

CHAPTER III METHODOLOGY

In the previous chapter, we have learnt much about the comprehensive literature review of the problem; starting from the derivation of the Schrödinger equation down to how researchers used other numerical computational methods to find the numerical solutions of different kinds of linear and nonlinear Schrödinger equation dating back since 1970s to date, we have also shown what MOL is and we discussed fully how this method can effectively used to find the numerical solution of most of the PDE system of equation. In the preceding chapter, we are going to give account of how this method can be fully used and applied to this problem of ours – Schrödinger equation.

3.1 Element of MOL

Basically, to apply the Mol normally involves the following stages;

1. Partitioning the solution region into layers
2. Discretizing the differential equation in one coordinate direction
3. Transformation to obtain decoupled ODE
4. Inverse transformation and introduction of boundary conditions
5. Solution of the equations

Or in clearer picture, as regards to the use of steps above, Hamdi et al. (2007) and Schiesser & Griffith (2009), said - The basic idea of the MOL is to *replace the spatial (boundary value) derivatives in the PDE with algebraic approximations*. Once this is done, the spatial derivatives are no longer stated explicitly in terms of the spatial independent variables. Thus, in effect *only the initial value variable, typically time in a physical problem, remains*. In other words, with only one remaining independent variable, we have a *system of ODEs* that approximate the original PDE. *The challenge, then, is to formulate the approximating system of ODEs*. Once this is done, we can apply any integration algorithm for initial value ODEs to compute an approximate

numerical solution to the PDE. Thus, one of the salient features of the MOL is the use of *existing, and generally well established, numerical methods for ODEs*.

Let us start by illustrating this procedure or methodology with some examples suggested by Hamdi et al. (2007) and Schiesser & Griffith (2009). We consider the MOL solution of equation (2.2.5.1). Following the above steps, we need to first replace the spatial derivative u_x with an algebraic approximation. In this case we will use a *finite difference (FD)* such as

$$u_x \approx \frac{u_i - u_{i-1}}{\Delta x} \quad (3.1.1)$$

where i is an index designating a position along a grid in x and Δx is the spacing in x along the grid, assumed constant for the time being. Thus, for the left end value of x , $i = 1$, and for the right end value of x , $i = M$, i.e., the grid in x has M points. Then the MOL approximation of eq. (2.2.5.1) is

$$\frac{du_i}{dt} = -v \frac{u_i - u_{i-1}}{\Delta x}, \quad 1 \leq i \leq M \quad (3.1.2)$$

Note that equation (3.1.2) is written as an ODE, since there is now *only one independent variable*, t . Note also that equation (3.1.2) represents a system of M ODEs.

This transformation of a PDE, equation (2.2.5.1), to a system of ODEs, equations (3.1.2), illustrates the essence of the MOL, namely, the replacement of the spatial derivatives, in this case u_x , so that the solution of a system of ODEs approximates the solution of the original PDE. Then, to compute the solution of the PDE, we compute a solution to the approximating system of ODEs. But before considering this integration in t , we have to complete the specification of the PDE problem. Since equation (2.2.5.1) is first order in t and first order in x , it requires one IC and one BC. These will be taken as

$$u(x, t = 0) = f(x) \quad (3.1.3a)$$

$$u(x = 0, t) = g(t) \quad (3.1.3b)$$

Since equation (3.1.2) constitute M initial value ODEs, M initial conditions are required and from equation (3.1.3a), these are

$$u(x_i, t = 0) = f(x_i), \quad 1 \leq i \leq M \quad (3.1.4a)$$

Also, application of BC (3.1.3b) gives for grid point $i = 1$

$$u(x_1, t) = g(t), \quad t \geq 0 \quad (3.1.4b)$$

Equations (3.1.2) and (3.1.4) now constitute the complete MOL approximation of equation (2.2.5.1) subject to the initial and boundary conditions stated in equations (3.1.3a) and (3.1.3b). The solution of this ODE system gives the M functions

$$u_1(t), u_2(t), \dots, u_{M-1}(t), u_M(t) \quad (3.1.5)$$

that is, an approximation to $u(x, t)$ at the grid points $i = 1, 2, \dots, M$.

What remains now is the consideration of the numerical integration of the approximating ODE, in this case, equation (3.1.2), but before then, let us consider further the finite difference (FD) approximation of equation (3.1.1) which can be written as

$$u_x \approx \frac{u_i - u_{i-1}}{\Delta x} + O(\Delta x) \quad (3.1.6)$$

where $O(\Delta x)$ denotes of order Δx , that is, the truncation error (from a truncated Taylor series) of the approximation of equation (3.1.2) is proportional to Δx (varies linearly with Δx); thus equation (3.1.6) is also termed a *first order* FD (since Δx is to the first power in the order or truncation error term). The term truncation error

reflect the fact that the finite difference of equation (3.1.1) comes from a truncated Taylor series.

When we note equation (3.1.1) again, we will see that, the numerator $u_i - u_{i-1}$, is a difference in two values of u . Also, the denominator Δx remains finite (nonzero). Hence the name finite difference (and it is an approximation because of the truncated Taylor series, so a more complete description is *first order* finite difference approximation). In fact, in the limit $\Delta x \rightarrow 0$ the approximation of equation (3.1.1) becomes *exactly the derivative*. However, in a practical computed-based calculation, Δx remains finite, so equation (3.1.1) remains as an approximation.

Also, equation (2.2.5.1) typically describes the flow of a physical quantity such as concentration of a chemical species or temperature, represented by u , from left to right with respect to x with velocity v . Then, using the FD approximation of equation (3.1.6) at i involves u_i and u_{i-1} . In a flowing system, u_{i-1} is to the left (in x) of u_i or is *upstream* or *upwind* of u_i (to use a nautical analogy). Thus, equation (3.1.6) is termed a *first order upwind FD approximation*. Generally, for strongly convective systems such as modeled by equation (2.2.5.1), *some form* of unwinding is required in the numerical solution of the descriptive PDEs; we will look at this requirement further in the subsequent discussion.

3.2 Integrating ODE with MOL

Let us now consider the numerical integration of the M ODEs of equation (3.1.2). If we approximate the derivative $\frac{du_i}{dt}$ by first order FD

$$\frac{du_i}{dt} \approx \frac{u_i^{n+1} - u_i^n}{\Delta t} + O(\Delta t) \quad (3.2.1)$$

where n is an index for the variable t (t moves forward in steps denoted or indexed by n), then a FD remains an approximation to the derivative of equation (3.1.2) is

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = -v \frac{u_i^n - u_{i-1}^n}{\Delta x} + O(\Delta t)$$

or solving for u_i^{n+1} ,

$$(u_i^{n+1} - u_i^n) \Delta x = v(-u_i^n - u_{i-1}^n) \Delta t$$

$$u_i^{n+1} \Delta x = v(-u_i^n - u_{i-1}^n) \Delta t + u_i^n \Delta x$$

$$u_i^{n+1} = -v \frac{\Delta t}{\Delta x} (u_i^n - u_{i-1}^n) + u_i^n$$

$$u_i^{n+1} = u_i^n - (v \Delta t / \Delta x)(u_i^n - u_{i-1}^n), i = 1, 2, \dots, M \quad (3.2.2)$$

Equation (3.2.2) has the important characteristic that it gives u^{n+1} explicitly, that is, we can solve for the solution at the advanced point in t , $n+1$, from the solution at the base point n . In other words, explicit numerical integration of equation (3.1.2) is by the *forward FD* of equation (3.2.1), and this procedure is generally termed the *forward Euler method* which is the most basic form of ODE integration.

While the explicit form of equation (3.2.2) is computationally convenient, it has a possible limitation. If the time step Δt is above a critical value, the calculation becomes unstable, which is manifest by successive changes in the dependent variable, $\Delta u = u_i^{n+1} - u_i^n$, becoming larger and eventually unbounded as the calculation moves forward in t (for increasing n). In fact, for the solution of equation (2.2.5.1) by the method of equation (3.2.2) to remain stable, the dimensionless group $v(\Delta t / \Delta x)$, which is called the *Courant-Friedricks-Lewy or CFL number*, must remain below a critical value, in this case, unity. Note that this *stability limit* places an upper limit on Δt for a given v and Δx ; if one attempts to increase the accuracy of equation (3.2.2) by using a smaller Δx (larger number of grid points in x by increasing M), a smaller value of Δt is required to keep the CFL number below its critical value. Thus, there is a *conflicting requirement of improving accuracy while maintaining stability*.

Let us now circumvent the stability limit of the explicit Euler method as implemented via the forward FD of equation (3.2.1) by using a *backward FD* for the derivative in t . therefore

$$\frac{du_i}{dt} \approx \frac{u_i^n - u_i^{n-1}}{\Delta t} + O(\Delta t) \quad (3.2.3)$$

so that the FD approximation of equation (3.1.2) becomes

$$\frac{u_i^n - u_i^{n-1}}{\Delta t} = -v \frac{u_i^n - u_{i-1}^n}{\Delta x}$$

or solving for u^{n-1} we have

$$\begin{aligned} u_i^n \Delta x - u_{i-1}^n \Delta x &= -v u_i^n \Delta t + v u_{i-1}^n \Delta t \\ -u_{i-1}^{n-1} \Delta x &= -v u_i^n \Delta t + v u_{i-1}^n \Delta t - u_i^n \Delta x \\ -u_{i-1}^{n-1} &= -v u_i^n \frac{\Delta t}{\Delta x} + v u_{i-1}^n \frac{\Delta t}{\Delta x} - u_i^n \\ u_{i-1}^{n-1} &= v u_i^n \frac{\Delta t}{\Delta x} - v u_{i-1}^n \frac{\Delta t}{\Delta x} + u_i^n \end{aligned} \quad (3.2.3)$$

Taking $\alpha = v \frac{\Delta t}{\Delta x}$, we have;

$$(1 + \alpha) u_i^n + \alpha u_{i-1}^n = u_{i-1}^{n-1}, i = 1, 2, \dots, M$$

Note that we cannot now solve equation (3.2.4) explicitly for the solution at the advanced point, u_i^n , in terms of the solution at the base point u_i^{n-1} . Rather, equation (3.2.4) is *implicit* in u_i^n because u_{i-1}^n is also unknown; that is, we must solve equation (3.2.4) written for each grid point $i=1,2,3,\dots,M$ as a simultaneous system of *bidiagonal equations* (bidiagonal because each of equation (3.2.4) has two unknowns so that *simultaneous solution of the full set* of approximating algebraic equations is

required to obtain the complete numerical solution $u_1^n, u_2^n, \dots, u_M^n$. Thus, the solution of equation (3.2.4) is termed the *implicit Euler method*.

We could then naturally ask why use equation (3.2.4) when equation (3.2.2) is so much easier to use (explicit calculation of the solution at the next step in t of equation (3.2.2) vs. the implicit calculation of equation (3.2.4)). Well the answer is that the implicit calculation of equation (3.2.4) is often worthwhile because the *implicit Euler method has no stability limit* (is *unconditionally stable* in comparison with the explicit method with the stability limit stated in terms of the CFL condition). However, there is a price to pay for the improved stability of the implicit Euler method, that is, we must solve a *system of simultaneous algebraic equations*; equation (3.2.4) is an example. Furthermore, if the original ODE system approximating the PDE is nonlinear, we have to solve a *system of nonlinear algebraic equations* (equations (3.2.4) are linear, so the solution is much easier). The system of nonlinear equations is typically solved by a *variant of Newton's method* which can become very demanding computationally if the number of ODEs is large (due to the use of a large number of spatial grid points in the MOL approximation of the PDE, especially when we attempt the solution of two dimensional (2D) and three dimensional (3D) PDEs) (Hamdi et al., 2007; Schiesser & Griffith, 2009).

Additionally, although there is no limit for Δt with regard to stability for the implicit method, there is a *limit with regard to accuracy*. In fact, the first order upwind approximation of u_x in equations (2.2.5.1), equation (3.1.6), and the first order approximation of u_t in equation (2.2.5.1), equation (3.2.1) or (3.2.3), taken together limit the accuracy of the resulting FD approximation of equation (2.2.5.1). One way around this accuracy limitation is to use *higher order FD approximations* for the derivatives in equation (2.2.5.1).

For example, if we consider the second order approximation for u_x at i

$$u_x \approx \frac{u_{i+1} - u_{i-1}}{2\Delta x} + O(\Delta x^2) \quad (3.2.5)$$

substitution in equation (2.2.5.1) gives the MOL approximation of equation (2.2.5.1)

$$\frac{du_i}{dt} = -v \frac{u_{i+1} - u_{i-1}}{2\Delta x}, \quad 1 \leq i \leq M \quad (3.2.6)$$

We could then reason that if the integration in t is performed by the explicit Euler method, i.e., we use the approximation of equation (3.2.1) for $u_i = \frac{du_i}{dt}$, the resulting numerical solution should be more accurate than the solution from equation (3.2.2). In fact, the MOL approximation based on this idea

$$u_i^{n+1} = \frac{v\Delta t}{2\Delta x} (u_{i+1}^n - u_{i-1}^n), \quad i = 1, 2, \dots, M \quad (3.2.7)$$

is *unconditionally unstable*; this conclusion can be demonstrated by a *von Neumann stability analysis* that we will not cover here. This surprising result demonstrates that *replacing the derivatives in PDEs with higher order approximations does not necessarily guarantee more accurate solutions, or even stable solutions.*

Another vital note that was highlighted again by Schiesser & Griffith (2009) and Hamdi et al. (2007) is the numerical diffusion and oscillation in respect to the just solved example. That, Even if the implicit Euler method is used for the integration in t of equation (3.2.6) to achieve stability (or a more sophisticated explicit integrator in t is used that automatically adjusts Δt to achieve a prescribed accuracy), we would find the solution *oscillates* unrealistically. This numerical distortion is one of two generally observed forms of numerical error. The other numerical distortion is *diffusion* which would be manifest in the solution from equation (3.2.2). Briefly, the solution would exhibit excessive smoothing or rounding at points in x where the solution changes rapidly. This overall observation that, first-order approximation of u_x produces numerical oscillation is predicted by the *Godunov order barrier theorem*). To briefly explain, the order barrier is first order and any linear approximation, including FDs, above first order will be oscillatory. Equation (2.2.5.1) is an example of a difficult Riemann problem) if IC equation (3.1.3a) is discontinuous; for example, $u(x, t=0)=h(t)$ where $h(t)$ is the *Heaviside unit step function*. The (exact) analytical

solution is the initial condition function $f(x)$ of equation (3.1.3a) moving left to right with velocity v (from equation (2.2.5.1) and without distortion, that is, $u(x, t) = f(x - v t)$; however, the numerical solution will oscillate if u_x in equation (2.2.5.1) is replaced with a linear approximation of second or higher order.

Furthermore, we should also mention one point of terminology for FD approximations. The RHS of equation (3.2.5) is an example of a *centered approximation* since the two points at $i+1$ and $i-1$ are centered around the point i . Equation (3.1.6) is an example of a *noncentered, one-sided or upwind approximation* since the points i and $i-1$ are not centered with respect to i . Another possibility would be to use the points i and $i+1$ in which case the approximation of u_x would be *downwind* (for $v > 0$). Although this might seem like a logical alternative to equation (3.1.1) for approximating equation (2.2.5.1) at point i , the resulting MOL system of ODEs is actually unstable. Physically, for a convective system modeled by equation (2.2.5.1), we would be using information that is downwind of the point of interest (point i) and thus unavailable in a system flowing with positive velocity v . If v is negative, then using points i and $i+1$ would be upwinding (and thus stable). This illustrates the concept that the *direction of flow can be a key consideration in forming the FD approximation of a (convective or hyperbolic) PDE*. We can also show this centered approximation when we consider a power series of two functions such that

If

$$y(x+h) = y(x) + \frac{hy'(x)}{1!} + \frac{h^2 y''(x)}{2!} + \frac{h^3 y'''(x)}{3!} + \dots \quad (a)$$

$$y(x-h) = y(x) - \frac{hy'(x)}{1!} + \frac{h^2 y''(x)}{2!} - \frac{h^3 y'''(x)}{3!} + \dots \quad (b)$$

(a)+(b)

$$\Rightarrow y(x+h) + y(x-h) = 2y(x) + \frac{h^2 y''(x)}{2!} + \frac{h^2 y''(x)}{2!}$$

$$y(x+h) + y(x-h) = 2y(x) + h^2 y''(x) + \dots \quad (c)$$

Approximately, equation (c) can be written as

$$y_{n+1} + y_{n-1} = 2y_n(x) + h^2 y''_n + \dots$$

$$\frac{y_{n+1} - 2y_n + y_{n-1}}{h^2} = y''_n \quad (d)$$

Likewise (a)-(b)

$$\Rightarrow y(x+h) - y(x-h) = 2hy''(x)$$

$$y_{n+1} - y_{n-1} = 2hy'_n(x)$$

$$\frac{y_{n+1} - y_{n-1}}{h} = y'_n \quad (e)$$

So far we have considered only the MOL solution of first order hyperbolic PDEs, such as equation (2.2.5.1). We will now conclude this discussion of the MOL by considering a second order PDE, the parabolic equation (2.2.1.1). We have to first begin by finding the approximation for the second derivative u_{xx} . A commonly used *second order, central approximation* is (again, derived from the Taylor series, so the term $O(\Delta x^2)$ represents the truncation error)

$$u_{xx} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} + O(\Delta x^2) \quad (3.2.8)$$

Substituting equation (3.2.8) into equation (2.2.5.1) gives a system of approximating ODEs

$$u_{xx} \approx \alpha \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} \quad i = 1, 2, \dots, M \quad (3.2.9)$$

Subject to the IC (2.2.2.1) and the two BCs (2.2.2.2a) and (2.2.2.2b) the Equation (3.2.9) can be integrated and solve using any method, be it Runge-Kutta, explicit Euler method, the implicit Euler method, or any other higher order integrator for initial value ODEs. This is because, stability is not as much of a concern as with the previous hyperbolic PDEs (a characteristic of parabolic PDEs which tend to smooth solutions rather than hyperbolic PDEs which tend to propagate non-smooth conditions). However, stability constraints do exist for explicit methods. For example, for the explicit Euler method with a step Δt in t , the stability constraint is $\alpha \Delta t / \Delta x^2 < \text{constant}$ (so that as Δx is reduced to achieve better spatial accuracy in x , Δt must also be reduced to maintain stability).

Before proceeding with the integration of equation (3.2.9), we must include BCs (2.2.2.2a) and (2.2.2.2b). The Dirichlet BC at $x=x_b$, therefore, equation (2.2.2.2a), will merely be

$$u_1 = u_b \quad (3.2.10)$$

and therefore the ODE of equations (3.2.9) for $i=1$ is not required and the ODE for $i=2$ becomes

$$\frac{du_2}{dt} \approx \alpha \frac{u_3 - 2u_2 + u_1}{\Delta x^2} \quad (3.2.11)$$

Though equation (3.2.10) is algebraic and with equation (3.2.11) will turn to differential algebraic equation (DAE) system. Our target is to have a system of ODEs. Anyway, we can still achieve our aim. But before this, let look at this equation, at $i=M$ we have (3.2.9)

$$\frac{du_M}{dt} \approx \alpha \frac{u_{M+1} - 2u_M + u_{M+1}}{\Delta x^2} \quad (3.2.12)$$

Since we have u_{M+1} outside the grid in x that is $M+1$ is a fictitious point. As we mentioned earlier we can approximate the BC in equation (2.2.2.2b) so that equation

(3.2.12) can be integrated. This will make u_{M+1} to have an assigned value. Now using centered FD approximation of equation (3.2.5)

$$u_x \approx \frac{u_{M+1} - u_{M-1}}{2\Delta x} = 0$$

$$u_{M+1} = u_{M-1} \quad (3.2.13)$$

Substituting (3.2.13) in (3.2.12) we obtain

$$\frac{du_M}{dt} \approx \alpha \frac{u_{M-1} - 2u_M + u_{M-1}}{\Delta x^2} \quad (3.2.14)$$

And arrive at an ODE system (3.2.9) for $i=3, \dots, M-1$, equation (3.2.11) for $i=2$ and equation (3.2.4) for $i=M$

3.3 Origin of the name “Method of lines”

The name “Method of lines” (Schiesser & Griffith, 2009) was derived from integration of ODE system (say equation (3.2.9)) which produces the solutions $u_2(t)$, $u_3(t)$, ..., $u_M(t)$. (Note: $u_1(t)=u_b$, a constant from BC (2.2.2.2b)). We could then plot these functions in an x - $u(x,t)$ plane as a vertical line at each $x(i=2,3,\dots, M)$, with the height of each line equal to $u(x,t)$. In other words, the plot of the solution would be a set of vertical parallel lines suggesting the name *Method of lines*.

To illustrate this interpretation, let us consider a 1- dimensional (1D) heat conduction equation ;

$$u_t = u_{xx} \quad (3.3.1)$$

or
$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

- Dirichlet boundary condition at the left end, $u(x=0,t)=0$

- Neumann boundary condition at the right end, $u_x(x=l,t)=0$ (spatial domain $(0 \leq x \leq 1)$)
- Time domain $0 \leq t \leq 5.0$
- Initial condition $u(x, t=0) = \sin(\frac{\pi x}{2})$ (Schiesser & Griffith, 2009)

The MOL solution for the problem is shown in Fig (3.1). The numerical solution was obtained by MATLAB and MOL library routine dss042, 044, 046, 048 or 050 of Schiesser & Griffith (2009) (Program is listed in Appendix A) with number of grid points $M=31$ (so that the grid spacing is $[1-(0)]/31-1 = 0.03$).

$$u(x, t) = e^{-\left(\frac{\pi^2}{4}\right)t} \sin\left(\frac{\pi x}{2}\right) \quad (\text{Schiesser \& Griffith, 2009}) \quad (3.3.2)$$

The plotted output shown below clearly indicates the origin of *method of lines* (MOL).

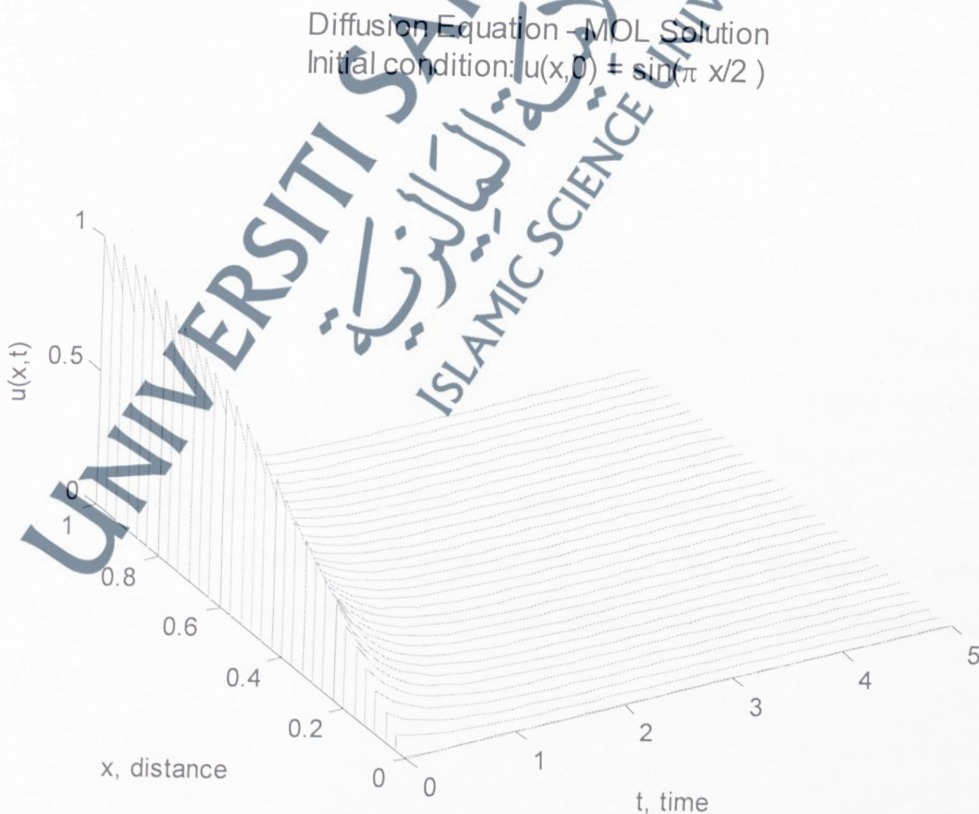


Figure 3.1 three dimensional graphical outputs from *Listing 3.0.1* illustrating the origin of MOL

3.4 Using Library Routine in MATLAB

Generally, when applying the MOL, integration of the approximating ODE/DAEs (e.g., equations (3.2.1) and (3.2.9)) is accomplished by using *library routines* for initial value ODE/DAEs. In other words, the explicit programming of the ODE/DAE integration (such as the explicit or implicit Euler method) is avoided; rather, an established integrator is used. This has the advantage that:

1. the detailed programming of the integration can be avoided, particularly the linear algebra (solution of simultaneous equations) required by an implicit integrator, so that the MOL analysis is substantially simplified, and
2. Library routines (usually written by experts) include features that make these routines especially effective (robust) and efficient such as *automatic integration step size adjustment* and the use of *higher order integration methods* (beyond the first order accuracy of the Euler methods); also, generally they have been thoroughly tested.

Thus, the use of quality ODE/DAE library routines is usually an essential part of MOL analysis. (Schiesser & Griffith, 2009; Hamdi et al., 2007).

Since the MOL essentially replaces the problem PDEs with systems of approximating ODEs, the addition of other ODEs is easily accomplished; this might occur, for example, if the BCs of a PDE are stated as ODEs. Thus, a mixed model consisting of systems of PDEs and ODEs is readily accommodated. Further, this idea can easily be extended to systems of ODE/DAE/PDEs. In other words, the MOL is a very flexible procedure for various combinations of algebraic and differential equations (and this flexibility generally is not available with other library software for PDEs) (Schiesser & Griffith, 2009; Hamdi et al., 2007).

3.5 Solving the Problem – Schrödinger equation – with MATLAB

Discussions about the Schrödinger equation and its derivation have been so far given in the previous chapters. And again, explanations (with examples) on how the Method of Lines can be applied on PDE have also been concluded. In the presiding discussion we are going to solve the Schrödinger equation using the method discussed in the above subsections. We shall also use the MATLAB library routines for a numerical MOL solution used by Schiesser & Griffith (2009) (though with some modifications) to find the numerical solution of the problem

Recall, from equation (2.1.3.7) of cubic Schrödinger equation. We rewrite this equation again for smoothness of our discussion thus;

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

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

Or in subscript notation

$$iu_t + u_{xx} + q|u|^2 u = 0 \quad (3.5.1)$$

Where u is complex dependent variable

x is a boundary-value (spatial) independent variable

t is initial-value independent variable

q is arbitrary parameter (constant) and

i is an imaginary complex number $\sqrt{-1}$

the term “Cubic” in CSE originate its name the term $q|u|^2 u = 0$, ($|u|$, denotes the absolute value of complex variable u).

Note that the formalities to be followed in solving the above equation (3.5.1) are already stated in the previous discussions. Following the steps one after the other, we have that;

First as mentioned earlier, we have to replace the spatial derivative u_{xx} with an algebraic approximation such of FD. But before this, for numerical treatment of nonlinear Schrödinger equation the complex $u(x,t)$ must be replaced by its real part $u(x,t)$ and imaginary part $\omega(x,t)$ ie

$$u(x,t) = v(x,t) + i\omega(x,t) \equiv u = [v\omega] \quad (3.5.2)$$

Substituting the new value of $u(x,t)$ into equation (3.5.1) we obtain;

$$i(v_t + i\omega_t) + v_{xx} + i\omega_{xx} + q|v + i\omega|^2(v + i\omega) = 0$$

$$\text{or } iv_t - \omega_t + v_{xx} + i\omega_{xx} + q(v^2 + \omega^2)(v + i\omega) = 0$$

Equating real and imaginary part to zero, we obtain the following nonlinear system of PDEs

$$v_t + \omega_{xx} + q(v^2 + \omega^2)\omega = 0 \quad (3.5.3a)$$

$$v_{xx} + q(v^2 + \omega^2)v - \omega_t = 0 \quad (3.5.3b)$$

Solving equations (3.5.3a) and (3.5.3b) we get;

For the (3.5.3a),

$$\frac{\partial v}{\partial t} + \frac{\partial^2 \omega}{\partial x^2} + q(v^2 + \omega^2)\omega = 0$$

$$\text{or } \frac{\partial v}{\partial t} = -q(v^2 + \omega^2)\omega - \frac{\partial^2 \omega}{\partial x^2} = 0 \quad (3.5.4a)$$

Also for the (3.5.3b),

$$\frac{\partial^2 v}{\partial x^2} + q(v^2 + \omega^2)v - \frac{\partial \omega}{\partial t} = 0$$

$$\text{or} \quad \frac{\partial \omega}{\partial t} = q(v^2 + \omega^2)v + \frac{\partial^2 v}{\partial x^2} \quad (3.5.4b)$$

Applying the three point central difference scheme to the equations (3.5.4a) and (3.5.4b) we get

From (3.5.4a),

$$\frac{\partial^2 \omega}{\partial x^2} = \frac{\omega_{n+1} - 2\omega_n + \omega_{n-1}}{h^2} \quad (3.5.5a)$$

And from (3.5.4b),

$$\frac{\partial^2 v}{\partial x^2} = \frac{v_{n+1} - 2v_n + v_{n-1}}{h^2} \quad (3.5.5b)$$

h is the spacing between the discretized line and n is the number of grids.

Substituting equations (3.5.5a) and (3.5.5b) in equations (3.5.4a) and (3.5.4b) we get;

$$\frac{dv_i}{dt} = -q(v_n^2 + \omega_n^2)\omega_n - \left(\frac{\omega_{n+1} - 2\omega_n + \omega_{n-1}}{h^2} \right) \quad (3.5.6a)$$

$$\frac{d\omega_i}{dt} = q(v_n^2 + \omega_n^2)v_n - \left(\frac{v_{n+1} - 2v_n + v_{n-1}}{h^2} \right) \quad (3.5.6b)$$

The question we may ask our selves is whether equations (3.5.6a) and (3.5.6b) going to make any sense or impact in the choice of our BCs and ICs. Thus, we shall see here in our discussion.

Now for the next step is that; equation (3.5.1) is first order in t and second order in x , it therefore requires 1 (one) IC and 2 (two) BCs respectively. The IC is taken from the analytical solution (with $q=1$) (Schiesser & Griffith).

$$u(x, t) = \sqrt{2}e^{i(0.5x+0.75t)} \operatorname{sech}(x-t) \quad (3.5.7a)$$

Since we already mentioned that we have to analyze $u(x, t)$ as a two real functions, we then therefore say from equation (3.5.7a)

$$|u(x, t)| = \sqrt{2} \operatorname{sech}(x-t) \quad (3.5.7b)$$

We will use equation (3.5.7b) for comparison with analytical solution. Then with $t=0$ in equation (3.5.7a), the IC is

$$u(x, t=0) = \sqrt{2}e^{i(0.5x+0)} \operatorname{sech}(x-0)$$

$$u(x, t=0) = \sqrt{2}e^{i(0.5x)} \operatorname{sech}(x) \quad (3.5.8)$$

Ideally we are suppose to consider the two BCs, but this not going to be possible since for a class of rapidly decreasing initial conditions i.e. $|u(x, t)| \rightarrow 0$ as $|x| \rightarrow +\infty$ that, if PDE is analyzed over an essential infinite domain $-\infty \leq x \leq \infty$ and if changes in the solution occur only over a finite interval in x , the BCs at infinite 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 PDE solution (Samrout, 2007; Schiesser & Griffith, 2009).

Consequently, equations (3.5.1) and (3.5.8) constitute the coupled PDE problem, equations (3.5.7a) and (3.5.7b) is used in subsequent programming and analysis to evaluate the numerical MOL solution. Note that equation (3.5.7a) and (3.5.7b) are traveling wave solution since the argument of *sech* function $x-t$ (Schiesser & Griffith, 2009).

As already stated, we will consider some MATLAB routines for numerical MOL solution of (3.5.1) and (3.5.8) that were used by Schiesser & Griffith (2009). The program is “main program”, `pde_1_main` which calls the `dss` library 044 with a listing 6.1. Further more, full detail on the subroutines called by main program is given in **Appendix B**. For the purpose of preparing the values of parameters required by the DSS library in the MATLAB, we explain in the following program the step by step parameter assignment for different set of cases in the experiment. It is crucial to notice the changes in the parameters involved and that is why we include this listing in the text rather than in appendix. Thus the program is given below with the solution;

Listing 3.1: program solution of CSE when the discretization size $\Delta x = 0.167$ (MOL grid has 301), spatial domain is $-10 \leq x \leq 40$ and step size is $0 \leq t \leq 30$.

```
% 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=301;
%
% 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);
    end
%
% 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,'o',x,v2w2_anal,'-')
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 3.2: program solution of CSE when the discretization size $\Delta x = 0.2$ (MOL grid has 251), spatial domain is $-10 \leq x \leq 40$ and step size is $0 \leq t \leq 30$;

```

% 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=init1_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);
    end
end
%
% 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,'o',x,v2w2_anal,'-')
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 3.3: program solution of CSE when the discretization size $\Delta x = 0.25$ (MOL grid has 201), spatial domain is $-10 \leq x \leq 40$ and step size is $0 \leq t \leq 30$;

```

% 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=301;
%
% Initial condition
t0=0.0;
u0=init1_1(t0);
%
% Independent variable for ODE integration
tf=30.0;
tout=(t0:5.0:tf)';
nout=7;
ncall=0;
%
% ODE integration
mf=1;
reitol=1.0e-06; abstol=1.0e-06;
options=odeset('RelTol',reitol,'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);
    end
%
% 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,'o',x,v2w2_anal,'-')
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 3.4: program solution of CSE when the discretization size $\Delta x=0.167$ (MOL grid has 301), spatial domain is $-10 \leq x \leq 40$ and step size is $0 \leq t \leq 32$:

```

% 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=301;
%
% Initial condition
t0=0.0;
u0=inital_1(t0);
%
% Independent variable for ODE integration
tf=32.0;
tout=(t0:5.0:tf);
nout=7;
ncallf=0;
%
% ODE integration
mf=1;
reitol=1.0e-06; abstol=1.0e-06;
options=odeset('RelTol',reitol,'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);
    end
%
% Output in the neighborhood of the soliton peak
if(abs(x(i)-t(it))<=1.0)
    fprintf('%5.2f%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,'o',x,v2w2_anal,'-')
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 3.5: program solution of CSE when the discretization size $\Delta x = 0.2$ (MOL grid has 251), spatial domain is $-10 \leq x \leq 40$ and step size is $0 \leq t \leq 32$

```

% 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=inita1_1(t0);
%
% Independent variable for ODE integration
tf=32.0;
tout=(t0:5.0:tf)';
nout=7;
ncall=0;
%
% ODE integration
mf=1;
re1tol=1.0e-06; abstol=1.0e-06;
options=odeset('RelTol',re1tol,'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);
    end
%
%   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,'o',x,v2w2_anal,'-')
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 3.6: program solution of CSE when the discretization size $\Delta x = 0.25$ (MOL grid has 201), spatial domain is $-10 \leq x \leq 40$ and step size is $0 \leq t \leq 32$;

```

% 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=201;
%
% Initial condition
t0=0.0;
u0=initial_1(t0);
%
% Independent variable for ODE integration
tf=32.0;
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);
    end
%
% 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,'o',x,v2w2_anal,'-')
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 3.7: program solution of CSE when the discretization size $\Delta x = 0.167$ (MOL grid has 301), spatial domain is $-10 \leq x \leq 40$ and step size is $0 \leq t \leq 35$;

```

% 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=301;
%
% Initial condition
t0=0.0;
u0=inital_1(t0);
%
% Independent variable for ODE integration
tf=35.0;
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);
    end
end
%
%   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
end

```



```

fprintf(' ncall = %4d\n\n',ncall);
%
% Plot numerical and analytical solutions
figure(1)
plot(x,v2w2,'o',x,v2w2_anal,'-')
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 3.8: program solution of CSE when the discretization size $\Delta x = 0.2$ (MOL grid has 251), spatial domain is $-10 \leq x \leq 40$ and step size is $0 \leq t \leq 35$;

```

% 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=initial_1(t0);
%
% Independent variable for ODE integration
tf=35.0;
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      /');
    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);
    end
end
%
% 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,'o',x,v2w2_anal,'-')
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 3.9: program solution of CSE when the discretization size $\Delta x = 0.25$ (MOL grid has 201), spatial domain is $-10 \leq x \leq 40$ and step size is $0 \leq t \leq 30$;

```

% 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=201;
%
% Initial condition
t0=0.0;
u0=inita1_1(t0);
%
% Independent variable for ODE integration
tf=35.0;
tout=(t0:5.0:tf);
nout=7;
ncall=0;
%
% ODE integration
mf=1;
re1tol=1.0e-06; abstol=1.0e-06;
options=odeset('RelTol',re1tol,'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);
    end
%
% 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,'o',x,v2w2_anal,'-')
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

```

As for the invariants discussed in the program, its integral for the single soliton are (Schiesser & Griffith, 2009);

$$u_1(t) = \int_{-\infty}^{+\infty} |u(x,t)|^2 dx \quad (3.5.9a)$$

$$u_2(t) = \int_{-\infty}^{+\infty} i(\overline{u}u_x - u\overline{u}_x)dx = \int_{-\infty}^{+\infty} i[(v+i\omega)(v_x-i\omega_x) - (v-i\omega)(v_x+i\omega_x)]dx$$

$$u_2(t) = \int_{-\infty}^{+\infty} i[(vv_x + i\omega v_x - iv\omega_x + \omega\omega_x) - (vv_x - i\omega v_x + iv\omega_x + \omega\omega_x)]dx$$

$$u_2(t) = \int_{-\infty}^{+\infty} (v\omega_x - \omega v_x)dx \quad (3.5.9b)$$

$$u_3(t) = \int_{-\infty}^{+\infty} \left[|u_x|^2 - \frac{1}{2}q|u|^4 \right] dx \quad (3.5.9c)$$

The computed values of these invariants (in all the programs) are also tabulated and discussed subsequently.