength of any matter obeyed the relationship $\lambda = h/p$, where h is Planck's constant and p is momentum. His approach successfully solved some simple problems, but his theory did not adequately explain the behavior of an electron when subjected to different types of external potential fields. This led *Erwin Schrödinger* to investigate alternatives and, ultimately, to

[1] Full name *Louis-Victor, Pierre, Raymond 7th duc de Broglie*, usually shortened to de Broglie.

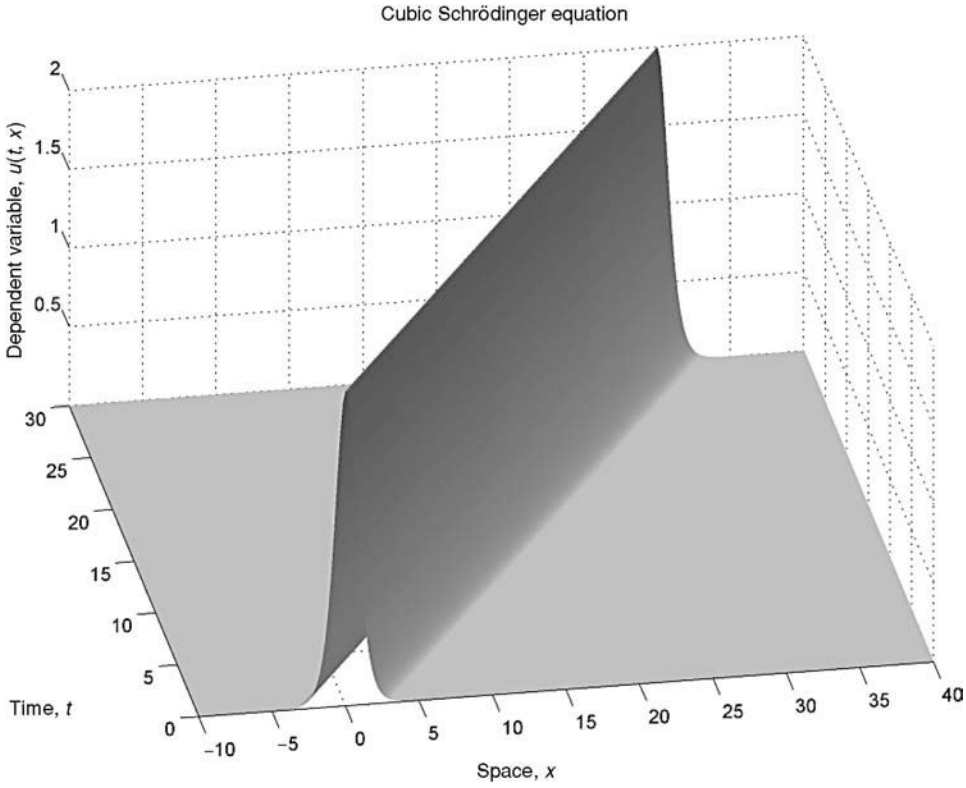

Figure 6.2. Three-dimensional output for main program `pde_1_main` with `nout=301, n=251`

derive a *wave function* solution to describe the electron that, crucially, conserved energy [2]. He then systematically applied his equation, the so-called *Schrödinger equation*, to problems involving different potentials applied to the electron, and the solutions were subsequently confirmed experimentally. It is the basis for our current understanding of the electron.

The *nonlinear Schrödinger equation* (NLS), or *cubic Schrödinger equation* (CSE), has the following *normalized* form in 3D space:

$$i\frac{\partial u(x,t)}{\partial t}+\frac{\partial^2 u(x,t)}{\partial x^2}+qu(x,t)\left|u(x,t)\right|^2=0,\quad x=(x_1,x_2,x_3) \tag{6.5}$$

where $q=\pm 1$. For $q=+1$, it is known as the focusing case and for $q=-1$, as the defocusing case (other nonzero values of q can be reduced to these two cases by rescaling the values of u [3]).

The cubic nonlinearity is one of the most common nonlinearities found in applications. It arises as a simplified model for studying Bose-Einstein condensates, Kerr media in nonlinear optics, and even freak waves in the ocean [3, 4].

An interesting property of the CSE is that it can sustain *solitary waves* or *solitons*, and it is this property that we have explored in the main text for the 1D case. A *soliton* is a *solitary wave* that has the additional property that other solitons can pass through it without changing its shape. The only difference after they emerge from the collision is that each exhibits a small phase shift.

A.2. A Nonrigorous derivation

A.2.1. The Hamiltonian

The Hamiltonian of an electron with mass m, momentum p, and coordinate q acted on by a conservative force is given by

$$H(q,p) = \frac{|p|^2}{2m} + V(q) \tag{6.6}$$

where physically, the quantity $|p|^2/2m$ represents *kinetic energy*, $V(q)$ represents *potential energy*, and $H(q,p)$ represents *total energy*. The motion of the electron then follows from *Hamiltonian's equations of motion*; that is,

$$\frac{\partial H}{\partial p} = \dot{q}(t); \qquad \frac{\partial H}{\partial q} = -\dot{p}(t) \tag{6.7}$$

where p and q are *vectors*, and $\dot{p}(t)$ and $\dot{q}(t)$ are the corresponding *gradients*. *Note:* This q should not be confused with the q in Eq. (6.5). Taking derivatives of Eq. (6.6), we obtain

$$\dot{q}(t) = \frac{1}{m}p(t); \qquad \dot{p}(t) = -\nabla V(q) \tag{6.8}$$

A consequence of Hamiltonian's equations of motion is that we obtain directly the principle of *conservation of energy*. Thus, we have $H(q(t), p(t)) = \text{constant}$, from which it follows that

$$\frac{d}{dt}H(q(t), p(t)) = 0, \quad \forall\, t \tag{6.9}$$

A.2.2. The Wave Function

Schrödinger's great insight was to make the assumption that the motion of an electron can be described by an appropriate *wave function*, which we choose to be of the following form:

$$u(x,t) \simeq \widehat{u}(x,t) = \sum_{\ell=1}^{\infty}\sum_{n=1}^{\infty} a_{\ell n}\, e^{i(k_\ell x - \omega_n t)} \tag{6.10}$$

where

$a_{\ell n}$ wave function coefficients
k_ℓ wave number ℓ
ω_n frequency n (r/s)
x position
t time

But we have the Planck-Einstein relation $E = h\nu = h\omega/2\pi = \hbar\omega$ and, also from de Broglie the relation, $k = 2\pi/\lambda = 2\pi(p/\hbar)$,
where

h Planck's constant
$\hbar$ reduced Planck's constant
ν frequency (1/s)
E energy
λ wave length

Thus, we obtain

$$\widehat{u}(x,t) = \sum_{\ell=1}^{\infty}\sum_{n=1}^{\infty} a_{\ell n}\, e^{i(p_\ell x/\hbar - E_n t/\hbar)} \tag{6.11}$$

A.2.3. Schrödinger's Equation

We now let $u_{\ell n}(x,t) = a_{\ell n}\, e^{i(p_\ell x/\hbar - E_n t/\hbar)}$ and for each term in the preceding summation we take the first derivative with respect to t and the second derivative with respect to x, to obtain

$$\frac{\partial u_{\ell n}}{\partial t} = u_{\ell n}(x,t)\left(-i\frac{E_n}{\hbar}\right) \tag{6.12}$$

$$\frac{\partial^2 u_{\ell n}}{\partial x^2} = u_{\ell n}(x,t)\left(-\frac{p_\ell^2}{\hbar^2}\right) \tag{6.13}$$

On rearranging and dropping the summation indices, we obtain

$$-\frac{\hbar}{i}\frac{\partial u}{\partial t} = u(x,t)\,E \tag{6.14}$$

$$-\hbar^2\frac{\partial^2 u}{\partial x^2} = u(x,t)\,p^2 \tag{6.15}$$

It should be noted that in quantum mechanics, unlike classical mechanics, a wave function $u(x,t)$ does not necessarily have specific position or momentum. Rather we consider it to possess an average speed and average position given by $\langle q(t)\rangle$ and $\langle p(t)\rangle$, respectively, where $\langle\cdot\rangle$ represents the *inner product* [5].

Now, on substituting Eqs. (6.14) and (6.15) into the Hamiltonian, Eq. (6.6), where we take $E = H$, $x = \langle q(t)\rangle$, and $p = \langle p(t)\rangle$, we obtain

$$H = E = -\frac{\hbar}{i}\frac{\partial u(x,t)}{\partial t}\left(\frac{1}{u(x,t)}\right) = -\frac{\hbar^2}{2m}\frac{\partial^2 u(x,t)}{\partial x^2}\left(\frac{1}{u(x,t)}\right) + V(x) \tag{6.16}$$

On rearranging we obtain the *Schrödinger equation* corresponding to a particular energy level,

$$i\hbar\frac{\partial u(x,t)}{\partial t} = -\frac{\hbar^2}{2m}\frac{\partial^2 u(x,t)}{\partial x^2} + u(x,t)\,V(x) \tag{6.17}$$

Finally, we now let $V(x) = -q|u|^2$ and normalize $\hbar$ and $2m$ to unity, when we obtain the preceding *normalized nonlinear* or *cubic Schrödinger equation*; that is,

$$i\frac{\partial u}{\partial t} + \frac{\partial^2 u}{\partial x^2} + qu|u|^2 = 0 \tag{6.18}$$

In Eq. (6.18) the term $|u|^2$ represents the *probability density* of the wave function. This means that the probability density for an electron being in a state defined by $u(x,t)$ is given by $p(x,t) = u^*(x,t)\,u(x,t) = |u|^2$, where u^* represents the complex conjugate of u (where, $p(x,t)$ should not be confused with p in the Hamiltonian).

The *normalization condition* implies that the spatial probability is given by

$$P(t) = \int_\Gamma p(x,t)\,dx = \int_\Gamma \left|u(x,t)\right|^2 dx = 1 \tag{6.19}$$

that is, the probability of the electron being located somewhere within the domain under consideration Γ is one.

Note: We can seek solutions to the Schrödinger equation (6.16) using the *separation of variables* (SOV) method, whereby we assume a solution of the form $u(x,t) = T(t)\,X(x)$. This leads to the *time-independent Schrödinger equation*

$$-\frac{\hbar^2}{2m}\nabla^2 u + Vu = Eu \tag{6.20}$$

and, under certain conditions, to the corresponding energy relationship

$$E = E_n = \frac{n^2\pi^2\hbar^2}{2ma^2} \tag{6.21}$$

where $0 \le x \le a$. This illustrates one of the extraordinary features of quantum mechanics, namely, that the energy of a system can take only certain discrete values [6].

For further details relating to the physics of Schrödinger's equation, readers are referred to references [6, 7].

REFERENCES

[1] Myint-U, T. and L. Debnath (1987), *Partial Differential Equations for Scientists and Engineers*, 3rd ed., North Holland, Amsterdam

[2] Schrodinger, E. (1926), Quantisierung als Eigenwertproblem, I–IV, *Annalen der Physik*, **79**:361–376, 489–527; **80**:437–490; **81**:109–139. Reprinted (English translation) in Erwin Schrodinger, *Collected Papers on Wave Mechanics*, Blackie & Son, London, 1928, pp. 325–432

[3] Killip, R., T. Tao, and M. Visan (2007), The Cubic Nonlinear Schrödinger Equation in Two Dimensions with Radial Data, *arXiv:0707.3188v1 [math.AP]*; available online at `http://arxiv.org/abs/0707.3188v`

[4] Antoine, X., C. Besse, and S. Descombes (2006), Artificial Boundary Conditions for One-Dimensional Cubic Nonlinear Schrodinger Equations, *SIAM J. Num. Anal.*, **43**(6): 2272–2293

[5] Tao, T. (2007), The Schrödinger Equation, In: *Princeton Companion to Mathematics*; available online at `http://www.math.ucla.edu/ tao/preprints/schrodinger.pdf`

[6] Hannabuss, K. (1997), *An Introduction to Quantum theory*, Oxford University Press, Oxford, UK

[7] French, A. P., and E. F. Taylor (1992), *An Introduction to Quantum Physics*, Chapman & Hall, Boca Raton, FL