NUMERICAL METHODS FOR SOLVING SCHRÖDINGER'S EQUATION

MOHAMED TAWHID

A Thesis

in

The Department

of

Mathematics and Statistics

Presented in Partial Fulfillment of Requirements

for the Degree of Master of Science at

Concordia University

Montreal, Quebec, Canada

November 1995

ISBN 0-612-18467-6

Abstract

Numerical methods for solving Schrödinger's equation

Mohamed Tawhid

Two kinds of numerical problems arising from the Schrödinger equation are studied: the eigenvalue (or Sturm-Liouville) problem and the initial value problem (known also as the Cauchy problem). Algorithms for the Cauchy problem allow the second problem to be solved numerically by the use (for example) of a shooting procedure. In other words, the knowledge of algorithms for the Cauchy (sub) problem is of great importance. Numerical methods for first order differential equations are surveyed. Schrödinger's equation is then broken into a system two first order differential equations which are solved numerically. The most effective methods chosen by physicists and numerical analysts for the numerical solution of Schrödinger's equation are studied. Two computer programs for this purpose are developed. The first program treats Schrödinger's equation as an initial value problem and is applicable for any symmetric potential. This program is applied to some specific examples, namely: the harmonic oscillator, sech–squared and finite square well potentials. In the second program Schrödinger's equation is treated as a boundary value problem, in which case the computer program is applied to the Woods–Saxon potential. In the calculations of the eigenvalues, the dependence of the errors on the integration step size is studied.

Acknowledgements

It is my pleasure to express my gratitude to all the people without whose support I would not have been able to finish not only this work but also my degree.

I am deeply indebted to my supervisor Dr. R L. Hall for his excellent guidance, support and generous encouragement. I really appreciate his kindness and profound knowledge.

I am indebted to my professors Dr. S. T. Ali, Dr. A. B. von Keviczky and Dr. M. Gabr (Alexandria University) for their invaluable assistance and constant advice.

Dr. M. Elalem, Dr. A. Elbakry and Dr. M. Radwan (Alexandria University) deserve a special thanks for their tireless encouragement.

A special thanks also to my professor Dr. E. J. Doedel (Computer Science) for his valuable help in the two courses he taught me.

Finally, I wish to express my sincere gratitude and appreciation to my family: for their love, understanding and undying support through the long days and snowy nights during my term of study at Concordia University.

To my parents

Contents

List of Tables

Introduction

Many problems in areas of application, such as pollution of the atmosphere, theoretical chemistry, astrophysics, laser physics, nuclear reaction, etc. reduce to a system of second order ODEs of Schrödinger form. Schrödinger's equation is the fundamental equation of nonrelativistic quantum mechanics which involves ordinary differential equations of second order in which the first derivative does not appear explicitly. Its importance for physicists is obvious and for numerical analysts it is often taken as an eloquent test equation for the comparison of the quality of various numerical approaches [1–5]. Therefore many authors have given intense interest to it over many years. The time-independent radial Schrödinger equation is

$$\left[\frac{d^2}{dr^2} - f(r)\right] y(r) = 0, \quad f(r) = u(r) - E,$$

$$u(r) = \left(\frac{2m}{\hbar^2}\right)\left[V(r) + \frac{\ell(\ell+1)}{r^2}\right], \qquad E = \left(\frac{2m}{\hbar^2}\right)\varepsilon, \tag{0.1}$$

where $\hbar$ is the Planck's constant, ε is the particle energy and m is the mass of the particle in the potential $V(r)$ [6–8]. It is important to note that the one-dimensional Schrödinger equation can be obtained by putting $\ell = -1$ in Eq.(0.1). For a few limited forms of simple potentials energy functions the one-dimensional Schrödinger equation can be solved analytically [9–12]. However, in most cases one would use numerical methods or some rough approximations.

Numerical methods for obtaining bound-state solutions of the one-dimensional Schrödinger equation may be classified broadly into two types: numerical integration (shooting methods) [13-21] and matrix methods [22-30].

In general, numerical integration of small sets of coupled equations has become feasible on microcomputers [17]. The solution of large sets remains a time-consuming, expensive and difficult operation. It is of considerable interest to find the most efficient method. However, for many applications solutions are often not required to be of very high accuracy and it is possible to design or accept methods that work very quickly for low accuracy, although they may lose their advantages for higher accuracy [77].

The integration process may start from both ends of the interval of integration with matched slopes at the meeting point, as Cooley [13] has done in solving Schrödinger's equation numerically. The eigenvalue was found by extrapolation or by successive trials. This method was discussed further and extended [14-15]. Integration may start at a point inside the interval and go in two opposite directions as was done by Hajj at el [16]. Hajj [17] found the eigenvalue by using only outward integration, this method has been discussed by Killingbeck [31-36].

Because Eq.(0.1) has been studied so extensively, many numerical methods have been proposed for its solution. Two of these methods are very popular, namely those of Numerov and De Vogelaer. Numerov's method is the optimal linear two step method [37]. Although Numerov's method is only of order 4, it has been found in practice to be often superior to higher order methods. However, Cash at el [38] have suggested a class of Runge-Kutta methods for numerical integration with a fixed step length which are superior to Numerov's method. For the error analysis of these methods see Ref. [39]. Raptis at el [40] developed a two-step method to approximate the solution of Schrödinger's equation via exponential functions.

Other authors have extended the work of Raptis at el and reported improved algorithms for the solution of the Schrödinger equation [41–45].

De Vogelaere's method enjoys wide popularity among physicists who are interested in solving the Schrödinger equation [46-50], because of its simple initialization as well as flexibility in allowing changes in stepsize during the integration.

For the discussion of the error analysis for De Vogelaere's method see [51–52]. Chandra [53] has published a computer program utilizing De Vogelaere's method to solve the differential equations arising in a close-coupling formulation of quantum mechanical scattering problems. Chandra's program makes no attempt to monitor the local truncation error and leaves the choice of the initial step length to the user. Coleman at el [54] also published a computer program using De Vogelaere's method to solve the single-channel Schrödinger equation for the scattering of an electron by the static potential of atomic hydrogen for specific energy and angular momentum. Their program automatically chooses the step-lengths in accordance with a local accuracy criterion supplied by the user.

A noniterative method for solving the one-dimensional eigenvalue problem was devised by Truhlar [25]. The method is based on treating the problem as a boundary value problem. For the application of this method to multidimensional eigenvalue problems see Refs. [55–57].

The contribution of physicists has also stimulated the introduction of numerical methods to problems originating in various branches of mathematical physics; specifically, perturbation theory and the method of potential envelopes. The method of potential envelopes gives us simple formulae for bounds to the eigenvalues [58–61] and has had applications to a variety of physical problems [61–68].

Perturbation theory was thought of for a long time as an efficient tool to describe physical phenomena in terms of analytic formulae thus postponing (or avoiding at least) a numerical approach. The complementary attitude, is to use perturbation theory as the very tool for computation itself [69-87].

Gordon [77] developed very efficient algorithms for the solution of Eq.(0.1). Since then, important progress has been made on their algorithms [70,76,78,79], and several problems of quantum mechanics and mathematical physics were solved [69,72,80,81]. Gordon [77] was the first to exploit perturbation theory in order to improve the solution furnished by this approach.

A perturbative numerical method (PNM) is suggested in each of the Refs. [70,74,77–79] . Two classes of PNM were examined. To distinguish between them, we shall note that the perturbative series giving $y(r)$ and $y'(r)$ are uniquely determined by the choice of the approximating function. Thus PNM in Refs. [71,78] utilizes step functions PNM [SF-PNM], while in Refs. [77,79] piecewise continuous linear functions PNM [PCLF-PNM] are used. Comparisons have been reported (for scattering problems) between an SF-PNM and the well-known Numerov method [78], as well as between the PCLF-PNM of Gordon and the same Numerov method [37]. The authors of these studies conclude that, in general, the Numerov method would be preferable to PNM. Adam at el [82] derived an improved SF-PN algorithm via first order perturbation theory, then compared these algorithms with the Numerov's method and concluded that this algorithm in the region of very small stepsize shows stability of the computed eigenvalues against round off error (in contrast with the instability of Numerov's method).

Our objective in this thesis is to study numerical methods for solving Schrödinger's equation. We organize our thesis as follows:

In Chapter I we review some basic ideas such as the difference approximation in which one uses a computer to transform a differential equation into recurrence relations or matrix calculations. In Chapter II we discuss some numerical methods for solving the initial value problem. We also show how a boundary value problem like (Schrödinger's equation) can be reduced to a sequence of initial value problems, which (in turn) are amenable to the numerical techniques studied previously. We further examine the extrapolation technique for increasing the accuracy of approximate solutions and we apply this to study the truncation error arising from various numerical methods.

In Chapter III we investigate how Schrödinger's equation is considered both as a IVP and as a BVP. We briefly examine the most popular numerical methods such as Numerov's method and de Vogelaere's method. We also describe some specific contributions of mathematical physics such as the method of potential envelopes and perturbation theory. In Chapter IV we develop some computer programs to treat Schrödinger's equation numerically, first as a IVP and second as a BVP. In both cases we give some examples to examine the efficiency of the programs. We give as a result some tables to indicate the efficiency of the programs and highlight the importance of choosing the appropriate step size. The relationship between the error in the final eigenvalue approximation and the local integration error is explored and some useful rules of thumb are proposed.

Chapter I

Preliminaries

In this chapter we shall review some basic facts concerning ordinary differential equation (ODEs), that will be useful later. In Section I.1 we describe what is the initial value problem (IVP) for first-order ODE as well as for a system of n ODEs of first order. This will guide us in treating Schrödinger's equation numerically by breaking it into two first order ODEs. In Section I.2 we show how the boundary value problem (BVP) can be reduced to a sequence of initial value problems. In other words, we can apply the numerical techniques of solving initial value problems to solve boundary value problems. In Section I.3 we mention a difference approximation which represents a way to translate problems of calculus into discrete numerical ones which the computer can handle. Difference approximations help us to derive multi-step methods and to study the error arising from numerical methods.

I.1. Generalities and Definitions

The equation

$$y' = f(x, y) \tag{I.1}$$

is called an *ordinary differential equation* of the first order, where $f(x, y)$ is a real valued function of two variables defined for all real y and real $x \in [a, b]$. If any real valued differentiable function $y(x)$ satisfies Eq.(I.1) for $x \in [a, b]$ then we say

$y(x)$ is solution. This solution in general is not unique, one of the reasons being the arbitrariness of the constant of integration. If we want to make this solution unique in terms of the constant of integration, then we must impose a condition on the solution to make it assume value y_0 at a particular point $\gamma \in [a, b]$, namely

$$y(\gamma) = y_0. \tag{I.2}$$

Eq.(I.1) and Eq.(I.2) are called an *initial value (or Cauchy) problem IVP*, Eq.(I.2) is called an *initial condition* and y_0 is the initial value of y.

For practical reasons we will take γ as the left-hand end of the domain of the equation - i.e. $\gamma = a$; thus Eq.(I.2) becomes

$$y(a) = y_0. \tag{I.3}$$

Our interest is in a system of first-order differential equations, especially a couple of first-order ODEs. The system is formulated as follows.

Consider

$$y^{i\prime} = f^i(x, y^1, y^2, \cdots, y^n) \qquad (1 \leq i \leq n), \quad x \in [a, b], \tag{I.4}$$

which represents a system of n first-order ODEs. Here $f^1, f^2, \cdots, f^n$ are known real functions of their arguments $x, y^1, y^2, \cdots, y^n$. The initial condition for given $\gamma \in [a, b]$ are given by

$$y^i(\gamma) = y_0^i, \tag{I.5}$$

where $y^1, y^2, \cdots, y^n$ satisfy Eq.(I.4). Let us write Eq.(I.4) and Eq.(I.5) in vector form

$$\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}), \qquad \mathbf{y}(\gamma) = \mathbf{y}_0, \tag{I.6}$$

where $\mathbf{y}, \mathbf{f}$, and $\mathbf{y}_0$ are real n-dimensional vectors

$$\mathbf{y} = \begin{pmatrix} y^1 \\ y^2 \\ \cdots \\ y^n \end{pmatrix}, \quad \mathbf{f}(x, \mathbf{y}) = \begin{pmatrix} f^1(x, y^1, y^2, \cdots, y^n) \\ f^2(x, y^1, y^2, \cdots, y^n) \\ \cdots \\ f^n(x, y^1, y^2, \cdots, y^n) \end{pmatrix}, \quad \mathbf{y}_0 = \begin{pmatrix} y_0^1 \\ y_0^2 \\ \cdots \\ y_0^n \end{pmatrix}. \tag{I.7}$$

We will concentrate on the case when $\gamma = a$, that is to say

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}), \ \mathbf{y}(a) = \mathbf{y}_0. \tag{I.8}$$

I.2. The Boundary Value Problem (BVP)

In this section we will prove that any boundary value problem can be reduced to a sequence of initial value problems. In other words, the techniques of solving initial value problems can be used to solve any boundary value problem.
Let us define the simplest BVP, which can be written as

$$y'' = f(x, y) \qquad (x \in [a, b]), \qquad y(a) = C, \qquad y(b) = D, \tag{I.9}$$

where C and D are constants. The existence of solutions of Eq.(I.9) depends not only the properties of $f(x, y)$ but also on the values of constants C and D. One interesting way to solve a BVP numerically is to reduce it to a sequence of IVP - i.e. we refer to Eq.(I.9) to which the following IVP

$$y'' = f(x, y) \qquad (x \in [a, b]), \qquad y(a) = C, \qquad y'(a) = \gamma \tag{I.10}$$

is associated. Here, $f(x, y)$ and C are the same as in Eq.(I.9), while γ is an arbitrary parameter. The solution $y(x, \gamma)$ of Eq.(I.10), computed by any of the methods in

Chapter II, is also the solution of BVP Eq.(I.9) only if $y(b,\bar{\gamma}) = D$ for some value $\bar{\gamma}$ of γ. In other words, $y(x,\gamma)$ is a solution of Eq.(I.9) when $\gamma = \bar{\gamma}$, where $\bar{\gamma}$ is the root of the equation

$$V(\gamma) \equiv y(b,\gamma) - D = 0.$$

By trying successive values $\gamma_0, \gamma_1, \gamma_2, \cdots$ we construct the corresponding values $V(\gamma_0), V(\gamma_1), V(\gamma_2), \cdots$ until it is observed that $V(\gamma_i)$ and $V(\gamma_{i+1})$ have different signs, then $\bar{\gamma}$ lies somewhere between γ_i and γ_{i+1}. By using some standard rootfinder technique as for example bisection or regula falsi [88,89], we can determine $\bar{\gamma}$ with the desired accuracy. However, if the computation indicates that $V(\gamma)$ has no root, then Eq.(I.9) may have no solution.

The technique which uses a numerical method for IVP plus some rootfinder technique is known as a *shooting technique*. Actually this approach is used by many authors to treat Schrödinger' s equation as a BVP.

I.3. Difference Approximation [90]

We consider three different approaches for deriving *difference approximations*. The first uses a *difference operator*, the second is based on a *Taylor expansion* of the function about a grid point and the third involves an *interpolating polynomial*. Let us briefly describe the three approaches.

I.3.1 Difference Operators [9,90]

We define the following three difference operators; in particular

$$\Delta f_i = f_{i+1} - f_i, \ \nabla f_i = f_i - f_{i-1}, \ \delta f_i = f_{i+\frac{1}{2}} - f_{i-\frac{1}{2}},$$

which are called the forward, backward and central difference operators respectively. We further note

$$\delta f_{i+\frac{1}{2}} = f_{i+1} - f_i, \qquad f_{i+\frac{1}{2}} = f(x_i + \frac{1}{2}h).$$

The higher n-th order difference operators may be written as powers of the preceeding difference operators

$$\Delta^n f = \Delta^{n-1}(\Delta f) = \Delta^{n-1}(f_{i+1} - f_i),$$

$$\nabla^n f = \nabla^{n-1}(\nabla f) = \nabla^{n-1}(f_i - f_{i-1}),$$

$$\delta^n f = \delta^{n-1}(\delta f) = \delta^{n-1}\left(f_{i+\frac{1}{2}} - f_{i-\frac{1}{2}}\right).$$

I.3.2 Taylor Expansion [89,91,92]

We derive the *difference approximation* for $f_i' = f'(x_i)$ using $f_i = f(x_i)$ and $f_{i+1} = f(x_{i+1})$, where $x_{i+1} = x_i + h$. The f values of all data points other than x_i are expanded in Taylor series. The Taylor expansion of f_{i+1} about x_i is

$$f_{i+1} = f_i + hf_i' + \frac{1}{2}h^2 f_i'' + \frac{1}{6}h^3 f_i''' + \frac{1}{24}h^4 f_i^{(4)} + \cdots, \tag{I.11}$$

from which

$$f_i' = \frac{f_{i+1} - f_i}{h} - \frac{1}{2}hf_i'' - \frac{1}{6}h^2 f_i''' - \cdots \tag{I.12}$$

follows. Truncating the first term of Eq.(I.12) yields the *forward difference approximation**

$$f_i' = \frac{f_{i+1} - f_i}{h} + O(h), \tag{I.13}$$

where the error is approximately $-\frac{1}{2}hf_i''$ - i.e. approximately proportional to the grid interval h. The *backward difference approximation* is similarly derived. From the Taylor expansion of f_{i-1},

$$f_{i-1} = f_i - hf_i' + \frac{1}{2}h^2 f_i'' - \frac{1}{6}h^3 f_i''' + \frac{1}{24}h^4 f_i^{(4)} + \cdots, \tag{I.14}$$

---

* To say that some quantity f is $O(h^p)$ as $h \to 0$ means that there is a number C independent of h such that $|f| \leq C|h^p|$.

we solve for f_i' and thereby obtain the *backward difference approximation*

$$f_i' = \frac{f_i - f_{i-1}}{h} + O(h), \tag{I.15}$$

where the error is approximately $-\frac{1}{2}hf_i''$ - i.e. approximately proportional to the grid interval h. The *central difference approximation* using f_{i+1} and f_{i-1} may be derived by the Taylor expansion of f_{i+1} and f_{i-1} already given - i.e. as Eq.(I.11) and Eq.(I.14) respectively. Subtracting Eq.(I.11) from Eq.(I.14) leads to

$$f_{i+1} - f_{i-1} = 2hf_i' + \frac{1}{3}h^3 f_i''' + \cdots$$

where the f_i'' term has been eliminated in the process of solving for f_i'. Thus we get the *central difference approximation*

$$f_i' = \frac{f_{i+1} - f_{i-1}}{2h} + O(h^2). \tag{I.16}$$

The difference approximation for f_i' needs at least two points. In general, the smallest number of data points necessary to derive a difference approximation of order p is $p + 1$.

I.3.3 Interpolation Polynomial [88,93,94]

Let us explain this part in more detail because we shall use it to derive the multi-step method. The divided difference of f with respect to $x_0, x_1, \cdots, x_n$ is derived by showing that P_n (the nth Lagrange polynomial agreeing with the function f at distinct numbers $x_0, x_1, \cdots, x_n$) has representation

$$P_n(x) = \alpha_0 + \alpha_1(x - x_0) + \alpha_2(x - x_0)(x - x_1) + \cdots$$

$$+\alpha_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}) \tag{I.17}$$

for appropriate constants $\alpha_0, \alpha_1, \cdots, \alpha_n$. To determine the $\alpha's$, we set $x = x_0$ in Eq.(I.14), which yields

$$\alpha_0 = P_n(x_0) = f(x_0), \tag{I.18}$$

whereas setting $x = x_1$ in Eq.(I.17) leads to

$$\alpha_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}. \tag{I.19}$$

Let us denote the 0-th divided difference of the function with respect to x_i by $f[x_i]$, where $f(x_i) = f[x_i]$, and the other divided difference are defined inductively. The first divided difference of f with respect to x_i and x_{i+1}, denoted by $f(x_i, x_{i+1})$, is defined as

$$f(x_i, x_{i+1}) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i},$$

whereas the k-th divided difference relative to $x_i, x_{i+1}, \cdots, x_{i+k}$ is

$$f(x_i, x_{i+1}, \cdots, x_{i+k-1}, x_{i+k}) = \frac{f(x_{i+1}, \cdots, x_{i+k}) - f(x_i, \cdots, x_{i+k-1})}{x_{i+k} - x_i}. \tag{I.20}$$

Therefore, we may write Eq.(I.19) as $\alpha_1 = f(x_0, x_1)$ and similarly obtain $\alpha_2, \cdots, \alpha_k$ - i.e. $\alpha_k = f(x_0, x_1, \cdots, x_k)$. Thus Eq.(I.14) can be written as

$$\begin{aligned} P_n &= f(x_0) + (x - x_0)f(x_0, x_1) + (x - x_0)(x - x_1)f(x_0, x_1, x_2) + \cdots \\ &\qquad \cdots + (x - x_0)(x - x_1)\cdots(x - x_{n-1})f(x_0, x_1, \cdots, x_n) \\ &= f(x_0) + \sum_{k=1}^{n}(x - x_0)\cdots(x - x_{k-1})f(x_0, x_1, \cdots, x_k), \end{aligned} \tag{I.21}$$

which is known as *Newton's interpolatory divided difference formula.*

By arranging the order $x_0, \cdots, x_n$ consecutively with equal spacing $h = x_{i+1} - x_i$ $(0 \leq i \leq n-1)$, writing $x = x_0 + \beta h$ and noting $x - x_i = (\beta - i)h$, Eq.(I.21) is rewritten as

$$P_n(x) = P_n(x_0 + \beta h) = f(x_0) + \beta h f(x_0, x_1) + \beta(\beta - 1)h^2 f(x_0, x_1, x_2) + \cdots$$

$$\cdots + \beta(\beta - 1)(\beta - n + 1)h^n f(x_0, x_1, \cdots, x_n)$$

$$= \sum_{k=0}^{n} \beta(\beta - 1)(\beta - k + 1)h^k f(x_0, \cdots, x_k) \tag{I.22}$$

with binomial coefficients

$$\binom{\beta}{k} = \frac{\beta!}{k!(\beta - k)!} = \frac{\beta(\beta - 1)\cdots(\beta - k + 1)}{k!}. \tag{I.23}$$

Thereafter Eq.(I.22) becomes

$$P_n(x) = P_n(x_0 + \beta h) = \sum_{k=0}^{n} \binom{\beta}{k} k! h^k f(x_0, \cdots, x_k), \tag{I.24}$$

which equation is referred to in the literature as the *Newton forward divided difference formula.*

To derive *Newton forward difference formula*; we write Eq.(I.20) in terms of the forward difference operator as

$$f(x_0, x_1) = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{1}{h}\Delta f(x_0)$$

$$f(x_0, x_1, x_2) = \frac{1}{2h}\left[\frac{\Delta f(x_1) - \Delta f(x_0)}{h}\right] = \frac{1}{2h^2}\Delta^2 f(x_0),$$

which is generalized to

$$f(x_0, \cdots, x_k) = \frac{1}{k! h^k}\Delta^k f(x_0).$$

Consequently, Eq.(I.24) becomes *Newton forward-difference formula*

$$P_n(x) = \sum_{k=0}^{n} \binom{\beta}{k} \Delta^k f(x_0).$$

To derive *Newton backward-difference formula*, we reorder the interpolating nodes as $x_n, \cdots, x_0$, so that

$$P_n(x) = f(x_n) + (x - x_n) f(x_{n-1}, x_n) + (x - x_n)(x - x_{n-1}) f(x_{n-2}, x_{n-1}, x_n)$$

$$+ \cdots + (x - x_n) \cdots (x - x_1) f(x_0, \cdots, x_n) \tag{I.25}$$

Using equal spacing with $x = x_n + \beta h$ and $x - x_i = (\beta + n - i)h$ we get

$$P_n(x) = P_n(x_n + \beta h) = f(x_n) + \beta h f(x_{n-1}, x_n) + \beta(\beta - 1) h^2 f(x_{n-2}, x_{n-1}, x_n) + \cdots$$

$$+ \beta(\beta - 1) \cdots (\beta + n - 1) h^n f(x_0, x_1, \cdots, x_n) \tag{I.26}$$

Eq.(I.26) is called the *Newton backward divided difference formula.* By means of Eq.(I.20) and backward difference operator

$$f(x_{n-1}, x_n) = \frac{1}{h} \nabla f(x_n), \qquad f(x_{n-2}, x_{n-1}, x_n) = \frac{1}{2h^2} \nabla^2 f(x_n).$$

In general

$$f(x_{n-k}, \cdots, x_{n-1}, x_n) = \frac{1}{k! h^k} \nabla^k f(x_n),$$

$$\begin{aligned} P_n(x) \;=\; & f(x_n) + \beta \nabla f(x_n) + \frac{\beta(\beta+1)}{2} \nabla^2 f(x_n) + \cdots \\ & \cdots + \frac{\beta(\beta+1) \cdots (\beta+k-1)}{k!} \nabla^k f(x_n). \end{aligned}$$

Using the generalized form of the binomial coefficients

$$\binom{\beta + k - 1}{k} = \frac{(\beta + k - 1)(\beta + k - 2)\cdots(\beta + 1)\beta}{k!}$$

with $\binom{\beta-1}{0} = 1$, $\binom{\beta}{1} = \beta$ and $\binom{\beta+1}{2} = \frac{\beta(\beta+1)}{2!}$ gives

$$\begin{aligned} P_n(x) &= f(x_n) + \binom{\beta}{0}\nabla f(x_n) + \binom{\beta+1}{2}\nabla^2 f(x_n) + \cdots \\ &\qquad \cdots + \binom{\beta + k - 1}{k}\nabla^k f(x_n) \\ &= \sum_{j=0}^{k} \binom{\beta + j - 1}{j}\nabla^j f_n \qquad -k \le \beta \le 0, \end{aligned} \tag{I.27}$$

which is known as the *Newton difference formula.*

Note that the equivalence of the forward and backward differences $\nabla^n f_n = \Delta^n f_{n-j}$ allows Eq.(I.27) to be rewritten as

$$P_n(x) = \sum_{j=0}^{k} \binom{\beta + j - 1}{j}\nabla^j f_{n-j} \qquad (-k \le \beta \le 0). \tag{I.28}$$

The following theorem will be useful in Chapter II.

Theorem I.1. *(Weighted Mean Value Theorem for Integrals)*

If $f \in C[a,b], g$ is integrable on $[a,b]$ and $g(x)$ does not change sign on $[a,b]$, then there exists a number ξ; $a < \xi < b$ such that

$$\int_a^b f(x)g(x)dx = f(\xi)\int_a^b g(x)\,dx.$$

For the proof of this theorem see for instance [95].

Round-Off Error [91]

We say numbers contain *round-off errors* if these numbers are rounded-off to fit into computer words of limited length.

Truncation Error

Truncation error (Discretization error) arises from truncating an infinite process and is due to approximations of the mathematical formulas utilized. The remainder in Taylor's theorem is always an estimate of the size of the truncation error.

Definition.

The initial value problem $y' = f(x,y)$ $(a \leq x \leq b)$ with $y(a) = \alpha$ is said to be a *well posed problem* if

1. A unique solution $y_1(x)$ exists.
2. For any $\delta > 0$ there exists a positive constant M such that for all ϵ with $|\epsilon| < \delta$ and $\lambda \in C[a,b]$ with $|\lambda(x)| < \delta$ $(a \leq x \leq b)$, the unique solution $y_2(x)$ of the IVP $y' = f(x,y) + \lambda(x)$ $(a \leq x \leq b)$ with $y(a) = \alpha + \epsilon$ satisfies $|y_2(x) - y_1(x)| < M\delta$ $(a \leq x \leq b)$.

Stability [96]

Stability means that errors made at one stage of the computations do not cause increasingly larger errors as the computations continue, but eventually dampen out.

Convergence

A numerical method is said to be convergent if

$$\lim_{h \to 0} y_n = y(x_n) \quad \text{for all} \quad x_n = a + nh \in [a,b],$$

where y_n is the computed solution and $y(x_n)$ the true solution (see for details Jean [97]). Note: convergence requires stability, whereas the converse is false. Henrici [98] has proved the following theorem which is very useful in the next chapter.

Theorem I.2. *If f is sufficiently differentiable and p is the order of numerical method, then ε_n satisfies*

$$\varepsilon_n = h^p \delta(x_n) + O(h^{p+1}), \tag{I.29}$$

where $\varepsilon_n = y_n - y(x_n)$ and the magnified error function $\delta(x)$ satisfies the initial value problem

$$\delta'(x) = f_y(x, y(x))\delta(x) - \frac{y^{p+1}(x)}{(p+1)!} \quad with \quad \delta(x_0) = 0.$$

Chapter II
Numerical Methods for Initial Value Problem of ODEs

Problems of ODEs are classified into initial value problems (IVP) and boundary value problems (BVP). Although many authors have worked with Schrödinger's equation as a boundary value problem, we will work with it as initial value problem with solutions selected to satisfy also the boundary conditions (shooting method). Thus, in this chapter we focus on numerical methods for solving the IVP of ODEs. Numerical techniques usually start with the division of the domain $[a, b]$ into subdomains, by means of the points

$$x_0 = a, x_1, x_2, \cdots, x_{i-1}, x_i = b$$

The problem is then solved progressively outwards by use of initial data Eq.(I.3) or Eq.(I.10) and the function f to estimate the solution at x_1. Next, this estimated solution at x_1 is then used as initial data to estimate the solution at x_2 in same fashion and so on.

We consider two classes of numerical techniques: the *one-step method* and the *multi-step method*. The one-step method computes the solution at x_{k+1} using information only from the immediately preceding mesh point x_k, whereas the multi-step method requires information from several previous points. In addition we investigate extrapolation methods, which are very useful for the calculation of the error and improvement of the accuracy of the approximation.

II.1 One-Step Methods

Any scheme for solving a differential equation, in which the approximation y_{i+1} to the solution at the point x_{i+1} can be estimated only if y_i, x_i and h are known, is called a *one-step method.*

II.1.1 Euler Methods [96,99]

We start our study with Euler methods which are easy to analyse and can be used to construct and examine more advance methods. However, in practical numerical problems they are not recommended because their accuracy is very limited. We consider three versions of Euler methods: forward, modified and backward Euler methods. The goal of these methods is to solve the well-posed initial value problem

$$y' = f(x, y) \ (a \leq x \leq b), \quad y(a) = \alpha. \tag{II.1}$$

Forward (Explicit) Euler Method

We obtain the forward Euler method for Eq.(II.1) by rewriting the forward difference approximation

$$\frac{y_{i+1} - y_i}{h} \approx y_i' \tag{II.2}$$

as

$$y_{i+1} = y_i + hf(x_i, y_i), \tag{II.3}$$

from which proceeds

$$y_i = y_{i-1} + hf(x_{i-1}, y_{i-1})$$

with $h = \frac{(b-a)}{i}$. This h is called the *step-size (mesh spacing)* and $x_i \in [a, b]$ is called *mesh point.* We assume these points equally spaced to make the construction

of polynomials easier. This method is called *explicit* because we can compute y_{i+1} without having to solve any equation. We can apply the forward Euler method to a set of first order ODEs and to higher order ODEs. The latter may be reduced to a set of coupled first order differential equation, e.g.

$$y' = f(x, y, z) \text{with } y(0) = y_0, \tag{II.4}$$

$$z' = g(x, y, z) \text{with } z(0) = z_0. \tag{II.5}$$

The Euler method for Eq.(II.4) and Eq.(II.5) becomes

$$y_{i+1} = y_i + hy'_i = y_i + hf(x_i, y_i, z_i), \tag{II.6}$$

$$z_{i+1} = z_i + hz'_i = z_i + hg(x_i, y_i, z_i), \tag{II.7}$$

where the error in one interval (local) is proportional to h^2, while the global error is proportional to h [100].

Modified Euler Method

We derive the modified Euler method by applying the trapeziodel rule to integrate Eq.(II.1), namely

$$y_{i+1} = y_i + \frac{1}{2}h[f(x_{i+1}, y_{i+1}) + f(x_i, y_i)] \tag{II.8}$$

with error in one interval proportional to h^3 and global error proportional to h^2. This method is more accurate and stable than the forward Euler method [93].

Backward (Implicit) Euler Method

A straightforward calculation, by means of the backward difference approximation, leads to the formula

$$y_{i+1} = y_i + hf(x_{i+1}, y_{i+1}) \tag{II.9}$$

of the backward Euler method, which is as accurate as the forward Euler method. This method is also called implicit, because y_{i+1} appears on both sides of Eq.(II.9) and we must solve an implicit equation to get it. Hindmarsh at el [101] have described why and how the implicit Euler method can provide useful solutions of stiff systems and they estimated the error.

II.1.2 Runge-Kutta Methods [96-98]

These methods use a weighted sum of the values of $f(x,y)$ evaluated at the starting point of each step and at various points across the integration step. If $f(x,y)$ is a complicated function, the computer time spent to evaluate specified functions may be excessive. However, these method are stable and relatively efficient if the function $f(x,y)$ is not too complicated. Furthermore, they have the advantage that a change of step causes no problem.

Second-Order Runge-Kutta Method :

The construction is by assuming the solution to have form

$$y_{i+1} = y_i + h(CK_1 + DK_2) \qquad \text{(II.10)}$$

with $K_1 = f(x_i, y_i), K_2 = f(x_i + ah, y_i + bhK_1)$ and arbitrary constants C, D, a and b. We expand y_{i+1} and $y(x_{i+1})$ using Taylor's theorem in terms of h. Assuming $y_i = y(x_i)$ with $y_{i+1} \neq y(x_{i+1})$, we find in general that

$$y(x_i + h) = y(x_i) + hy'(x_i) + \frac{1}{2!}h^2y''(x_i) + \frac{1}{3!}h^3y'''(\xi) = y(x_i) + hf(x_i, y_i) +$$
$$\frac{1}{2!}h^2\left[f_x(x_i, y_i) + f_y(x_i, y_i)f(x_i, y_i)\right] + \frac{1}{3!}h^3y'''(\xi)(x_i < \xi < x_{i+1}) \qquad \text{(II.11)}$$

with $f_x = \frac{\partial f}{\partial x}$ and $f_y = \frac{\partial f}{\partial y}$. Next we find

$$y_{i+1} = y_i + h[CK_1 + DK_2] = y_i + Chf(x_i, y_i) + Dhf(x_i + ah, y_i + bhK_1) =$$
$$y_i + Chf(x_i, y_i) + Dh\left[f(x_i, y_i) + ahf_x(x_i, y_i) + bhf_y(x_i, y_i)f(x_i, y_i) + O(h^2)\right] \qquad \text{(II.12)}$$

with $O(h^2)$ standing for the sum of all terms in h of degree two or higher. Equating coefficients of h^0, h^1, h^2 in Eq.(II.11) and Eq.(II.12) leads to $y_i = y(x_i)$, $(C+D)f(x_i, y_i) = f(x_i, y(x_i))$ and $Daf_x(x_i, y_i) + Dbf_y(x_i, y_i)f(x_i, y_i) = \frac{1}{2}[f_x(x_i, y_i) + f_y(x_i, y_i)f(x_i, y_i)]$, from which in turn the two conditions $[C+D] = 1$ and $aD = bD = \frac{1}{2}$ follow. Therefore, we can create different methods, because we have only three conditions for four coefficients sought after. We have namely:

1. $C = D = \frac{1}{2}$; $a = b = 1$. This is Heun's method [104], proceeds in the two stages $y_{i+1} = y_i + hf(x_i, y_i) = y_i + \frac{1}{2}h[f(x_i, y_i) + f(x_{i+1}, y_{i+1})]$, uses an Euler estimate for y_{i+1} to find an approximation at x_{i+1} and is a *one step predictor corrector method.*
2. $D = 1$; $C = 0$; $a = b = \frac{1}{2}$. This yields the modified Euler's method, whose procedure is $y_{i+\frac{1}{2}} = y_i + \frac{1}{2}hf(x_i, y_i)$ and $y_{i+1} = y_i + hf(x_i + \frac{1}{2}h, y_{i+\frac{1}{2}})$. It is more accurate than Euler's original method because the slope at the (estimated) midpoint is more representative of the whole interval slope than the slope at the left end.

In summary, the second-order Runge-Kutta method may be written as

$$K_1 = hf(x_n, y_n),\ K_2 = hf(x_{n+1}, y_n + K_1),\ y_{n+1} = y_n + \frac{1}{2}[K_1 + K_2].$$

Third-Order Runge-Kutta Method

A Runge-Kutta method of order three is more accurate than second order Runge-Kutta method. We will derive it by using a high-order numerical integration scheme for the second term of equation

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} f(x, y)\, dx. \qquad \text{(II.13)}$$

Using the Simpson's $\frac{1}{3}$ rule, Eq. (II.13) is approximately

$$y_{i+1} = y_i + \frac{1}{6}h\left[f(x_i, y_i) + 4f(x_{i+\frac{1}{2}}, \tilde{y}_{i+\frac{1}{2}}) + f(x_{i+1}, \tilde{y}_{i+1})\right] \tag{II.14}$$

with estimated $\tilde{y}_{i+1}$ and $\tilde{y}_{i+\frac{1}{2}}$, since $y_{i+\frac{1}{2}}$ and y_{i+1} are not known. We obtain

$$\tilde{y}_{i+\frac{1}{2}} = y_i + \frac{1}{2}hf(x_i, y_i). \tag{II.15}$$

from the forward Euler method and the further estimates

$$\tilde{y}_{i+1} = y_i + hf(x_i, y_i), \tag{II.16}$$

$$\tilde{y}_{i+1} = y_i + hf(x_{i+\frac{1}{2}}, \tilde{y}_{i+\frac{1}{2}}), \tag{II.17}$$

$$\tilde{y}_{i+1} = y_i + h\left[\lambda f(x_i, y_i) + [1-\lambda]f(x_{i+\frac{1}{2}}, \tilde{y}_{i+\frac{1}{2}})\right], \tag{II.18}$$

where the last equation is a convex combination in terms of parameter λ of the previous two and λ is so determined that it maximizes the accuracy of the numerical method. From Eq.(II.18), we write the scheme in the following form:

$$K_1 = hf(x_i, y_i), \tag{II.19}$$

$$K_2 = hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}K_1), \tag{II.20}$$

$$K_3 = hf(x_i + h, y_i + \lambda K_1 + [1-\lambda]K_2), \tag{II.21}$$

$$y_{i+1} = y_i + \frac{1}{6}[K_1 + 4K_2 + K_3]. \tag{II.22}$$

After expanding the expressions for K_1, K_2 and K_3 using Taylor's theorem in two variables, for purposes of optimization, we have

$$K_1 = hf \tag{II.23}$$

$$K_2 = hf + \frac{1}{2}h^2[f_x + f_y f] + \frac{1}{8}h^3[f_{xx} + 2f_{xy}f + f_{yy}f^2] \tag{II.24}$$

$$K_3 = hf + h^2[f_x + f_y f] + \frac{1}{2}h^3\left[f_{xx} + 2f_{xy}f + f_{yy}f^2 + [1-\lambda][f_x + f_y f]f_y\right], \tag{II.25}$$

where f and its derivatives are estimated at x_i. On the other hand, a direct expansion of the exact solution y_{i+1} by means of Taylor's theorem leads to

$$y_{i+1} = y_i + hf + \frac{1}{2}h^2[f_x + f_y f] + \frac{1}{6}h^3\left[f_{xx} + 2f_{xy}f + f_{yy}f^2 + f_x f_y + f_y^2 f\right] + O(h^4). \quad \text{(II.26)}$$

By introducing Eq.(II.23)–Eq.(II.25) into Eq.(II.22) and comparing it to Eq.(II.26), we obtain $\lambda = -1$, because Eq.(II.22) agrees with Eq(II.26) up to the third-order term. To summarize the preceeding, the third-order Runge-Kutta method is

$K_1 = hf(x_i, y_i),\ K_2 = hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}K_1),\ K_3 = hf(x_i + h, y_i - K_1 + 2K_2)$

$$y_{i+1} = y_i + \frac{1}{6}\left[K_1 + 4K_2 + K_3\right] \quad \text{(II.27)}$$

Fourth-Order Runge-Kutta Method

Fourth-order Runge-Kutta is the most commonly used and is accurate to the fourth term of the Taylor expansion with local error proportional to h^5. The derivation of the fourth-order Runge-Kutta method is similar to that of third-order method except for an additional intermediate step needed for evaluating the derivative. There are two versions of the fourth-order Runge-Kutta method:

(i) The first version is derived by using the Simpson's $\frac{1}{3}$ rule and is written as

$K_1 = hf(x_i, y_i),\ K_2 = hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}K_1),$

$K_3 = hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}K_2),\ K_4 = hf(x_i + h, y_i + K_3),$

$$y_{i+1} = y_i + \frac{1}{6}\left[K_1 + 2K_2 + 2K_3 + K_4\right]. \quad \text{(II.28)}$$

(ii) The second version is derived by using the Simpson's $\frac{3}{8}$ rule and is written as

$K_1 = hf(x_i, y_i),\ K_2 = hf(x_i + \frac{1}{3}h, y_i + \frac{1}{3}K_1),$

$$K_3 = hf(x_i + \tfrac{2}{3}h, y_i - \tfrac{1}{3}K_1 + \tfrac{1}{3}K_2),\ K_4 = hf(x_i + h, y_i + K_1 - K_2 + K_3),$$

$$y_{i+1} = y_i + \frac{1}{8}[K_1 + 3K_2 + 3K_3 + K_4]. \qquad \text{(II.29)}$$

Application of the fourth-order Runge-Kutta method to higher-order ODEs is straightforward. As an illustration, we consider the second-order differential equation

$$y'' = f(x, y, y'), \qquad y(a) = y_0, \qquad y'(a) = y_{00}$$

and reduce it to two first order ODEs, that is

$$z(x) = y'(x), \qquad z(a) = y_{00},$$

$$y' = g(x, y, z), \qquad y(a) = y_0,$$

$$z' = f(x, y, z), \qquad z(a) = y_{00}.$$

Thus the fourth-order Runge-Kutta becomes

$$K_1 = hg(x_i, y_i, z_i),\ L_1 = hf(x_i, y_i, z_i),$$
$$K_2 = hg(x_i + \tfrac{1}{2}h, y_i + \tfrac{1}{2}K_1, z_i + \tfrac{1}{2}L_1),\ L_2 = hf(x_i + \tfrac{1}{2}h, y_i + \tfrac{1}{2}K_1, z_i + \tfrac{1}{2}L_1),$$
$$K_3 = hg(x_i + \tfrac{1}{2}h, y_i + \tfrac{1}{2}K_2, z_i + \tfrac{1}{2}L_2),\quad L_3 = hf(x_i + \tfrac{1}{2}h, y_i + \tfrac{1}{2}K_2, z_i + \tfrac{1}{2}L_2),$$
$$K_4 = hg(x_i + h, y_i + K_3, z_i + L_3),\ L_4 = hf(x_i + h, y_i + K_3, z_i + L_3),$$
$$y_{i+1} = y_i + \tfrac{1}{6}[K_1 + 2K_2 + 2K_3 + K_4],$$
$$z_{i+1} = z_i + \tfrac{1}{6}[L_1 + 2L_2 + 2L_3 + L_4].$$

Runge-Kutta-Fehlberg Method (RKF45) [104]

This technique consists of using a Runge-Kutta method with local truncation error of order five, which has the following two advantages:

(i) Only six values for the function f are required per step, whereas in general, a Runge-Kutta method of order four or five requires four values for the function.

(ii) This technique determines if the proper step size h is being used. At each step two different approximations for the solution are made and compared. The step is reduced if the answers do not agree to a specified accuracy, and the required step size is increased if the answers agree to more significant digits than required.

The $RKF45$ formulas are

$$K_1 = hf(x_i, y_i),\ K_2 = hf(x_i + \frac{1}{4}h, y_i + \frac{1}{4}K_1),$$
$$K_3 = hf(x_i + \frac{3}{8}h, y_i + \frac{3}{32}K_1 + \frac{9}{32}K_2),$$
$$K_4 = hf(x_i + \frac{12}{13}h, y_i + \frac{1932}{2197}K_1 - \frac{7200}{2197}K_2 + \frac{7296}{2197}K_3),$$
$$K_5 = hf(x_i + h, y_i + \frac{439}{216}K_1 - 8K_2 + \frac{3680}{513}K_3 - \frac{845}{4104}K_4),$$
$$K_6 = hf(x_i + \frac{1}{2}h, y_i - \frac{8}{27}K_1 + 2K_2 - \frac{3544}{2565}K_3 + \frac{1859}{4104}K_4 - \frac{11}{40}K_5),$$
$$y_{i+1} = y_i + \left[\frac{25}{216}K_1 + \frac{1408}{2565}K_3 + \frac{2197}{4104}K_4 - \frac{1}{5}K_5\right], \qquad \text{(II.30)}$$
$$\bar{y}_{i+1} = y_i + \left[\frac{16}{135}K_1 + \frac{6656}{12825}K_3 + \frac{28561}{56430}K_4 - \frac{9}{50}K_5 + \frac{2}{55}K_6\right]. \qquad \text{(II.31)}$$

To use Runge–Kutta method of order four - i.e. Eq.(II.30) - to approximate the solution of the IVP, we need the four quantities K_1, K_3, K_4, and K_5.

A better value for the solution is estimated by using a Runge-Kutta method of order five Eq.(II.31) . The optimal step size $\bar{\beta}h$ can be determined by multiplying the scalar $\bar{\beta}$ times the current step size, where

$$\bar{\beta} = \left[\frac{Tol\ \ h}{2|\ \bar{y}_{i+1} - y_{i+1}\ |}\right]^{\frac{1}{4}} \qquad \text{(II.32)}$$

and the specified error control tolerance Tol as well as the derivation of Eq.(II.32) may be found in [105].

Sixth-Order Runge-Kutta Method

There are three versions of Runge-Kutta of order six. The first version, due to Huta, was analysed by Butcher [97] with

$$y_{i+1} = y_i + \frac{1}{840}\left[41K_1 + 216K_2 + 27K_4 + 272K_5 + 27K_6 + 216K_7 + 41K_8\right],$$

where

$$K_1 = hf(x_i, y_i),\ K_2 = hf(x_i + \frac{1}{9}h, y_i + \frac{1}{9}K_1),$$
$$K_3 = hf(x_i + \frac{1}{6}h, y_i + \frac{1}{24}\left[K_1 + 3K_2\right]),\ K_4 = hf(x_i + \tfrac{1}{3}h, y_i + \tfrac{1}{6}\left[K_1 - 3K_2 + 4K_3\right]),$$
$$K_5 = hf(x_i + \frac{1}{2}h, y_i + \frac{1}{8}\left[-5K_1 + 27K_2 - 24K_3 + 6K_4\right]),$$
$$K_6 = hf(x_i + \frac{2}{3}h, y_i + \frac{1}{9}\left[221K_1 - 981K_2 + 867K_3 - 102K_4 + K_5\right]),$$
$$K_7 = hf(x_i + \frac{5}{6}h, y_i + \frac{1}{48}\left[-183K_1 + 678K_2 - 472K_3 - 66K_4 + 80K_5 + 3K_6\right]),$$
$$K_8 = hf(x_i + h, y_i + \frac{1}{82}\left[761K_1 - 2079K_2 + 1002K_3 + 834K_4 - 454K_5 - 9K_6 + 72K_7\right]).$$

The second version due to Butcher [103] has

$$y_{i+1} = y_i + \left[\frac{11}{120}K_1 + \frac{27}{40}[K_3 + K_4] - \frac{4}{15}[K_5 + K_6] + \frac{11}{120}K_7\right],$$

where

$$K_1 = hf(x_i, y_i),\ K_2 = hf(x_i + \frac{1}{3}h, y_i + \frac{1}{3}K_1),$$
$$K_3 = hf(x_i + \frac{2}{3}h, y_i + \frac{2}{3}K_2),\ K_4 = hf(x_i + \frac{1}{3}h, y_i + \frac{1}{12}[K_1 + 4K_2 - K_3]),$$
$$K_5 = hf(x_i + \frac{1}{2}h, y_i + \frac{1}{16}[-K_1 + 18K_2 - 3K_3 - 6K_4]),$$
$$K_6 = hf(x_i + \frac{1}{2}h, y_i + \frac{1}{8}[9K_2 - 3K_3 - 6K_4 + 4K_5]),$$
$$K_7 = hf(x_i + h, y_i + \frac{1}{44}[9K_1 - 36K_2 + 63K_3 + 72K_4 - 64K_6]).$$

On the other hand, the third version, due to Shanks [102], written as follows

$$y_{i+1} = y_i + \frac{1}{3698}\,[198K_1 + 1225K_3 + 1540K_4 + 810K_5 - 77K_6]$$

has

$$K_1 = hf(x_i, y_i),\ K_2 = hf(x_i + \frac{1}{300}h, y_i + \frac{1}{300}K_1),$$
$$K_3 = hf(x_i + \frac{1}{5}h, y_i + \frac{1}{5}[-29K_1 + 30K_2]),$$
$$K_4 = hf(x_i + \frac{3}{5}h, y_i + \frac{1}{5}[323K_1 - 330K_2 + 10K_3]),$$
$$K_5 = hf(x_i + \frac{14}{15}h, y_i + \frac{1}{810}[-510104K_1 + 521640K_2 - 12705K_3 + 1925K_4]),$$
$$K_6 = hf(x_i + h, y_i + \frac{1}{77}[-417923K_1 + 427350K_2 - 10605K_3 + 1309K_4 - 54K_5]).$$

II.2 Multi-Step Methods [99]

In these methods we use the approximation at more than one previous mesh point to determine the approximation at the next point. For instance, to estimate y_{i+1} we must know y_i and a certain number of preceding values y_{i-1}, y_{i-2}, $y_{i-3}, \cdots$; whereas in one-step methods y_{i+1} is estimated in terms of y_i only - i.e. no information is needed about the preceding values.

II.2.1 Predictor-Corrector Methods

A predictor-corrector-method consists of a predictor step and a corrector step in each interval. The predictor estimates the solution for the new points and then the corrector improves its accuracy. Predictor-Corrector methods use the solutions for previous points instead of using intermediate points in each interval. The starting value in these methods must be specified by assuming $\tilde{y}_0 = \alpha$ followed by the use of one-step techniques to generate the remaining values. For further extensions

and discussion see [106], and [107] for application to stiff systems. Let us derive multi-step methods to calculate y_{i+1} at $x_{i+1} = x_i + h$ from a known value y_i. We first integrate Eq.(II.1) over the interval $[x_i, x_{i+1}]$

$$y_{i+1} - y_i = \int_{x_i}^{x_{i+1}} y'\,dx = y(x_{i+1}) - y(x_i) = \int_{x_i}^{x_{i+1}} f(x, y(x))\,dx, \qquad \text{(II.33)}$$

where the integrand $f(x, y(x))$ is replaced by an interpolation polynomial P determined on the previously obtained data points $(x_0, \tilde{y}_0)$, $(x_1, \tilde{y}_1)$, $\cdots$, $(x_i, \tilde{y}_i)$. If $y_i \approx \tilde{y}_i$, then Eq.(II.33) becomes

$$y_{i+1} \approx y_i + \int_{x_i}^{x_{i+1}} P(x)\,dx. \qquad \text{(II.34)}$$

To derive an Adams-Bashforth explicit m-step technique, we form the back difference polynomial P_{m-1} through the points (x_{i+1}, y_{i+1}), (x_i, y_i), (x_{i-1}, y_{i-1}),..., (x_{i+1-m}, y_{i+1-m}). However, P_{m-1} is an interpolation polynomial of degree $m-1$, thus we have

$$f(x, y(x)) = P_{m-1}(x) + \frac{1}{m!} f^m(\xi_i, y(\xi_i))(x - x_i)(x - x_{i-1}) \cdots (x - x_{i+1-m}).$$

for some number ξ_i $(x_{i+1-m} \le \xi \le x_{i+1})$. The substitutions $x = x_i + sh$ and $dx = hds$ yield

$$\int_{x_i}^{x_{i+1}} f(x, y(x))\,dx = \int_{x_i}^{x_{i+1}} \sum_{k=0}^{m-1} (-1)^k \binom{-s}{k} \nabla^k f(x_i, y(x_i))\,dx$$

$$+ \int_{x_i}^{x_{i+1}} \frac{1}{m!} f^m(\xi_i, y(\xi_i))(x - x_i)(x - x_{i-1}) \cdots (x - x_{i+1-m})\,dx =$$

$$\sum_{k=0}^{m-1} \nabla^k f(x_i, y(x_i)) h (-1)^k \int_0^1 \binom{-s}{k} ds$$

$$+ \frac{1}{m!} h^{m+1} \int_0^1 s(s+1) \cdots (s+m-1) f^{(m)}(\xi_i, y(\xi_i))\,ds. \qquad \text{(II.35)}$$

The integrals $I = (-1)^k \int_0^1 \binom{-s}{k} ds$ for various values of k are easily evaluated from the subsequent table.

k	0	1	2	3	4	5
I	1	$\frac{1}{2}$	$\frac{5}{12}$	$\frac{3}{8}$	$\frac{251}{720}$	$\frac{95}{288}$

As a consequence thereof,

$$\int_{x_i}^{x_{i+1}} f(x, y(x))\, dx = h\left[f(x_i, y_i) + \frac{1}{2}\nabla f(x_i, y_i) + \frac{5}{12}\nabla^2 f(x_i, y_i) + \cdots\right]$$

$$+\frac{1}{m!}h^{m+1}\int_0^1 s(s+1)\cdots(s+m-1) f^{(m)}(\xi_i, y(\xi_i))\, ds, \qquad \text{(II.36)}$$

since $s(s+1)\cdots(s+m-1)$ does not change sign in $[0,1]$. We can apply the Weighted Mean Theorem I.1. to the error term for $x_{i+1-m} < \mu_i < x_{i+1}$ and thus the error term in Eq.(II.35) becomes

$$\frac{1}{m!}h^{m+1}\int_0^1 s(s+1)\cdots(s+m-1) f^{(m)}(\xi_i, y(\xi_i))\, ds$$

$$= \frac{1}{m!}h^{m+1} f^{(m)}(\mu_i, y(\mu_i))\int_0^1 s(s+1)\cdots(s+m-1)\, ds$$

$$= h^{m+1} f^{(m)}(\mu_i, y(\mu_i))(-1)^m \int_0^1 \binom{-s}{m} ds. \qquad (II.37)$$

Utilizing Eq.(II.37) leads to

$$\int_{x_i}^{x_{i+1}} f(x, y(x))\, dx = h\left[f(x_i, y_i) + \frac{1}{2}\nabla f(x_i, y_i) + \frac{5}{12}\nabla^2 f(x_i, y_i) + \cdots\right]$$

$$+h^{m+1} f^{(m)}(\mu_i, y(\mu_i))(-1)^m \int_0^1 \binom{-s}{m} ds, \qquad \text{(II.38)}$$

and applying Eq.(II.38) to Eq.(II.33) implies

$$y_{i+1} = y_i + h\left[f(x_i,y_i) + \frac{1}{2}\nabla f(x_i,y_i) + \frac{5}{12}\nabla^2 f(x_i,y_i) + \cdots\right]$$

$$+h^{m+1}f^{(m)}(\mu_i,y(\mu_i))(-1)^m \int_0^1 \binom{-s}{m}\,ds. \qquad \text{(II.39)}$$

Eq.(II.39) is called the Adams-Bashforth explicit m-step technique. For instance, when $m = 3$, Eq.(II.39) gives us

$$y_{i+1} \approx y_i + h\left[f(x_i,y_i) + \frac{1}{2}\nabla f(x_i,y_i) + \frac{5}{12}\nabla^2 f(x_i,y_i)\right] =$$

$$y_i + h\left[f(x_i,y_i) + \frac{1}{2}[f(x_i,y_i) - f(x_{i-1},y_{i-1})]\right.$$

$$\left.+\frac{5}{12}[f(x_i,y_i) - 2f(x_{i-1},y_{i-1}) + f(x_{i-2},y_{i-2})]\right] =$$

$$y(x_i) + \frac{1}{12}h\,[23f(x_i,y_i) - 16f(x_{i-1},y_{i-1}) + 5f(x_{i-2},y_{i-2})], \qquad \text{(II.40)}$$

which relations carry the name of *Three Step Adams-Bashforth Technique.* Therefore, for various values of m, Eq.(II.39) produces the subsequent table.

Order $= m$	Adams-Bashforth(explicit) formula with order 1 to 6
1	$y_{i+1} = y_i + hy'_{i+1}$
2	$y_{i+1} = y_i + \frac{1}{2}h[3y'_i - y'_{i-1}]$
3	$y_{i+1} = y_i + \frac{1}{12}h\,[23y'_i - 16y'_{i-1} + 5y'_{i-2}]$
4	$y_{i+1} = y_i + \frac{1}{24}h\,[55y'_i - 59y'_{i-1} + 37y'_{i-2} - 9y'_{i-3}]$
5	$y_{i+1} = y_i + \frac{1}{720}h\,[1901y'_i - 2774y'_{i-1} + 2616y'_{i-2} - 1274y'_{i-3} + 251y'_{i-4}]$
6	$y_{i+1} = y_i + \frac{1}{1440}h\,[4227y'_i - 7673y'_{i-1} + 9482y'_{i-2} - 6798y'_{i-3} + 2627y'_{i-4} - 425y'_{i-5}]$

The Corrector formulas or *Adams-Moulton techniques* may be derived by using $(x_{i+1}, f(x_{i+1}, y(x_{i+1})))$ as an additional interpolation node. In the approximation of the integral $\int_{x_i}^{x_{i+1}} f(x, y(x))\, dx$

$$P_n(x) \approx \sum_{k=0}^{m} (-1)^k \binom{-s+1}{k} \nabla^k f(x_{i+1}, y_{i+1}) =$$

$$\sum_{k=0}^{m} (-1)^k \binom{-s+1}{k} \Delta^k f(x_{i+1-k}, y_{i+1-k}). \qquad \text{(II.41)}$$

The use of Eq.(II.33) yields the *Adamas-Moulton* corrector formula

$$y_{i+1} = y_i + h \left[c_0 y'_{i+1} + c_1 \Delta y'_i + \cdots + c_m \Delta^m y'_{i-m} \right],$$

$$c_k = (-1)^k \int_0^1 \binom{-s+1}{k} ds. \qquad \text{(II.42)}$$

It is easy to evaluate this integral for various values of k via the table appearing next.

k	0	1	2	3	4
c_k	1	$\frac{-1}{2}$	$\frac{-1}{12}$	$\frac{-1}{24}$	$\frac{-19}{720}$

By setting $m = 2$ in Eq.(II.42), we obtain the *third-order Adamas-Moulton Corrector method*

$$y_{i+1} = y_i + h \left[y'_{i+1} - \frac{1}{2} \Delta y'_i - \frac{1}{12} \Delta^2 y'_{i-1} \right] = y_i + \frac{1}{12} h \left[5y'_{i+1} + 8y'_i - y'_{i-1} \right].$$

Order $= m-1$	Adams-Bashforth (implicit) formula with order 1 to 6
1	$y_{i+1} = y_i + hy'_i$
2	$y_{i+1} = y_i + \frac{1}{2}h[y'_{i+1} + y'_i]$
3	$y_{i+1} = y_i + \frac{1}{12}h\left[5y'_{i+1} + 8y'_i - y'_{i-1}\right]$
4	$y_{i+1} = y_i + \frac{1}{24}h\left[9y'_{i+1} + 19y'_i - 5y'_{i-1} + y'_{i-2}\right]$
5	$y_{i+1} = y_i + \frac{1}{720}h\left[251y'_{i+1} + 646y'_i - 264y'_{i-1} + 106y'_{i-2} - 19y'_{i-3}\right]$
6	$y_{i+1} = y_i + \frac{1}{1440}h\left[475y'_{i+1} + 1427y'_i - 798y'_{i-1} + 482y'_{i-2} - 173y'_{i-3} + 27y'_{i-4}\right]$

Disadvantages and Advantages of a Predictor-Corrector Methods

Some disadvantages are as follows:

(a) The predictor-Corrector method cannot be used if y' becomes discontinuous.

(b) Changing the interval size in the middle of solution is not easy, because of the use of previous points.

(c) The method cannot be started by itself because previous points are used until the solutions for enough points are determined; instead, another method, such as a Runge-Kutta methods must be used. [108]

Nevertheless, there are some advantages, which are as follows:

(a) The local error can be detected at each step with a small computing effort.

(b) These methods use information from previous steps - i.e. $f(x, y)$ is evaluated twice in each step, regardless of the order of the predictor corrector method. On the other hand, the fourth-order Runge-Kutta method evaluates $f(x, y)$ four times in each interval.

Summary of previously discussed methods occurs in the following table:

Methods	Relevant formula	Error		Other features
		Local	Global	
Euler methods				
Forward	Forward difference	$O(h^2)$	$O(h)$	SS,SC
Modified	Trapeziodal rule	$O(h^3)$	$O(h^2)$	SS,SC
Backward	Backward difference	$O(h^2)$	$O(h)$	SS,SC
Runge-Kutta				
Second-order	Trapezoidal rule	$O(h^3)$	$O(h^2)$	SS,SC
Third-order	Simpson's $\frac{1}{3}$ rule	$O(h^4)$	$O(h^3)$	SS,SC
Fourth-order	Simpson's $\frac{1}{3}$ or $\frac{3}{8}$	$O(h^5)$	$O(h^4)$	SS,SC
Predictor-Corrector				
Second-order	Trapezoidal rule	$O(h^3)$	$O(h^2)$	SS,SC
Third-order	Newton backward	$O(h^4)$	$O(h^3)$	NS,DC
Fourth-order	Newton backward	$O(h^5)$	$O(h^4)$	NS,DC

where the symbols SS, SC, NS and DC stand for *self-starting capability, stepsize can be changed easily in the middle of solution, no self-starting capability* and *difficult to change the stepsize* respectively.

II.3 Approximation of Truncation Error [99]

Suppose we calculate $y(x)$ using a certain h and $y_n(h)$, then we repeat the calculation using $\frac{h}{2}$ and thereby obtain $y_n(\frac{h}{2})$. It follows from Eq.(I.33) that

$$y_n(h) = y(x_n) + h^p \delta(x_n) + O(h^{p+1}), \tag{II.43}$$

$$y_n(\frac{h}{2}) = y(x_n) + (\frac{h}{2})^p \delta(x_n) + O(h^{p+1}). \tag{II.44}$$

We now have two computed values for y_n. The objective is to use these two values to produce a more accurate approximation of $y(x_n)$. From Eq.(I.33) and Eq.(II.44) it becomes apparent that

$$y_n(h) - y_n(\frac{h}{2}) = \left(1 - \frac{1}{2^p}\right) h^p \delta(x_n) + O(h^{p+1}), \tag{II.45}$$

and Eq.(II.44) and Eq.(II.45) let us conclude

$$\varepsilon_n = \frac{2^p}{2^p - 1}\left[y_n(h) - y_n(\frac{h}{2})\right] + O(h^{p+1}). \tag{II.46}$$

Herewith we obtain the *Richardson extrapolation* to the true solution at mesh point x_n, namely

$$y(x_n) = \frac{2^p y_n(\frac{h}{2}) - y_n(h)}{2^p - 1} + O(h^{p+1}). \tag{II.47}$$

These last two results - i.e. Eq.(II.46) and Eq.(II.47) - we write as follows:

$$\varepsilon_n \cong \frac{2^p}{2^p - 1}\left[y_n(h) - y_n(\frac{h}{2})\right], \tag{II.48}$$

$$y(x_n) \cong \frac{2^p y_n(\frac{h}{2}) - y_n(h)}{2^p - 1}. \tag{II.49}$$

Eq.(II.49) is a new approximation to $y(x_n)$ and it differs from $y(x_n)$ by only $O(h^{p+1})$. Hence, by forming an appropriate linear combination of the numerically evaluated results by using h and $\frac{h}{2}$, we have eliminated the leading error term in

the asymptotic error expansion. This procedure can be repeated indefinitely with $\frac{h}{4}, \frac{h}{8}, etc.$

For estimating the accumulated truncation error and the true solution at x_n, the error of order exceeds the order of the method by one. If we assume P_n to be the predicted accumulated error and T_n the actual error in the extrapolated solution, then

$$P_n = \frac{2^p}{2^p - 1}\left[y_n(h) - y_n(\frac{h}{2})\right], \; T_n = \frac{2^p y_n(\frac{h}{2}) - y_n(h)}{2^p - 1} - y(x_n).$$

II.4 Extrapolation Method

The goal of this technique is to obtain results of high accuracy from a formula of lower order. The history and application of extrapolation techniques is discussed in a very interesting article by Joyce [109] and for further discussion see Deuflhard [110].

Richardson's extrapolation technique for estimating the accumulated truncation (discretization) error and improving the approximate value of $y(x_n)$ has the advantage that its order exceeds the order of the numerical method by one. We shall now discuss a successive repeated application of this procedure so that the approximate value of the solution tends to the exact value as $h \longrightarrow 0$. Let $y(x)$ be the true solution of differential equation

$$y' = f(x, y), \qquad y(x_0) = y_0, \qquad x \in [x_0, b].$$

Let us denote the approximate solution by $y(x, h)$, which is obtained by using step length h and a suitable numerical method; the value of $y(x, h)$ will contain an error. We assume that $y(x, h)$ admits an asymptotic expansion in h of the form

$$y(x,h) = y(x) + \chi_1 h^{s_1} + \chi_2 h^{s_2} + \chi_3 h^{s_3} + \cdots + \chi_n h^{s_n} + \chi_{n+1} h^{s_{n+1}} + \cdots, \quad \text{(II.50)}$$

where: $0 < s_1 < s_2 < \cdots < s_n$, $\chi_1, \chi_2, \cdots$ are independent of h and are determined by evaluating $y(x,h)$ with step length h_i $(i = 0,1,2,\cdots)$. For purposes of practicality, we take either $s_i = is$ with step sequence h_i such that $h_0 > h_1 > h_2 > \cdots$ or $h_i = h_0 b^i$ $0 < b < 1$ ($b = \frac{1}{2}$ preferably). The case $s_i = is$ with $h_0 > h_1 > h_2 \cdots > h_m$, in which the approximation $y(x,h)$ could be improved in accuracy by forming linear combinations of $y(x,h)$ at various values of h, is called *" the deferred approach to the limit"* because after a large number of linear combinations one has a very close approximation to the true value $y(x)$. Eq.(II.50) thus becomes

$$y(x,h) = y(x) + \chi_1 h^{s_1} + \chi_2 h^{s_2} + \chi_3 h^{s_3} + \cdots + \chi_n h^{ns} + \mathrm{h.i.}_{n+1} h^{(n+1)s} + \cdots . \quad \text{(II.51)}$$

We now eliminate χ_1, χ_2, χ_3, $\cdots$ by evaluating $y(x,h)$ for $h_0 > h_1 > h_2 \cdots$, and thereby get

$$\begin{aligned}
y(x,h_0) &= y(x) + \chi_1 h_0^s + \chi_2 h_0^{2s} + \chi_3 h_0^{3s} + \cdots + \chi_n h_0^{ns} + \cdots \\
y(x,h_1) &= y(x) + \chi_1 h_1^s + \chi_2 h_1^{2s} + \chi_3 h_1^{3s} + \cdots + \chi_n h_1^{ns} + \cdots \\
y(x,h_2) &= y(x) + \chi_1 h_2^s + \chi_2 h_2^{2s} + \chi_3 h_2^{3s} + \cdots + \chi_n h_2^{ns} + \cdots \\
&\vdots \quad \vdots \quad \vdots \\
y(x,h_n) &= y(x) + \chi_1 h_n^s + \chi_2 h_n^{2s} + \chi_3 h_n^{3s} + \cdots + \chi_n h_n^{ns} + \cdots . \qquad \text{(II.52)}
\end{aligned}$$

Further, eliminating χ_1 in Eq.(II.52) gives

$$\begin{aligned}
\frac{h_0^s y(x,h_1) - h_1^s y(x,h_0)}{h_0^s - h_1^s} &= y(x) - h_0^s h_1^s \chi_2 - h_0^s h_1^s \left[h_0^s + h_1^s\right] \chi_3 - \cdots \\
\frac{h_1^s y(x,h_2) - h_2^s y(x,h_1)}{h_1^s - h_2^s} &= y(x) - h_1^s h_2^s \chi_2 - h_1^s h_2^s \left[h_1^s + h_2^s\right] \chi_3 - \cdots \\
&\vdots \quad \vdots \quad \vdots
\end{aligned}$$

$$\frac{h_{n-1}^s y(x,h_n) - h_n^s y(x,h_{n-1})}{h_{n-1}^s - h_n^s} =$$

$$y(x) - h_{n-1}^s h_n^s \chi_2 - h_{n-1}^s h_n^s \left[h_{n-1}^s + h_n^s\right] \chi_3 - \cdots . \qquad \text{(II.53)}$$

Let us introduce the following notation:

$$Y_n^{(k)} = \frac{h_k^s\, Y_{n-1}^{(k+1)} - h_{k+n}^s\, Y_{n-1}^{(k)}}{h_k^s - h_{k+n}^s},$$

where the subscript n denotes the number of times the linear combination has been applied to eliminate $\chi_1, \chi_2, \cdots, \chi_n$ and the superscript k stands for $y(x,h_k)$ - i.e. the approximation obtained by the numerical method with step length h_k. Thus $Y_0^{(k)} = y(x,h_k)$ and $Y_n^{(k)}$ imply that n linear combinations have been performed to eliminate χ_n. Eq.(II.53) can now be written as

$$Y_1^{(0)} = y(x) - h_0^s h_1^s \chi_2 - h_0^s h_1^s \left[h_0^s + h_1^s\right] \chi_3 - \cdots$$

$$Y_1^{(1)} = y(x) - h_1^s h_2^s \chi_2 - h_1^s h_2^s \left[h_1^s + h_2^s\right] \chi_3 - \cdots$$

$$\vdots \quad \vdots \quad \vdots$$

$$Y_1^{(n-1)} = y(x) - h_{n-1}^s h_n^s \chi_2 - h_{n-1}^s h_n^s \left[h_{n-1}^s + h_n^s\right] \chi_3 - \cdots$$

$$\vdots \quad \vdots \quad \vdots$$

Eliminating χ_2, we obtain

$$Y_2^{(0)} = y(x) + h_0^s h_1^s h_2^s \chi_3 + \cdots$$

$$Y_2^{(1)} = y(x) + h_1^s h_2^s h_3^s \chi_3 + \cdots$$

$$\vdots \quad \vdots \quad \vdots$$

$$Y_2^{(n-2)} = y(x) + h_{n-2}^s h_{n-1}^s h_n^s \chi_3 + \cdots$$

Therefore, we need $Y_n^{(k)}$ for eliminating χ_n, which means

$$Y_n^{(k)} = y(x) + (-1)^n h_k^s h_{k+1}^s \cdots h_{k+n}^s \left[\chi_{n+1} + O(h_k^s)\right], \qquad \text{(II.54)}$$

where the calculation of $Y_n^{(k)}$ can be simplified by means of the triangular array called Y-scheme.

Y-scheme

$$
\begin{array}{lccccccc}
y(x,h_0) = & Y_0^{(0)} & & & & & & \\
 & & Y_1^{(0)} & & & & & \\
y(x,h_1) = & Y_0^{(1)} & & Y_2^{(0)} & & & & \\
 & & Y_1^{(1)} & & Y_3^{(0)} & & & \\
y(x,h_2) = & Y_0^{(2)} & & Y_2^{(1)} & & Y_4^{(0)}. & & \\
 & & Y_1^{(2)} & & Y_3^{(1)} & & & \\
y(x,h_3) = & Y_0^{(3)} & & Y_2^{(2)} & & & & \\
 & & Y_1^{(3)} & & & & & \\
y(x,h_4) = & Y_0^{(4)} & & & & & &
\end{array}
$$

Here, two values are used to form the next approximation according to the rule

$$Y_n^{(k)} = \frac{h_k^s Y_{n-1}^{(k+1)} - h_{k+n}^s Y_{n-1}^{(k)}}{h_k^s - h_{k+n}^s}, \qquad \text{(II.55)}$$

which for $h_i = h_0 b^i$ with $0 < b < 1$ becomes

$$Y_n^{(k)} = \frac{Y_{n-1}^{(k+1)} - b^{ns} Y_{n-1}^{(k)}}{1 - b^{ns}}. \qquad \text{(II.56)}$$

Eq.(II.56) for $b = \frac{1}{2}$ simplifies to

$$Y_n^{(k)} = \frac{2^n Y_{n-1}^{(k+1)} - Y_{n-1}^{(k)}}{2^n - 1} \quad (s = 1) \qquad \text{(II.57)}$$

and

$$Y_n^{(k)} = \frac{4^n Y_{n-1}^{(k+1)} - Y_{n-1}^{(k)}}{4^n - 1} \quad (s = 2). \qquad \text{(II.58)}$$

From Eq.(II.55) we notice that each $Y_n^{(k)}$ is a linear combination of $y(x, h_i)$

$(i = k, k+1, \cdots, k+n)$, which can be written in the form

$$Y_n^{(k)} = \sum_{j=0}^{n} C_{n,n-j} Y_0^{(k+j)} \tag{II.59}$$

with constant coefficients $C_{n,n-j}$. Substituting Eq.(II.59) into Eq.(II.55) yields the following recursion relation for these coefficients:

$$C_{n,n-j} = \frac{h_k^s C_{n-1,n-j} - h_{k+n}^s C_{n-1,n-j-1}}{h_k^s - h_{k+n}^s}, \qquad C_{n-1,n} = C_{n-1,-1} = 0. \tag{II.60}$$

Using Eq.(II.60) and Eq.(II.59) we write

$$\begin{pmatrix} Y_0^{(0)} \\ Y_1^{(0)} \\ \vdots \\ Y_n^{(0)} \end{pmatrix} = \begin{pmatrix} C_{00} & & & & \\ C_{11} & C_{10} & & & \\ & & & & \\ C_{22} & C_{21} & & C_{20} & \\ \vdots & & \vdots & \vdots & \\ C_{nn} & & & & Cn0 \end{pmatrix} \begin{pmatrix} Y_0^{(0)} \\ Y_0^{(1)} \\ \vdots \\ Y_0^{(n)} \end{pmatrix}.$$

It is desirable that numerical methods converge to the true solution as the step size tends to zero - i.e.

$$\lim_{k \to \infty} Y(x, h_k) = Y_0^{(k)} = y(x). \tag{II.61}$$

The convergence of $Y_0^{(n)}$ to $y(x)$ can be readily seen from Eq.(II.54). Eq.(II.54) states that each column of the Y-scheme converges to $y(x)$ faster than the preceeding one. Indeed, the principal diagonal converges faster than any column.

Chapter III

The Numerical Solution of Schrödinger's Equation

In this chapter we examine the Schrödinger equation as an initial as well as boundary value problem. We also investigate some popular numerical methods which are commonly used in the literature to solve the radial Schrödinger equation and also a version for dimension one.

III.1 Schrödinger's Equation

We consider two approaches. First we treat Schrödinger's equation as a Cauchy problem and second (the most common approach) as a boundary value problem.

(a) The one-dimensional Schrödinger equation may be written as

$$\left[\frac{d^2}{dx^2} - f(x)\right] y(x) = 0, \qquad f(x) = u(x) - E,$$

$$u(x) = \left(\frac{2m}{\hbar^2}\right) V(x), \qquad E = \left(\frac{2m}{\hbar^2}\right) \varepsilon,$$

where $u(x)$ and E represent the potential and the energy in dimensionless form. For the purpose of numerical treatment, we write it as

$$y'' + [E - u(x)]\, y = 0. \qquad \text{(III.1.}a\text{)}$$

If the symmetric potentials $u(-x) = u(x)$ are given, then the solutions $y(x)$ are classified as even or odd and the following two sets of initial conditions arise. When $u(x)$ is even, we impose the initial conditions

$$y(0) = 1, \qquad y'(0) = 0, \tag{III.1.b}$$

and in case of $u(x)$ being odd, the conditions

$$y(0) = 0, \qquad y'(0) = 1. \tag{III.1.c}$$

(b) The problem to be considered numerically, called the radial Schrödinger equation, is

$$y'' + (E - u(r))\, y = 0, \qquad r \in (0, \infty), \tag{III.2}$$

with the conditions

$$y(0) = 0, \tag{III.3}$$

$$y(r) \quad \text{is finite} \quad r > 0. \tag{III.4}$$

We note that Eq.(III.3) is similar to the preceeding odd case (a). Our first goal is to transform Eq.(III.4) into a standard mathematical form. Therefore, we make the following assumptions: $\lim_{r\to\infty} u(r) = 0$ and there exists some r_{max} in the asymptotic range of r such that $|E| \gg |u(r)|$ for $r \geq r_{max}$. If this is the case, then the approximation $E - u(r) \simeq E$ is justified and Eq.(III.2) becomes approximately

$$y'' + Ey = 0, \qquad r \geq r_{max}. \tag{III.5}$$

We distinguish for E the following three situations:

(i) $E < 0$. The general solution of Eq.III.5) is

$$y(r) = A\ exp((-E)^{\frac{1}{2}}r) + B\ exp(-(-E)^{\frac{1}{2}}r), \qquad \text{(III.6)}$$

where A and B are arbitrary constants. The only point where condition Eq.(III.4) may be violated is the point at infinity; but it shall be immediately satisfied if $A = 0$ - i.e. the condition Eq.(III.4) is equivalent to the behaviour

$$y(r) = B\ exp(-(-E)^{\frac{1}{2}}r), \qquad r \geq r_{max}. \qquad \text{(III.7)}$$

However, $y(x)$ is the solution of the following differential equation

$$y'(r) + (-E)^{\frac{1}{2}}y(r) = 0, \qquad r \geq r_{max}, \qquad \text{(III.8)}$$

so that it can altogether be said that Eq.(III.4) is equivalent to the condition Eq.(III.8), which looks like the second boundary condition of a standard Sturm-Liouville problem (see Appendix C, Eq.(C.1)). The original problem Eqs(III.2, 3, 8) corresponds to a Sturm-Liouville problem (SLP) with the following three differences:

(1) p,q, and r of the SLP described in Eq.(C.1) were continuous, whereas $u(r)$ appearing in problem Eqs(III.2, 3, 8) may be singular.

(2) The domain in the SLP is finite, whereas it is infinite in problem Eqs(III.2, 3,8).

(3) The last coefficient of the boundary condition Eq.(III.8) is strictly E-dependent, while in the Sturm-Liouville problem the boundary conditions are totally independent of λ. Thus the problem given by Eqs(III.2, 3, 8) is actually an eigenvalue problem just like the original Sturm-Liouville problem, but some of the properties (1)-(4) enumerated for the latter in Appendix C will change.

For instance, property (1) is changed in the sense that the set of eigenvalues $E_0, E_1, E_2, \cdots$ (called eigenenergies) is generally finite, although the corresponding eigenfunctions $y_i(r)$ $(i = 0, 1, 2, \cdots, N)$ remain orthogonal (Appendix C, Eq.(C.3)) if the integration limits are taken from zero to infinity. The rest of the properties remain unchanged.

Now we want to solve the problem in Eqs(III.2, 3, 8) numerically. We consider two main issues, namely the singularity of u at $r = 0$ and the infinite range of the independent variable. The first issue is treated by separating a small subrange $(0, a]$ on which the solution is constructed by some ad-hoc numerical procedure. For the second issue it must be noted, that any attempt to make the range finite by a suitable change of variables may produce a complicated equation with singular coefficients. Thus it is preferable to deal with the original equation and to approximate the infinite range. Two recipes are often taken. First, simply replace condition Eq.(III.7) or Eq.(III.8) (both imply that $y(r) \to 0$ when $r \to \infty$) by

$$y(b) = 0, \tag{III.9}$$

where b is some value in the asymptotic zone - i.e. $b > r_{max}$. The second consists of treating Eq.(III.8) at $r = b$,

$$y'(b) + (-E)^{\frac{1}{2}} y(b) = 0. \tag{III.10.a}$$

Both of these approaches are used in practical applications. The latter is more realistic in the sense that it simulates better the original condition Eq.(III.4) - i.e. $A = 0$ in Eq.(III.6). The first condition guarantees that coefficient A is a small number not exactly zero (as it must be). Also, the first condition requires values of b larger than the second and conesquently a longer time to compute the solution. Using Taylor's theorem, Eq.(III.5) and Eq.(III.8) we get

$$y(r - h) = y(r) Exp\left[(-E)^{\frac{1}{2}} h\right],$$

wherein we replaced r by b and thereby arive at

$$y(b-h) - Exp\left[(-E)^{\frac{1}{2}}h\right]y(b) = 0. \qquad \text{(III.10.}b\text{)}$$

(ii) $E > 0$. In this case, the general solution of the asymptotic Eq.(III.5) is

$$y = A\ sin(E^{\frac{1}{2}}r) + B\ cos(E^{\frac{1}{2}}r) \qquad \text{(III.11)}$$

and hence, condition Eq.(III.4) is satisfied.

(iii) $E = 0$. The general solution of Eq.(III.5) is $y(r) = Ar + B$, so that $|y(r)|$ increases with r.

III.2 Numerov's Method [15,16,99]

Many authors have praised the virtues of the Numerov's method [15,37,111]. We can write the Schrödinger equation Eq.(III.2) in the more general form

$$y'' = f(r,y), \qquad \text{(III.12)}$$

which reduces to Eq.(III.2) if $f(r,y)$ is linear in y. Now we can derive difference equations corresponding to Eq.(III.12) by either: transforming it into a system of two first order equations, or proceeding directly from Eq.(III.12) and noting the absence y' in it. Proceeding by means of the latter, we consider the uniform partition of the finite interval $(0,\bar{r})$:

$$0 = r_0 < r_1 < r_2 \cdots < r_n = \bar{r}, r_k = kh \ \ (k = 0,1,2\cdots,n).$$

Using Taylor's theorem under the assumption that f and $y(r)$ are sufficiently smooth, we write

$$y(r_i + h) = y(r_{i+1}) = y(r_i) + hy'(r_i) + \frac{h^2}{2!}y''(r_i) + \cdots. \qquad \text{(III.13)}$$

Replacing h by $-h$ in Eq.(III.13) entails

$$y(r_i - h) = y(r_{i-1}) = y(r_i) - hy'(r_i) + \frac{h^2}{2!}y''(r_i) - \cdots, \qquad \text{(III.14)}$$

and Eq.(III.13) combined with Eq.(III.14) leads to

$$y(r_{i+1}) - 2y(r_i) + y(r_{i-1}) = \frac{2h^2}{2!}y''(r_i) + \frac{2h^4}{4!}y^{(4)}(r_i) + \cdots, \qquad \text{(III.15)}$$

which equation we differentiate twice and thus write

$$y''(r_{i+1}) - 2y''(r_i) + y''(r_{i-1}) = \frac{2h^2}{2!}y^{(4)}(r_i) + \frac{2h^4}{4!}y^{(6)}(r_i) + \cdots. \qquad \text{(III.16)}$$

From Eq.(III.15) follows

$$h^2y^{(4)}(r_i) = y''(r_{i+1}) - 2y''(r_i) + y''(r_{i-1}) - \frac{2h^4}{4!}y^{(6)}(r_i) + \cdots \qquad \text{(III.17)}$$

and this equation yields in turn, with the use of Eq.(III.17) in Eq.(III.15),

$$y(r_{i+1}) - 2y(r_i) + y(r_{i-1}) = h^2y''(r_i) +$$

$$\frac{h^4}{12}\left[\frac{1}{h^2}[y''(r_{i+1}) - 2y''(r_i) + y''(r_{i-1}) - \frac{2h^4}{4!}y^{(6)}(r_i) + \cdots]\right] + \frac{2h^6}{6!}y^{(6)}(r_i) =$$

$$\frac{h^2}{12}[y''(r_{i+1}) + 10y''(r_i) + y''(r_{i-1})] - \frac{3h^6}{6!}y^{(6)}(r_i) + O(h^8). \qquad \text{(III.18)}$$

Substituting by means of Eq.(III.12) and eliminating the sixth-order terms in Eq.(III.18) gives us *Numerov's method* of order four, which is a two-step method. *Numerov's method* is generally implicit and is reducible to an explicit equation whenever f is linear in y, as is the case with Schrödinger's equation. We may consider two methods to derive higher order schemes, namely

(i) increasing the number of steps if we use $f(r, y)$ only

(ii) keeping the same number of the steps and then using the derivatives of f - i.e. $f_r, f_y, f_{ry}, \cdots$.

For the second method, it must comented that in general the potential $V(r)$ may be given discretely and hence, the derivatives are difficult to calculate. For Numerov's method of order six see Hajj at el [16], and for further extensions as well as applications see [112–122]

III.3 The Method of De Vogelaere [46–54]

The method of De Vogelaere enjoys a large popularity, especially among physicists who are involved in solving Schrödinger's equation (for instance [48–50]). This method is attractive because of its simplicity as well as its flexibility for allowing changes to the step size during an integration. We now derive De Vogelaere's algorithm for the solution of the initial value problem of a second-order equations with the first derivative absent. Consider equidistant points $x_0 = a, x_1, x_2, \cdots, x_{2n} = b$ with mesh spacing h. It is important to note that the odd- and even-labelled points play distinct roles since the last point for the computation must be even-labelled. At each iteration k ($k = 0, 1, 2, \cdots, n-1$) the De Vogelaere algorithm consists of three formulas which compute (in the stated order) $\{\bar{y}_{2k+1}, \bar{y}_{2k+2}, \bar{y}'_{2k+2}\}$ in terms of given $\{\bar{y}_{2k-1}, \bar{y}_{2k}, \bar{y}'_{2k}\}$. Using Taylor's theorem to expand $y(x)$ and $f(x) \equiv f(x, y(x))$ about the point x_{2k} we obtain

$$y(x_{2k} + \lambda) = y(x_{2k}) + \lambda y'(x_{2k}) + \frac{\lambda^2}{2!} y''(x_{2k}) + \cdots + \frac{\lambda^p}{p!} y^p(x_{2k}) + \cdots \quad \text{(III.19)}$$

and

$$f(x_{2k} + \lambda) = f(x_{2k}) + \lambda f'(x_{2k}) + \frac{\lambda^2}{2!} f''(x_{2k}) + \cdots + \frac{\lambda^p}{p!} f^p(x_{2k}) + \cdots, \quad \text{(III.20)}$$

where $y''(x) = f(x)$, $y'''(x) = f'(x)$, $y^{(4)}(x) = f''(x)$, etc. The first expression is

$$I_1 = \frac{h^2}{6}\left[4f(x_{2k}) - f(x_{2k-1})\right],$$

which we want to transform into a suitable form of order h^5. On using Eq.(III.20) with $\lambda = -h$, we have

$$\frac{h^2}{6}\left[4f(x_{2k}) - f(x_{2k-1})\right] =$$

$$\frac{h^2}{6}\left[4f(x_{2k}) - \left\{f(x_{2k}) - hf'(x_{2k}) + \frac{h^2}{2}f''(x_{2k}) + O(h^3)\right\}\right] =$$

$$\frac{h^2}{6}\left[3f(x_{2k}) + hf'(x_{2k}) - \frac{h^2}{2}f''(x_{2k})\right] + O(h^5) =$$

$$\frac{h^2}{2!}f(x_{2k}) + \frac{h^3}{3!}f'(x_{2k}) + \frac{h^4}{4!}f''(x_{2k}) - \left(\frac{1}{12} + \frac{1}{4!}\right)h^4 f''(x_{2k}) + O(h^5) =$$

$$\frac{h^2}{2!}y''(x_{2k}) + \frac{h^3}{3!}y'''(x_{2k}) + \frac{h^4}{4!}y^{(4)}(x_{2k}) - \frac{1}{8}h^4 f''(x_{2k}) + O(h^5) =$$

$$y(x_{2k+1}) - y(x_{2k}) - hy'(x_{2k}) - \frac{1}{8}h^4 f''(x_{2k}) + O(h^5),$$

where the first and last terms give

$$y(x_{2k+1}) = y(x_{2k}) + hy'(x_{2k}) + \frac{h^2}{6}\left[4f(x_{2k} - f(x_{2k-1})\right]$$

$$+\frac{1}{8}h^4 f''(x_{2k}) + O(h^5). \qquad \text{(III.21)}$$

The second expression to be dealt with is

$$I_2 = \frac{h^2}{3}\left[4f(x_{2k+1}) + 2f(x_{2k})\right].$$

Similarly, by setting $\lambda = h$ in Eq.(III.20) and retaining terms up to order h^6 we write

$$\frac{h^2}{3}\left[4f(x_{2k+1}) + 2f(x_{2k})\right] =$$

$$\frac{h^2}{3}\left[4\left\{f(x_{2k})+hf'(x_{2k})+\frac{h^2}{2}f''(x_{2k})+\frac{h^3}{6}f'''(x_{2k})+O(h^4)\right\}+2f(x_{2k})\right]=$$

$$\frac{h^2}{3}\left[6f(x_{2k})+4hf'(x_{2k})+2h^2f''(x_{2k})+\frac{2h^3}{3}f'''(x_{2k})\right]+O(h^6)=$$

$$\frac{(2h)^2}{2!}f(x_{2k})+\frac{(2h)^3}{3!}f'(x_{2k})+\frac{(2h)^4}{4!}f''(x_{2k})+\frac{(2h)^5}{5!}f'''(x_{2k})+\left(\frac{2}{9}-\frac{2^5}{5!}\right)$$

$$h^5f'''(x_{2k})+O(h^6)=y(x_{2k+2})-y(x_{2k})-2hy'(x_{2k})-\frac{2}{45}h^5f'''(x_{2k})+O(h^6),$$

from which we solve for

$$y(x_{2k+2})=y(x_{2k})+2hy'(x_{2k})+$$

$$\frac{h^2}{3}[4f(x_{2k+1})+f(x_{2k})]+\frac{2}{45}h^5f'''(x_{2k})+O(h^6). \qquad \text{(III.22)}$$

The remaining third expression to be calculated is

$$I_3 \;=\; \frac{h}{3}\left[f(x_{2k})+4f(x_{2k+1})+f(x_{2k+2})\right]$$

and we evaluate this up to terms of $O(h^6)$, namely

$$\frac{h}{3}[f(x_{2k})+4f(x_{2k+1})+f(x_{2k+2})]=\frac{h}{3}\left[f(x_{2k})+\right.$$

$$4\left\{f(x_{2k})+hf'(x_{2k})+\frac{h^2}{2}f''(x_{2k})+\frac{h^3}{6}f'''(x_{2k})+\frac{h^4}{24}f^{(4)}(x_{2k})+O(h^5)\right\}+$$

$$\left.f(x_{2k})+(2h)f'(x_{2k})+\frac{(2h)^2}{2}f''(x_{2k})+\frac{(2h)^3}{6}f'''(x_{2k})+\frac{(2h)^4}{24}f^{(4)}(x_{2k})+O(h^5)\right]=$$

$$(2h)f(x_{2k})+\frac{(2h)^2}{2!}f'(x_{2k})+\frac{(2h)^3}{3!}f''(x_{2k})+$$

$$\frac{(2h)^4}{4!}f'''(x_{2k})+\frac{(2h)^5}{5!}f^{(4)}(x_{2k})+\left(\frac{5}{18}-\frac{2^5}{5!}\right)h^5f^{(4)}(x_{2k})+O(h^6)=$$

$$(2h)y''(x_{2k})+\frac{(2h)^2}{2!}y'''(x_{2k})+\frac{(2h)^3}{3!}y^{(4)}(x_{2k})+\frac{(2h)^4}{4!}y^{(5)}(x_{2k})+$$

$$\frac{(2h)^5}{5!}y^{(6)}(x_{2k})+\frac{1}{90}h^5f^{(4)}(x_{2k})+O(h^6)=\;y'(x_{2k+2})-y'(x_{2k})+\frac{1}{90}h^5f(x_{2k})+O(h^6),$$

so that

$$y'(x_{2k+2}) = y'(x_{2k}) +$$

$$\frac{h}{3}\Big[f(x_{2k}) + 4f(x_{2k+1}) + f(x_{2k+2})\Big] - \frac{1}{90}h^3 f^{(4)}(x_{2k}) + O(h^6). \qquad \text{(III.23)}$$

The De Vogelaere algorithm consists of the three formulas Eq.(III.21), Eq.(III.22) and Eq.(III.23), wherein the terms of order h^4, h^5 and h^5 are neglected. Therefore, the algorithm is as follows.

Given $\{\bar{y}_{2k-1}, \bar{y}_{2k}, \bar{y}'_{2k}\}$, compute first $\{f_{2k} = f(x_{2k}, \bar{y}_{2k}), f_{2k-1} = f(x_{2k-1}, \bar{y}_{2k-1})\}$ and thereafter $\bar{y}_{2k+1}$ from

$$\bar{y}_{2k+1} = \bar{y}_{2k} + h\bar{y}'_{2k} + \frac{h^2}{6}\left[4f_{2k} - f_{2k-1}\right], \qquad \text{(III.24)}$$

compute second $f_{2k+1} = f(x_{2k+1}, \bar{y}_{2k+1})$ and thereafter $\bar{y}_{2k+2}$ from

$$\bar{y}_{2k+2} = \bar{y}_{2k} + 2h\bar{y}'_{2k} + \frac{h^2}{3}\left[4f_{2k+1} + 2f_{2k}\right] \qquad \text{(III.25)}$$

and compute third $f_{2k+2} = f(x_{2k+2}, \bar{y}_{2k+2})$ and thereafter $\bar{y}'_{2k+2}$ from

$$\bar{y}'_{2k+2} = \bar{y}'_{2k} + \frac{h}{3}\left[f_{2k} + 4f_{2k+1} + f_{2k+2}\right]. \qquad \text{(III.26)}$$

Note that the derivatives at the odd-labelled mesh points are absent in this algorithm and the values of the resultant solution at the even labelled points are more accurate than those at odd-labelled ones. For the error analysis of this method see Coleman [51].

III.4 Piecewise perturbation methods [42,77,79,82]

In a *perturbation approximation* the given equation is replaced by another differential equation (called a *reference differential equation*), which can be solved

exactly. Perturbation theory estimates the deviation of the solution of the reference equation from the solution of the original equation. To derive numerical perturbation methods means necessarily to follow the idea of the perturbation approximation, which in mathematical physcis is applied globally (on the whole domain of the equation at once, see Bellman [123]), however it in the spirit of numerical analysis - i.e. piecewise. The condition, that the reference equation must admit an analytic solution, significantly restricts the class of differential equations for which piecewise perturbation can be formulated. Fortunately however, linear equations belong to this class. We discuss the case of homogeneous second order linear equations, which includes the Schrödinger equation. The first proposal for piecewise perturbation techniques may be found in Ref.[77]. Each piecewise perturbation method is defined by the recipe used for the piecewise approximation of the coefficients of the differential equation considered: if they are approximated by piecewise constants, then the method is referred to as a *constant perturbation method*, whereas if they are approximated by piecewise lines, then it is called a *line perturbation method.*

General Piecewise Perturbation Approach

If we have a homogeneous second order linear differential equations without the presence of a first order oderivative, then the IVP takes the form

$$y'' = f(x)y(x \in [a,b]), \quad y(a) = y_0, \quad y'(a) = y_0',$$

where $f(x)$ is some given bounded real function. Note that algorithms for such an equation are equally applicable to linear homogeneous second order equations of the general form

$$z'' + a(x)z' + b(x)z = 0,$$

whose general solution may be written in the form

$$z(x) = exp\left[\frac{-1}{2}\int^x a(x')dx'\right] y(x)$$

with y satisfying $y'' = f(x)y$ and

$$f(x) = \frac{1}{2}a'(x) + \frac{1}{4}a^2(x) - b(x). \tag{III.27}$$

We introduce a partition of $[a,b]$: $x_0 = a < x_1 < x_2 \cdots < x_n = b$. There is no special restriction upon the manner of distributing the mesh points except if $f(x)$ is discontinuous at some point, then such a point should be taken as a mesh point. We concentrate on the interval $[x_k, x_{k+1}]$ of length h_k and look for a piecewise perturbation algorithm which propagates the solution from x_k all the way up to x_{k+1}. We introduce variable $\delta = x - x_k$, where $\delta \in [0, h_k]$, and if we denote $X = x_k$ and $g(\delta) \equiv f(X + \delta)$, then the one step problem is

$$y''(X + \delta) = g(\delta)y(X + \delta) \tag{III.28}$$

with the given initial conditions $y(X) = A$ and $y'(X) = B$. Let us denote by u and v the two solutions of Eq.(III.28), which satisfy the particular initial conditions

$$u'' = g(\delta)u, \quad u(0) = 1, \quad u'(0) = 0 \tag{III.29}$$

and

$$v'' = g(\delta)v, \quad v(0) = 0, \quad v'(0) = 1. \tag{III.30}$$

By construction, the functions u and v are linearly independent - i.e. the Wronskian $W[u,v] = uv' - u'v = 1$ at $x = 0$. The solution of Eq.(III.28) with initial conditions A and B can be written in matrix form as follows:

$$\begin{pmatrix} y(X+\delta) \\ y'(X+\delta) \end{pmatrix} = \begin{pmatrix} u(\delta) & v(\delta) \\ u'(\delta) & v'(\delta) \end{pmatrix} \begin{pmatrix} A \\ B \end{pmatrix}. \tag{III.31}$$

The role of u and v is that of propagating the exact solution from x_k to the desired $x_k + \delta$ and consequently, u and v are called *exact propagators*. We associate with Eq.(III.28) an equation of the same form

$$\tilde{y}''(X+\delta) = \tilde{g}(\delta)\tilde{y}(X+\delta), \quad \delta \in [0, h_k], \tag{III.32}$$

where $\tilde{g}(\delta)$ is selected in such a way that this equation has known analytic solutions. We are interested in the propagators $\tilde{u}$ and $\tilde{v}$ associated with Eq.(III.32) - i.e. solutions of the IVPs

$$\tilde{u}'' = \tilde{g}(\delta)\tilde{u}, \quad \tilde{u}(0) = 0, \quad \tilde{u}'(0) = 0 \tag{III.33}$$

$$\tilde{v}'' = \tilde{g}(\delta)\tilde{v}, \quad \tilde{v}(0) = 0, \quad \tilde{v}'(0) = 1. \tag{III.34}$$

The function $\tilde{g}$, Eq.(III.32) and its propagators $\tilde{u}$ and $\tilde{v}$ will be called *a reference function , equation and propagators* respectively. Our goal is to construct the unknown propagators u and v in terms of the known reference propagators $\tilde{u}$ and $\tilde{v}$. We apply perturbation theory to achieve this goal. The idea is to introduce a parametrically dependent function $\mathcal{F}(\delta;\lambda)$ ($\lambda \in [0,1]$), which reproduces the given $g(\lambda)$ and the reference $\tilde{g}(\lambda)$ when λ assumes its extreme value viz. $\mathcal{F}(\delta;0) = \tilde{g}(\delta)$ and $\mathcal{F}(\delta;1) = g(\delta)$. Once $\mathcal{F}(\delta;\lambda)$ is constructed, the propagators u and v sought and the ones $\tilde{u}$ and $\tilde{v}$ given are the particular cases $\lambda = 1, 0$ of the propagators $u(\delta;\lambda)$ and $v(\delta;\lambda)$ of the differential equation

$$y''(X+\delta;\lambda) = \mathcal{F}(\delta;\lambda)y(X+\delta;\lambda), \ \delta \in [0, h_k]. \tag{III.35}$$

The simplest form of $\mathcal{F}(\delta;\lambda)$ is

$$\mathcal{F}(\delta;\lambda) = \tilde{g}(\delta) + \lambda\Delta g(\delta), \tag{III.36}$$

where the deviation

$$\Delta g(\delta) \equiv g(\delta) - \tilde{g}(\delta) \tag{III.37}$$

is called *a perturbation.* The propagators $u(\delta;\lambda)$ and $v(\delta;\lambda)$ are sought as power series in the coupling parameter λ - i.e.

$$u(\delta;\lambda) = \sum_{q=0}^{\infty} \lambda^q u_q(\delta), \quad v(\delta;\lambda) = \sum_{q=0}^{\infty} \lambda^q v_q(\delta). \tag{III.38}$$

Both propagators will be generically denoted by P. Eq.(III.38) are thus collectively written as

$$P(\delta;\lambda) = \sum_{q=0}^{\infty} \lambda^q P_q(\delta), \tag{III.39}$$

where $P = u$ if $P(0;\lambda) = 1$ and $P'(0;\lambda) = 0$, and also $P = v$ if $P(0;\lambda) = 0$ and $P'(0;\lambda) = 1$. To calculate P_q we introduce $P(\delta;\lambda)$ into Eq.(III.35), namely

$$P''(\delta;\lambda) = \left(\tilde{g}(\delta) + \lambda \Delta g(\delta)\right) P(\delta;\lambda), \tag{III.40}$$

and rearrange the terms in powers of λ - i.e.

$$\lambda^0 \left[P_0'' - \tilde{g}(\delta) P_0\right] + \sum_{q=1}^{\infty} \lambda^q \left[P_q'' - \tilde{g}(\delta) P_q - \Delta g(\delta) P_{q-1}\right] = 0. \tag{III.41}$$

This is identically satisfied for any $\lambda \in [0,1]$ if the δ dependent weight of λ^q vanishes for any $q = 0, 1, 2, \cdots$ - i.e.

$$P_0'' = \tilde{g}(\delta)\ P_0 \tag{III.42a}$$

$$P_q'' = \tilde{g}(\delta)\ P_q + \Delta g(\delta) P_{q-1}(\delta) (q = 1, 2, 3, \cdots). \tag{III.42b}$$

To find suitable initial conditions for these differential equations, set $\lambda = 0$ in Eq.(III.39) and thereby obtain $P(\delta;0) = P_0(\delta)$. On the other hand, we also get $\tilde{P}(\delta) = P_0(\delta)$. Thus the initial values of the differences $P(\delta;\lambda) - P_0(\delta)$ and $P'(\delta;\lambda) - P_0'(\delta)$ are forced to vanish, namely

$$\sum_{q=1}^{\infty} \lambda^q P_q(0) = 0, \quad \sum_{q=1}^{\infty} \lambda^q P_q'(0) = 0 (\lambda \in [0,1]), \tag{III.43}$$

whence $P_q(0) = P'(0) = 0 (q = 1, 2, 3, \cdots)$.These results we now summarize in the following

Theorem.

Eq.(III.28) with the initial conditions A and B has the solution Eq.(III.31), where the propagators u and v are written as the perturbation series

$$P(\delta) = P_0(\delta) + P_1(\delta) + P_2(\delta) + \cdots \tag{III.44}$$

with P standing for both u and v. The 0-th order propagator $P_0(\delta)$ is exactly the reference propagator $\tilde{P}(\delta)$, while the correction $P_q(\delta)$ $(q = 1, 2, 3, \cdots)$ is the solution to the problem

$$P_q'' = \tilde{g}(\delta)P_q + \Delta g(\delta)P_{q-1}(\delta), \quad P_q(0) = P_q'(0) = 0. \tag{III.45}$$

In practice, only a finite number of terms are retained in the sum Eq.(III.44). If P_S is the last term retained, then $P(\delta)$ is approximated by

$$\tilde{P}(\delta) = \tilde{P}(0) + P_1(\delta) + \cdots + P_S(\delta)$$

and the resultant algorithm

$$\begin{pmatrix} \bar{y}(X+\delta) \\ \bar{y}'(X+\delta) \end{pmatrix} = \begin{pmatrix} \bar{u}(\delta) & \bar{v}(\delta) \\ \bar{u}'(\delta) & \bar{v}'(\delta) \end{pmatrix} \begin{pmatrix} A \\ B \end{pmatrix} \tag{III.46}$$

is called called a S-th order perturbation algorithm with respect to the reference function $\tilde{g}(\delta)$. The quality of this algorithm depends upon the approximation of the given $g(\delta)$ through the reference function $\tilde{g}(\delta)$ and also the number of perturbative corrections retained.

III.5 The Method of Potential Envelopes of R. Hall

The Schrödinger equation in three dimensions has the form

$$\left[-\frac{\hbar^2}{2m}\Delta + V_0 f(\frac{r'}{a})\right] \psi(r') = \varepsilon\, \psi(r'), \tag{III.47}$$

where V_0, a, m are physical parameters, ε is the energy and $V_0 f(\frac{r'}{a})$ is a central potential. Eq.(III.47) may be transformed to dimensionless form $H\psi = E\psi$ by scaling; thus:

$$r = \frac{r'}{a}, \quad \psi(r') = \psi(r), \quad v = \frac{2mV_0a^2}{\hbar^2}, \quad E = \frac{2m\varepsilon a^2}{\hbar^2}$$

and Eq.(III.47) becomes

$$[-\Delta + vf(r)]\,\psi(r) = E\psi(r) \quad (v > 0, \quad r = |r|), \tag{III.48}$$

where we can write Eq.(III.48) in the form $H\psi(r) = E\psi(r)$. $H = -\Delta + vf(r)$ is called the Hamiltonian, $f(r)$ is a central potential and v is a positive coupling constant. Let us assume $f(r)$ is monotone increasing, smooth and has a nice behaviour at the origin - i.e.

$$f'(r) > 0, \quad \lim_{r\to 0} |r^2 f(r)| = 0. \tag{III.49}$$

With these restrictions and sufficiently large v, the bottom E of the spectrum of H will be a nondegenerate discrete eigenvalue E_{nl} $(l = 0, 1, 2, \cdots)$ and the Hamiltonian H is self-adjoint on some suitable domain $D(H) \subset L^2(\mathrm{R}^3)$. Eigenvalues so labelled have degeneracy precisely $(2l+1)$ and these l's are called angular-momentum quantum numbers. The relation between E_{nl} and v is described by curves

$$E_{nl} = F_{nl}(v) \tag{III.50}$$

called *energy trajectories* of f. We consider a new Hamiltonian $-\Delta + V(r)$ in which the potential V is a smooth increasing transformation $V(r) = g(h(r))$ of the potential h, which is the curve tangent to V. We assume that g is either convex or concave - i.e. $g'' > 0$ or $g'' < 0$. These cases give rise respectively to lower and upper energy bounds.

Now we can describe the situation by the following two expressions

$$-\Delta + vh(r) \longrightarrow F_{nl}(v), \tag{III.51.a}$$

$$-\Delta + V(r) \longrightarrow E_{nl}(v). \tag{III.51.b}$$

For definiteness, we suppose that g is convex so that $g'' > 0$ is convex - i.e. the tangent lines to $g(h)$ all lie below g - and we can consequently write

$$V(r) = g(h(r)) \geq A + vh(r), \tag{III.52}$$

where $A = g(h(t)) - h(t)g(h(t))$, $v = g'(h(t))$ $(t \in (0, \infty))$ and h(t) is the point of contact of $V(r)$ with its tangent potential

$$V^{(t)}(r) = A(t) + v(t)h(r). \tag{III.53}$$

Since the Hamiltonians we consider are self adjoint and bounded below, we can use the variational characterization of the eigenvalues to deduce that the potential inequality Eq.(III.52) implies the corresponding spectral inequality

$$E_{nl} \geq A(t) + F_{nl}(v(t)) = g(h) - hg'(h) + F_{nl}(g'(h)) \tag{III.54}$$

expressed in terms of $h = h(t)$. We now maximize the lower bound Eq.(III.54). By differentiating the right-hand side with respect to h and canceling the factor $g'' > 0$ we get the equation for the critical point

$$h = F'_{nl}(g'(h)). \tag{III.55}$$

Now by use of a Legendre transformation we may reformulate the expression for the best lower bound as

$$E_{nl} \geq \min_{r>0} [K_{nl}(r) + V(r)], \tag{III.56}$$

where

$$K_{nl}(r) = F_{nl}(u) - uF'_{nl}(u), \quad h(r) = F'_{nl}(u). \tag{III.57}$$

Eqs(III.57) representing the functions (K_{nl}) are well defined because of the concavity of F_{nl}: F'_{nl} is monotone and therefore invertible. Eq.(III.56) represents a semi-classical approximation, which is valid whenever the potential $V(r)$ is a convex transformation $g(h(r))$ of the potential $h(r)$.

We note that if the transformation g is concave, then we reverse the inequalities to obtain upper bounds. A more general description of this geometrical theory may be found in Refs. [58-60] and applications of the method are discussed in Refs.[61-68].

Chapter IV

Practical Applications

In this chapter we develop two computer programs for solving the Schrödinger equation with spherically symmetric potentials. Both of these programs are applicable to Schrödinger's equation in one as well as three-dimensions. The one-dimensional case is obtained simply by putting $\ell = -1$, where ℓ is the angular momentum quantum number. The first program treats Schrödinger's equation as an initial value problem and, as examples of this, we solve the following cases: the harmonic oscillator, sech-squared, and the finite square-well potentials. We note that the corresponding eigenvalues of these Schrödinger equations are known exactly. The second program treats Schrödinger's equation as a boundary value problem and as an example of this we study the Woods-Saxon potential. For the convenience of the reader, we exhibit the programs in Appendix A. They are written for a microcomputer using Turbo Pascal version 6; however, the logic of the programs can easily be translated to any computer language accessible to the reader.

IV.1 Program (1)

The program consists of two separate modules, namely the main program and nine subprograms (comprising 3 functions and 6 procedures). The main program controls the solution of the problem, which is executed by the subprograms labeled: Initialize, Goalnodes and Squeeze (which includes the subprograms Nodes

and Iteration). Thereafter, the resulting eigenvalues with the exact eigenvalues and corresponding error are printed. We carry out the program with various values of h, the results of which are collected in the tables of Appendix B. Further, stepsizes of the form $h = 2^{-n}$ $(n = 2, 3, 4, 5, 6)$ may be used for any of the integration methods discussed in Chapters II and III. In addition, the user is free to substitute any other numerical integration procedure.

Function Pot

This is a function-subprogram which calculates the value of $u(x)$ as given in Eq.(III.1). When we write the potential or any other expressions to be programmed, we keep in mind the following:

(1) Among the four elementary arithmetic operations, multiplication and division utilize much more computational time than addition or subtraction. Consequently, we try to keep the number of multiplications and divisions to a bare minimum in expressions which include these arithmetic operations.

(2) The retrieval of data from the core memory is one of the fastest operations in computers; in particular, significantly faster than any arithmetic operation. For this reason, if a program involves repeated calculation of one and the same quantity, then the computation effort is significantly decreased provided that this quantity is computed separately and stored in the core memory for further use.

Functions F_1 and F_2

Here we break down Schrödinger's equation into a system of two first order differential equations in order that we may apply the methods in Chapters II and III.

Procedure Initialize

The program reads the initial data by this procedure, the interval of integration $[a, b]$ for instance. Thereafter, we force the value of the eigenfunctions to decay to zero at the boundary point $x = b$. To find such a suitable b, we first graph the approximate wave function for a given problem and determine therefrom the nearest value of x where the wave function can be assumed to have function-value close to zero. This is called a practical infinity for the problem (or for a class of problems). Further, let E_u and E_l be the upper and lower bounds for the eigenvalues respectively. They are first determined by general mathematical estimates and are inserted into the program. That is to say, we determine two numbers E_u and E_l such that the desired eigenvalue E_n lies between them - i.e. $E_l < E_n < E_u$. We then utilize the value

$$E = 0.5(E_l + E_u) \qquad \text{(IV.1)}$$

and ascertain by means of the program whether E lies above or below the desired E_n. E in Eq.(IV.1) represents a trial value for E. This process is called *halving or bisection*; however, this situation is complicated by the fact that the program's decision depends on the integration step-size h while E depends on h. We shall discuss this further in Section IV.3 .

Procedure Iteration

In this procedure we can use any integration technique discussed in this thesis to compute the eigenfunctions. This procedure further counts the number of nodes - i.e. as soon as the eigenfunction-value equals zero, or the product of the compute eigenfunction-value with the previously computed eigenfunction-value is negative, it augments the node counter P by one.

Procedure Nodes

In this procedure we repeat the iteration of the eigenfunctions, count each node and augment. The process is repeated until the node counter P exceeds the node goal ng or the absolute value of the eigenfunctions exceeds the value YB. YB is a somewhat mysterious entity in the program, typically $YB = 4$. We have found that if $|y|$ exceeds YB and the node goal is not met, then E is too small. In other words, we suppose that the wave function shall not become small again after exceeding YB (as x increases) and produce another node.

Procedure Goalnodes

The purpose of this procedure is to classify two sets of the initial conditions corresponding to the eigenvalues. The even set of eigenvalues $E_n(n = 0, 2, 4, ...)$ corresponds to $y(0) = 1$ and $y'(0) = 0$, while the odd set $E_n(n = 1, 3, 5, \cdots)$ corresponds to $y(0) = 0$ and $y'(0) = 1$. The idea is to look (numerically) for a wave function which is L^2 but not necessarily normalized (i.e of norm one).

Procedure Squeeze

Here we recall the procedure nodes in which we count each node and accumulate. If the node count (at any stage) exceeds ng, then the first trial value E from Eq.(IV.1) was too high. In other words E lies above the desired E_n; therefore, we replace E_u by E and repeat the bisection step with the new trial value for $E = 0.5(newE_u + E_l)$. Conversely, if at the end of generating $y(x)$ (by procedure Iteration) the node count is below the node goal ng, then the first trial value for E in Eq.(IV.1) was too low; therefore, we replace E_l by E and repeat the bisection with the new value for $E = 0.5(E_u + newE_l)$. To guarantee the accuracy of the desired eigenvalue, we repeat this procedure until the difference between E_u and E_l is less than the choosen tolerance (say 10^{-9}).

IV.2 Program (2)

Here we consider the Schrödinger equation

$$y'' + [E - u(x)]\, y = 0, \quad x \in [x_{min} = 0, x_{max}], \quad E < 0 \tag{IV.2}$$

with boundary conditions $y(0) = 0$ and $P(E) = 0$, where

$$P(E) \equiv y'(x_{max}) + (-E)^{\frac{1}{2}} y(x_{max}) \tag{IV.3}$$

of Eq.(III.10.a) or

$$P(E) \equiv y(x_{max}) - Exp\left[(-E)^{\frac{1}{2}} h\right] y(x_{max} + h) \tag{IV.4}$$

of Eq.(III.10.b). A shooting method is used to compute the eigenvalues, which shall require the calculation of $P(E)$ at any E. The objective of this program is to solve Eq.(IV.2) with initial condition $y(0) = 0$ and arbitrary $y'(0) \neq 0$ by generating $y(x_{max})$ and $y'(x_{max})$ or $y(x_{max})$ and $y(x_{max}+h)$. The value of $P(E)$ is determined either by Eq.(IV.3), if one uses one-step methods, or by Eq.(IV.4), if one uses multi-step methods.

Let us explain quite briefly the role of the shooting method in this program. In practice, the energy domain is so chosen that the values $E^0 = E_{min} < E^1 < E^2 < \cdots$ and the values $P(E^0), P(E^1), P(E^2), \cdots$ are successively calculated until the signs of $P(E^i)$ and $P(E^{i-1})$ becomes different. This means that the first eigenvalue lies in the interval (E^{i-1}, E^i), which can be located by computing the root of $P(E)$ via some standard rootfinding procedure such as the bisection method. Once the first eigenvalue is determined, the process goes forward to calculate and compare $P(E^i), P(E^{i+1})$, etc. with the objective of looking for the next eigenvalue, and so on.

Four numerical methods are used in this program to generate $P(E)$, in particular: Standard Numerov Method, De Vogelaere Method, Runge Kutta Method of Order Four and Piecewise Perturbation Methods of Zero Order Corrections. The first two methods belong to the multi-step class, the third belongs to the class of one-step methods and the last one belongs to the class of piecewise perturbation methods.

The program consists of six separate modules, namely the main program and five subprograms labeled as Pot, Prelim, Numsol, P and Root.

Main Program

The reference eigenvalues, energy, stepsizes h and h_{en}, the energy domain E_{min} and E_{max} and the parameters a_0, x_0, PP and QQ of the potentials are given. The main program organizes the solution of the problem and is executed by subprograms Prelim and Numsol. The computed eigenvalues and the reference eigenvalues are finally compared and the errors are printed.

Procedure Numsol

This subprogram explores the mesh points E^i of the energy interval (E_{min}, E_{max}) to determine the positions of the eigenvalues. $P(E^i)$ is computed and its sign is compared with that of $P(E^{i-1})$. Whenever these signs are different, the subprogram Root is called upon to localize the eigenvalue with the desired accuracy.

Function Root

In this subprogram, successive bisections are used to compute the root of $P(E)$ until the length of the interval containing the root is smaller than $Tol = 10^{-8}$.

Function Pot

This function computes the value of $u(x)$ in an efficient way - i.e. in terms of the conditions described in **Function Pot** (Program 1).

Procedure Prelim

In this Procedure the scaled potential values needed by the numerical methods are generated. Prelim calculates the total number of steps in the interval (x_{min}, x_{max}).

IV.3 Applications

We apply the two computer programs to some potentials to confirm the quality of the algorithms.

Example 1: Harmonic Oscillator Potential

The potential function is given by

$$u(x) = ax^2, \tag{IV.5}$$

where a is a free parameter.The vibrational behaviour of continuous physical systems such as an elastic medium can be described by a superposition of harmonic oscillators. One simple and important property of the harmonic oscillator (in one-dimension) is that the potential is an even function of x - i.e. $u(x) = u(-x)$ - and consequently the solution $y(x)$ of Schrödinger's equation is either even or odd (the eigenvalues are of multiplicity 1).

The harmonic oscillator problem is an excellent example to test the accuracy of numerical methods, because the exact eigenvalues are well known [6]. We test the

efficiency of our numerical method by comparing the computational results with the exact eigenvalues (see Appendix B)

$$E_n = 2n + 1 \ (n = 0, 1, 2, 3, \cdots).$$

We compared our computations result for various numerical methods with the exact values for different stepsize in tables B.1–20 of Appendix B.

Example 2: Sech-squared Potential

Here the potential function is given by

$$u(x) = -vsech^2(x), \tag{IV.6}$$

where v is a coupling constant taken to be $v = 100$ so that 10 exact eigenvalues exist. The problem is also solved analytically Flügge [124] and the exact eigenvalues are well known (see Hall [62]), specifically

$$E_n = -\left[(v + \frac{1}{4})^{\frac{1}{2}} - (n + \frac{1}{2})\right]^2.$$

We apply various numerical methods with different stepsizes and we compare our computations with the corresponding exact values. The results are exhibited in tables B.20–40 of Appendix B.

Example 3: Woods-Saxon Potential

The potential $u(x)$ is given by

$$u(x) = \frac{PP}{C(x)} + \frac{Q}{C^2(x)} \, exp\left(\frac{x - x_0}{a_0}\right) \left(C(x) = 1 + exp\left(\frac{x - x_0}{a_0}\right)\right), \tag{IV.7}$$

where PP, Q, a_0 and x_0 are numerical parameters. The Woods-Saxon potential is very popular in nuclear physics. The exact solution is available to be used as

reference to check for the accuracy of the numerical scheme [82] . We will be interested in the numerical computations of the eigenvalues. The following choice of parameters

$$x_0 = 7, \quad a_0 = 0.6, \quad PP = -50, \quad Q = \frac{PP}{a_0}$$

has been studied by Adam at el [82].

Example 4: Finite Square Well Potential

$$u(X) = \begin{cases} -V_0 & | X | < a, \\ 0 & \text{elsewhere.} \end{cases}$$

The finite square well is used as a one dimensional approximation to the potential function for an electron moving through a piece of metal or neutron moving through a nucleus. The potential is $-V_0$ in the region from $x = \pm a$ and zero elsewhere; that is to say, the square well potential is attractive. Now, suppose that a stream of electrons is directed at it from the left. According to classical physics, no electrons would be turned back, but from the wave theory, electrons will be reflected from the sharp edges at $x = \pm a$ and as a result, there will be a reflected and transmitted wave.

The eigenvalue for the square well are well-known (Flügge [124]), they are given in parametric form by the equations (Hall [127])

$$a^2 E_n(V_0) = \begin{cases} -\beta^2 tan^2\beta & n \text{ even}, \\ -\beta^2 cot^2\beta & n \text{ odd}, \end{cases}$$

$$a^2 V_0 = \begin{cases} \beta^2 sec^2\beta & n \text{ even}, \\ -\beta^2 cosec^2\beta & n \text{ odd}, \end{cases}$$

where $\beta_n = \frac{n\pi}{2}$ $(n = 0, 1, 2, \cdots)$ and $\beta \in [\beta_n, \beta_{n+1})$.

IV.4 The Error in the Eigenvalues

Our goal is to study the error of the computed eigenvalues and Eq.(II.49) and to determine whether the simple extrapolation formula is suitable for every numerical method herein mentioned or not. This is very important for reducing the computational effort and for improving the accuracy for the approximate solutions. The reader is referred to sections II.3 and II.4 before reading this section. We assume $E(h)$ admits an asymptotic expansion in h of the following form

$$E(h) = E_0 + \sum_{k=1}^{i} a_k h^k. \tag{IV.5}$$

If $i = 2, 3, \text{or } 4$, then the numerical method we used is of order $2, 3, \text{or} 4$ respectively. For illustration, we use Runge Kutta of order four - i.e. $i = 4$ - and if we replace h by $\frac{h}{2}, \frac{h}{4}, \frac{h}{8}$ and $\frac{h}{16}$ then we get

$$\begin{aligned}
E(h) &= E_0 + a_1 h + a_2 h^2 + a_3 h^3 + a_4 h^4, \\
E\left(\frac{h}{2}\right) &= E_0 + a_1 \left(\frac{h}{2}\right) + a_2 \left(\frac{h}{2}\right)^2 + a_3 \left(\frac{h}{2}\right)^3 + a_4 \left(\frac{h}{2}\right)^4, \\
E\left(\frac{h}{4}\right) &= E_0 + a_1 \left(\frac{h}{4}\right) + a_2 \left(\frac{h}{4}\right)^2 + a_3 \left(\frac{h}{4}\right)^3 + a_4 \left(\frac{h}{4}\right)^4, \\
E\left(\frac{h}{8}\right) &= E_0 + a_1 \left(\frac{h}{8}\right) + a_2 \left(\frac{h}{8}\right)^2 + a_3 \left(\frac{h}{8}\right)^3 + a_4 \left(\frac{h}{8}\right)^4, \\
E\left(\frac{h}{16}\right) &= E_0 + a_1 \left(\frac{h}{16}\right) + a_2 \left(\frac{h}{16}\right)^2 + a_3 \left(\frac{h}{16}\right)^3 + a_4 \left(\frac{h}{16}\right)^4.
\end{aligned}$$

These equation represent a system of five equations in five unknowns - i.e. E_0, a_1, a_2, a_3 and a_4. We can solve this system for the unknowns using any method from linear algebra, Gauss elimination for instance.

For example, we applied the Runge Kutta method of order four to the first eigenvalue of the harmonic oscillator and obtained thereby the following results

The Coeff. of RK4

E_0	0.9999999961
a_1	0.0000004489
a_2	−0.0000169927
a_3	0.0002478665
a_4	0.0091858586

TableIV.1 HO

Similarly, if we apply Runge Kutta of order three and two for the first eigenvalue of harmonic oscillator we get a system of 4 equations with 4 unknowns and a system of 3 equations 3 unknowns respectively, for these two situations we found the following respective results

The Coeff. of RK3

E_0	1.0000001694
a_1	0.3761060421
a_2	0.1487586475
a_3	0.0227654900

TableIV.2 HO

The Coeff. of RK2

E_0	1.0000148055
a_1	−0.0007620016
a_2	−0.2405022891

TableIV.3 HO

Consequently, we found extrapolation equation Eq.(II. 49) is suitable for the Runge Kutta method of order two and four while is unsuitable for the Runge Kutta

method of order three. In other words, for Runge Kutta of order three we have to apply a more elaborate extrapolation formula, that is to say we have to find the coefficients a_1, a_2, a_3 in terms of $E(h), E(\frac{h}{2}), E(\frac{h}{4}), E(\frac{h}{8})$. We applied Runge Kutta of order four, three, and two and thus got the following respective results

The Coeff. of RK4

E_0	−9.2992585452
a_1	−2.1351567705
a_2	−102.2667998834
a_3	−1581.1403537262
a_4	7053.6274812330

TableIV.4 SQ

The Coeff. of RK3

E_0	−9.3059819404
a_1	0.126644037
a_2	−8.0314568192
a_3	72.5502757937

TableIV.5 SQ

The Coeff. of RK2

E_0	−9.2891510897
a_1	−1.0718564672
a_2	7.0520485547

TableIV.6 SQ

From these tables we conclude that the extrapolation Eq.(II.49) is not suitable for Runge Kutta of order two, three and four. Therefore, it is desirable to find the

coefficients a_1 and a_2 for Runge Kutta of order two in terms of $E(h)$ and $E(\frac{h}{2})$ to ensure an accurate approximation. Similar conclusions hold for for Runge Kutta of order four and three.

The finite difference table is one of the most effective procedures for studying polynomial dependence on small parameters. The reader is referred to sectionII.4. We use Eq.(II.58) for the first eigenvalue of the harmonic oscillator and the potentials sech-squared and finite square well. For these we obtained the following results

$E(\frac{1}{4})$ = 1.0000388015

0.9999904179

$E(\frac{1}{8})$ = 1.0000025135 0.9999999707

0.9999993739 .9999999997

$E(\frac{1}{16})$ = 1.0000001585 .9999999994 0.9999999999

0.9999999602 0.9999999998

$E(\frac{1}{32})$ = 1.0000000099 1.0000000004

0.9999999979

$E(\frac{1}{64})$ = 1.0000000005

TableIV.7 Finite Difference Procedure of the First Eigenvalue of the HO

$E(\frac{1}{4}) =$	-90.4645098981			
		-90.4923040706		
$E(\frac{1}{8}) =$	-90.4853555276	-90.4877437287		
		-90.4880287499	-90.4875083955	
$E(\frac{1}{16}) =$	-90.4873604446	-90.4875120727	-90.4875078035	
		-90.4875443649	-90.4875078059	
$E(\frac{1}{32}) =$	-90.4874983851	-90.4875078727		
		-90.4875101529		
$E(\frac{1}{64}) =$	-90.4875072109			

TableIV.8 Finite Difference Procedure of the First Eigenvalue of the Sech-Squared

$E(\frac{1}{4}) =$	-9.2931486868			
		-9.3480466042		
$E(\frac{1}{8}) =$	-9.3343221249	-9.3011153360		
		-9.3040485402	-9.3065474168	
$E(\frac{1}{16}) =$	-9.3116169366	-9.3064625407	-9.3159115440	
		-9.3063116656	-9.3158749651	
$E(\frac{1}{32}) =$	-9.3076379834	-9.3157278960		
		-9.3151393816		
$E(\frac{1}{64}) =$	-9.313240321			

TableIV.9 Finite Difference Procedure of the First Eigenvalue of the SQ

As evident from the finite difference table for harmonic oscillator and sech-squared potential, strong convergence of the computed eigenvalue to the exact eigenvalue takes place. Each column converges to the exact eigenvalue faster than the preceeding one. Moreover, the principal diagonal converges faster than any column.

This justifies the use of Richardson extrapolation Eq.(II.49) for RK4; nevertheless, the finite difference table for the first eigenvalue of the finite square well potential indicates no convergence of the computed eigenvalue to the exact eigenvalue in each column. We can hence say that Eq.(II.49) is unsuitable for SQ.

Chapter V
Conclusion

In this thesis we have studied some effective methods for finding numerical solutions to the time-independent Schrödinger equation. The numerical solution of the time-dependent Schrödinger equation has been discussed in Refs. [125,126]. We have addressed many issues regarding numerical methods for the solution of Schrödinger's equation. In Chapters I and II we surveyed some numerical methods for the initial value problem of ordinary differential equations. We have considered two approaches for solving IVP, in particular the one-and multi-step methods. These methods are applicable to any boundary value problem and yield a sequence of initial value problems in the numerical calculation scheme. We further discussed the extrapolation technique and explained why this technique is so effective in reducing the error arising from the numerical methods. In Chapter III we studied various ways of treating the Schrödinger equation numerically. Although the most common approach is to treat it as an eigenvalue problem, in this thesis we have treated it as a Cauchy problem as well as a boundary-value problem. Moreover, we have investigated some analytic approximation methods, such as the method of potential envelopes, which give us formulae for upper and lower bounds for the eigenvalues, and also perturbation theory . We have studied two of the most powerful numerical methods to be found in the literature, namely Numerov's and De Vogelaere's methods. In Chapter IV we developed two computer programs for Schrödinger's equation considered as a IVP and as a BVP. We applied these programs to the spe-

of the harmonic oscillator, the sech-squared potential and the finite square well and Woods-Saxon potentials. We have also explored the question of the relation between the local integration error and the final error in the eigenvalue estimate.

• The advantages of our program are as follows.

(1) We can easily change the stepsize. Of course, the smaller the stepsize, the more accurate the numerical solution.

(2) Flexibility: the user can use any convenient method in the procedure Iteration.

(3) The procedure Squeeze in Program (1) expresses a simple but powerful idea that easily yields accurate eigenvalues with tolerance (difference of the upper and lower bounds) as small as 10^{-9}.

(4) The procedure Numsol in Program (2) determines the eigenvalue in the interval (E_{min}, E_{max}) by computing $P(E^i)$ and comparing its sign with the sign of $P(E^{i-1})$. Once it detects that both signs differ, the procedure Root is called upon to find the eigenvalue with the desired accuracy.

• We have explored the error of the eigenvalue and found that the simple extrapolation technique in Eq.(II. 49) can be applied to numerical methods of order two and four such as the Runge-Kutta methods of order two and four. However, this simple formula cannot be applied to numerical methods of order three, specifically Runge-Kutta of order three. In the Runge Kutta method of order four (i.e $p = 4$ in Eq.(II.49), we get the following Richardson extrapolation for the eigenvalue

$$E = \frac{16E(\frac{h}{2}) - E(h)}{15} + O(h^6). \qquad \text{(V.1)}$$

We applied Eq.(V.1) to the harmonic oscillator, assuming in program(1) that the parameters have the following values: $b = 8$, $h = 0.25$ and $NSTEPS = 32$. Eq.(V.1) gave us accurate eigenvalues with errors of order h^6 and thus the following table.

RK4

RK4	RK4 with Richardson extrapolation
1.0000388015	1.0000000943
3.0005638208	3.0000023124

TableV.1 Comparison Two Eigenvalues of HO

If we apply Richardson extrapolation Eq.(V.1) to SQ, then the assumption in program(1) of the parameter values $b = 18$, $h = 0.25$, and $V_0 = 10$ yields the following

RK4

RK4	RK4 with Richardson extrapolation
−9.2931486868	−9.337067207
−7.2078134296	−7.3811028167

TableV.2 Comparison Two Eigenvalues of SQ

From this and results from Chapter IV, we found by using Eq.(V.1) that the computed eigenvalues fail to converge to the exact eigenvalue, in other words Eq.(V.1) is unsuitable for SQ. Consequently, we conclude that the applicability of Richardson extrapolation for the numerical computation of the eigenvalues of the Schrödinger equation depends on the nature of the potential. More specifically, if the potential vanishes very quickly as $|x|$ increases (that is to say, faster than exponential), then in the asymptotic expansion

$$E(h) = E_0 + a_1 h + a_2 h^2 + \cdots$$

none of the coefficients may *a priori* be assumed to be small.

Appendix A

Program 1

```
Program Schrödinger;

*************************************************************************
This program is for solving Schrodinger's equation

                        [-∇² + v * u(x)]Ψ = EΨ,

where  v is coupling constant and u(x) is symmetric potential.
Use any method from Chapter II and III
to solve Schrödinger's equation Eq. by breaking it up into two first ODEs.
*************************************************************************

{$N+}
{$F+}
{$R-}
Uses crt, dos, graph, printer;
Const
    prog  =  'Second order Diff. Eqn. by using Runge Kutta of order 4 ';
    TOL   =  0.000000001;
Type
    secondrow=array[0..50] of extended;
Var
    X,a,b,Y,Z,Y0,Z0,h :extended;
    E, EU, EL, P, YB, ERR :extended;
    I, J, Nsteps, n, ng :integer;
    replyc :char;
    EXEV :secondrow;
    elarge :boolean;

Function POT( X : extended): extended;
Begin
    POT := u(X) ;
End;

Function f1( X,Y,Z : extended): extended;
Begin
    f1 := Z;
End;
```

```
Function f2( X,Y,Z : extended): extended;
Var
    Z1 : extended;
Begin
    Z1 := POT( X )- E ;
    f2 := Z1*Y ;
End;

Procedure setparams;
Begin
    Clrscr;
    Writeln('.....');
    Writeln(' ',prog);
    Writeln(' .....');
    Writeln;
    Writeln(' We shall solve  y" = f(x,y)  with:');
    Writeln;
    Writein('  f(X,Y) = (u(x)-E)*Y  ');
    Writeln;
    Writeln('**** ************************************');
    Writeln(' Any method of from chapter II, III ');
    Writeln('****************************************');
    Writeln(' ',' Exact EXEV ', ' COMPUTED EV. ',' ',' ERROR ');
    Writeln('.....');
End;

Procedure Iteration;
Var
Begin
    This procedure is to compute the eigenfunctions
    as soon as the value of the eigenfunction is zero or
    the product of the computed eigenfunction-value
    and
    the previous computed eigenfunction-value is neg-
    ative it counts the nodes.
    We can use any method from Chapter II and III.
End;

Procedure Nodes;
Begin
    P .= 0 ;
    Repeat
    Solve;
    Until ( ( I = Nsteps-1 ) Or ( P>ng )Or(abs( Y ) > YB ));
    If P > ng Then elarge := true Else elarge := false;
End;
```

```
Procedure Goalnodes;
Begin
    If Odd(n) Then
    Begin
        ng := round((n-1)/2);
        Y0 := 0.0;          Z0 := 1.0
    End
    Else
    Begin
        ng := round(n/2);
        Y0 := 1;            Z0 := 0.0
    End;
End;

Procedure Squeeze;
Begin
    Repeat
    Nodes;
    If elarge Then
    Begin
        EU := E ;
        E := (EU+EL)/2 ;
    End
    Else
    Begin
        EL:= E ;
        E := (EU+EL)/2 ;
    End;
    Until (EU-EL) < TOL ;
End;

Procedure Initalize;
Begin
    a := 0.0 ;          b := ? ;          {End points}
    Nsteps := ? ;                         {Number of steps}
    EU := ? ;           EL := ? ;         {The upper and lower bound}
    YB := ? ;
    E := (EU+EL)/2 ;
End;

Begin                   {main Program}
    clrscr;
    Setparams;
    EXEV[0] := ? ;                        EXEV[1] := ? ;
    EXEV[2] := ? ;                        EXEV[3] := ? ;
    EXEV[4] := ? ;                        EXEV[5] := ? ;
    EXEV[6] := ? ;                        EXEV[7] := ? ;
    EXEV[8] := ? ;                        EXEV[9] := ? ;
    EXEV[10] := ? ;                       EXEV[11] := ? ;
    EXEV[12] := ? ;
```

```
    For J := 0 To 12 Do
    Begin
        n := J ;
        Initalize;
        Goalnodes;
        Squeeze;
        ERR := EXEV[J]-E ;
        Writeln( J ,')', EXEV[J] : 15 : 10 ,' ', E : 15 : 10 ,' ', ERR : 15 : 10 );
    End;
    Writeln;
    Writeln('Step size = ', h : 13 : 8 );Writeln;
    Writeln('Number of steps = ', NSTEPS : 8 );
replyc := readkey
End.
```

Program 2

```
Program Schrödinger;

*************************************************************************
This is program for solving Schrodinger's equation

                        [-∇^2 + v * u(x)]Ψ = EΨ.

Where v is coupling constant and u(x) here is the Woods-saxon potential.
Using any method from chapter II, III.
*************************************************************************

{$N+}
{$F+}
{$R-}
Uses crt, dos, graph, printer;
Type
    lastrow =  array[0..200] of extended;
    firstrow = array [1..1500] of extended;
    secondrow = array[0..13] of extended;
    func = function(X : extended) : extended;
Var
    tt, ss, A0, X0, PP, QQ, ERR, h1, h2, H, xmin, xmax
    ,emin , emax, xmin1, xmax1, emin1, emax1 : extended;
    nstep, nstep1, nstep2 : integer;
    L,I,iref,imax,hen,hen1,imax1 : integer;
    EXEV : secondrow;
    Compev,COMPEV1 : secondrow;
    replyc : char;
    V,V1,V2 : firstrow;
```

```
Function POT(X : extended): extended;
Var
    EXPON,B : extended
Begin
    EXPON := EXP((X-X0)/A0);
    B := 1+EXPON;
    POT := PP/B+QQ*EXPON/(B*B);
End;

Procedure PRELIM(Var xmin,xmax,h : extended; Var nstep : integer);
Var
    K,N : integer;
    X,HH : extended;
Begin  Procedure
    NSTEP := round((xmax-xmin)/ H);          {Numerov's Method}
    X := xmin;
    N := nstep+1;
    HH := H*H/12.;
    For K := 1 To N Do
    Begin
        V[K] := HH*POT(X);
        X := X+H;
    End;
    V[N+1] := 0.0;
End;                                  {end of the procedure}

Function P(Var NSTEP :  integer; Var H,E :  extended) : extended;
Var
    EH,Y0,D0,D1,D2,Y1,Y2 : extended;
    k,NMAX : integer;
Begin
    EH := H*H*E/12;
    NMAX := nstep+2;
    D0 := EH-V[1] ;
    D1 := EH-V[2];
    Y0 := 0.0;
    Y1 := H;
    For K := 3 To NMAX Do
    Begin
        D2 := EH-V[K];
        Y2 := ( (2-10*D1)*Y1-(1+D0)*Y0)/ (1+D2);
        D0 := D1;
        D1 := D2;
        Y0 := Y1;
        Y1 := Y2;
    End;          {for}
    P := Y1-Y0*EXP(-H*SQRT(-E));
End;                              {Function}
```

```
Function Root(Var EA,EB,PA,PB,h2 : extended; Var nstep2 : integer) : extended;
Var
    E1,E2,P1,P2,PTEST,ETEST,PROD,DEV,tol : extended;
    label 10,20,30,40;
Begin
    TOL := 8.E-5;
    E1 := EA;
    E2 := EB;
    P1 := PA;
    P2 := PB;
  20: ETEST := 0.5*(E1+E2);
    PTEST := P(nstep2,h2,Etest);
    If PTEST = 0 Then GoTo 40;
    PROD: = P1*PTEST;
    If PROD > 0 Then GoTo 10;
    E2 := ETEST;
    P2 := PTEST;
    GOTo 30 ;
  10 :E1 := ETEST;
    P1 := PTEST;
  30 :DEV := ABS(E2-E1);
    If DEV > TOL Then GoTo 20;
    ROOT := (E1*P2-E2*P1)/(P2-P1);
  40 :ROOT  := ETEST;
End;

Procedure NUMSOL(Var nstep2,imax1,hen1 : integer;
Var h1,emin1,emax1 : extended;Compev1 : secondrow);
Var
    E,PROD,P1,P2,ENEXV : extended;
    label 10,20;
Begin
    IMAX1 := 0;
    E := EMIN1;
    P1 := P(nstep2,H1,E);
  20 :ENEXT := E+HEN1;
    If ENEXT > EMAX1 Then ENEXT := EMAX1;
    P2 := P(nstep2,h1,ENEXT);
    PROD := P1*P2;
    If PROD > 0 Then GoTo 10;
    Compev[imax1] := ROOT(E,ENEXT,P1,P2,h1,nstep2);
    imax1 := imax1+1;
  10: P1 := P2;
    E := ENEXT;
    If(E < EMAX1)Or(E > EMAX1)Then GoTo 20 ;
End;

Begin                              {main Program}
    clrscr;
    EXEV[0] := -49.457788728;      EXEV[1] := -48.148430420;
    EXEV[2] := -46.290753954;      EXEV[3] := -43.968318432;
```

```
    EXEV[4] := -41.23607772;        EXEV[5] := -38.122785097;
    EXEV[6] := -34.672313206;       EXEV[7] := -30.912247488;
    EXEV[8] := -26.873448916;       EXEV[9] := -22.588602258;
    EXEV[10] := -18.094688282;      EXEV[11] := -13.436869040;
    EXEV[12] := -8.676081671;       EXEV[13] := -3.908232481;
    A0:=0.6;          X0 := 7.0;
    PP := -50.0;          QQ := -PP/A0;
    Xmin := 0;          Xmax := 10;
    Emin:=-50;          Emax := 0.0;
    Hen := 1;
    h := 1./16;
    Writeln(' ***********************************');
    Writeln(' Standard Numerov method ');
    Writeln(' ***********************************');
    Writeln;
    WRITELN('STEPSIZE' : 25,H : 12);
    PRELIM(xmin,xmax,H,nstep);
    NUMSOL(nstep,imax,hen,H,emin,emax,Compev);
    Writeln('imax =',imax);
    Writeln;
    Writeln('Number of steps ',nstep);
    writeln;
    WRITELN(' ',' Exact EXEV ', ' COMPUTED EV. ', 'ERROR ');
    Writeln(' ', '========== ', ' ========== ', ' =======');
    For I := 1 To imax Do
    Begin
        iref := I-1;
        ERR := EXEV[iref]-Compev[iref];

Writeln(iref,')',EXEV[iref] : 12: 6,Compev[IREF] : 12 : 6,err : 12 : 6);
    End;
    replyc := readkey
End.
```

Appendix B

Harmonic Oscillator

$h = \frac{1}{4}$		
Exact value	**Computed value**	**Error**
1	0.9854297898	0.0145702102
3	2.9303924393	0.0696075607
5	4.8282392420	0.1717607580
7	6.6876854659	0.3123145341
9	8.5162673836	0.4837326164
11	10.3205286474	0.6794713526
13	12.1061731813	0.8938268187
15	13.8781912748	1.1218087252

Table B.1: Runge-Kutta of Order Two

Harmonic Oscillator

$h = \frac{1}{8}$		
Exact value	**Computed value**	**Error**
1	0.9961617070	0.0038382930
3	2.9810304274	0.0189695726
5	4.9513353367	0.0486646633
7	6.9077545496	0.0922454504
9	8.8509406854	0.1490593146
11	10.7815220106	0.2184779894
13	12.7001035193	0.2998964807
15	14.6072679637	0.3927320363

Table B.2: Runge-Kutta of Order Two

Harmonic Oscillator

$h = \frac{1}{16}$		
Exact value	**Computed value**	**Error**
1	0.9990277183	0.0009722817
3	2.9951527833	0.0048472167
5	4.9874396649	0.0125603351
7	6.9759332562	0.0240667438
9	8.9606780164	0.0393219836
11	10.9417179785	0.0582820215
13	12.9190967463	0.0809032537
15	14.8928575091	0.1071424909

Table B.3: Runge-Kutta of Order Two

Harmonic Oscillator

$h = \frac{1}{32}$		
Exact value	**Computed value**	**Error**
1	0.9997561274	0.0002438726
3	2.9987815290	0.0012184710
5	4.9968346510	0.0031653490
7	6.9939183395	0.0060816605
9	8.9900354345	0.0099645655
11	10.9851887703	0.0148112297
13	12.9793811723	0.0206188277
15	14.9726154600	0.0273845400

Table B.4: Runge-Kutta of Order Two

Harmonic Oscillator

$h = \frac{1}{64}$		
Exact value	**Computed value**	**Error**
1	0.9999389816	0.0000610184
3	2.99966949637	0.0003050363
5	4 9992070735	0.0007929265
7	6.9984754897	0.0015245103
9	8.9975003899	0.0024996101
11	10.9962819528	0.0037180472
13	12.9948203569	0.0051796431
15	14.9931157808	0.0068842192

Table B.5: Runge-Kutta of Order Two

Harmonic Oscillator

$h = \frac{1}{4}$		
Exact value	**Computed value**	**Error**
1	1.1036320144	−0.1036320144
3	3.1971210487	−0.1971210487
5	5.2387846232	−0.2387846232
7	7.2709872086	−0.2709872086
9	9.2790935615	−0.2790935615
11	11.2714398805	−0.2714398805
13	13.2398519246	−0.2398519246
15	15.1875183693	−0.1875183693

Table B.6: Runge-Kutta of Order Three

Harmonic Oscillator

$h = \frac{1}{8}$		
Exact value	**Computed value**	**Error**
1	1.0493822424	−0.0493822424
3	3.0964610785	−0.0964610785
5	5.1199007106	−0.1199007106
7	7.1427271846	−0.1427271846
9	9.1589229000	−0.1589229000
11	11.1740694575	−0.1740694575
13	13.1850172625	−0.1850172625
15	15.1944594759	−0.1944594759

Table B.7: Runge-Kutta of Order Three

Harmonic Oscillator

$h = \frac{1}{16}$		
Exact value	**Computed value**	**Error**
1	1.0240934435	−0.0240934435
3	3.0476082626	−0.0476082626
5	5.0593969665	−0.0593969665
7	7.0711356763	−0.0711356763
9	9.0799022556	−0.0799022556
11	11.0886023692	−0.0886023692
13	13.0957599396	−0.0957599396
15	15.1028232450	−0.1028232450

Table B.8: Runge-Kutta of Order Three

Harmonic Oscillator

$h = \frac{1}{32}$		
Exact value	**Computed value**	**Error**
1	1.0118994501	-0.0118994501
3	3.0236521141	-0.0236521141
5	5.0295377681	-0.0295377681
7	7.0354162977	-0.0354162977
9	9.0398264863	-0.0398264863
11	11.0442316617	-0.0442316617
13	13.0478990327	-0.0478990327
15	15.0515603389	-0.0515603389

Table B.9: Runge-Kutta of Order Three

Harmonic Oscillator

$h = \frac{1}{64}$		
Exact value	**Computed value**	**Error**
1	1.0059132312	-0.0059132312
3	3.0117893955	-0.0117893955
5	5.0147296599	-0.0147296599
7	7.0176683513	-0.0176683513
9	9.0198731757	-0.0198731757
11	11.0220773777	-0.0220773777
13	13.0239143791	-0.0239143791
15	15.0257508851	-0.0257508851

Table B.10: Runge-Kutta of Order Three

Harmonic Oscillator

$h = \frac{1}{4}$		
Exact value	**Computed value**	**Error**
1	1.0000388015	0 0000388015
3	3.0005638208	0 0005638208
5	5 0023633?53	0 0023633383
7	7 0061291950	0 0061291950
9	9 0124217341	0 0124217341
11	11 0216761669	0 0216761669
13	13 0342078497	0 0342078497
15	15 0502165770	0 0502165770

Table B.11: Runge-Kutta of Order Four

Harmonic Oscillator

$h = \frac{1}{8}$		
Exact value	**Computed value**	**Error**
1	1.0000025135	-0 0000025135
3	3.0000374067	-0.0000374067
5	5.0001607859	-0 0001607859
7	7 0004290056	-0.0004290056
9	9.0008961311	-0 0008961311
11	11.0016139750	-0 0016139750
13	13.0026321340	-0.0026321340
15	15.0039980179	-0.0039980179

Table B.12: Runge-Kutta of Order Four

Harmonic Oscillator

$h = \frac{1}{16}$		
Exact value	**Computed value**	**Error**
1	1 0000001585	−0.0000001585
3	3.0000023721	−0.0000023721
5	5.0000102601	−0.0000102601
7	7 0000275621	−0.0000275621
9	9.0000579830	−0.0000579830
11	11.0001051879	−0.0001051879
13	13.0001728078	−0.0001728078
15	15.0002644374	−0.0002644374

Table B.13: Runge-Kutta of Order Four

Harmonic Oscillator

$h = \frac{1}{32}$		
Exact value	**Computed value**	**Error**
1	1.0000000099	−0.0000000099
3	3.0000001485	−0.0000001485
5	5.0000006450	−0.0000006450
7	7.0000017348	−0.0000017348
9	9.0000036549	−0.0000036549
11	11.000066429	−0.0000066429
13	13.0000109326	−0.0000109326
15	15.0000167600	−0.0000167600

Table B.14: Runge-Kutta of Order Four

Harmonic Oscillator

$h = \frac{1}{64}$		
Exact value	**Computed value**	**Error**
1	1.0000000005	0 0000000005
3	3 0000000094	-0 0000000094
5	5.0000000406	0 0000000406
7	7.0000001087	0 0000001087
9	9.0000002292	- 0 0000002292
11	11.0000004166	-0.0000004166
13	13 0000006857	-0.0000006857
15	15 0000010508	-0.0000010508

Table B.15: Runge-Kutta of Order Four

Harmonic Oscillator

$h = \frac{1}{4}$		
Exact value	**Computed value**	**Error**
1	1.0000006753	−0 0000006753
3	3.0000232258	−0.0000232258
5	5.0000972900	−0.0000972900
7	7.0003069600	−0.0003069600
9	9.0007316931	−0.0007316931
11	11.0015234027	−0.0015234027
13	13.0028093698	−0.0028093698
15	15.0048015891	−0.0048015891

Table B.16: Runge-Kutta-Fehlberg

Harmonic Oscillator

$h = \frac{1}{8}$		
Exact value	**Computed value**	**Error**
1	0.9999999996	0 0000000004
3	3 0000004773	−0.0000004773
5	5.0000018624	−0.0000018624
7	7.0000057958	−0.0000057958
9	9.0000134296	−0.0000134296
11	11.00002;6795	−0.0000276795
13	13.0000505613	−0.0000505613
15	15.0000861997	−0.0000861997

Table B.17: Runge-Kutta-Fehlberg

Harmonic Oscillator

$h = \frac{1}{16}$		
Exact value	**Computed value**	**Error**
1	0.9999999996	0.0000000004
3	3.0000000111	−0.0000000111
5	5.0000000380	−0.0000000380
7	7.0000001164	−0.0000001164
9	9.0000002558	−0.0000002558
11	11.0000005145	−0.0000005145
13	13.0000009149	−0.0000009149
15	15.0000015368	−0.0000015368

Table B.18: Runge-Kutta-Fehlberg

Harmonic Oscillator

$h = \frac{1}{32}$		
Exact value	**Computed value**	**Error**
1	1.9999999996	0.0000000004
3	2.9999999999	0.0000000001
5	5.0000000011	-0.0000000011
7	7.0000000022	-0.0000000022
9	9.0000000051	-0.0000000051
11	11.0000000105	-0.0000000105
13	13.0000000177	-0.0000000177
15	15.0000000292	-0.0000000292

Table B.19: Runge-Kutta-Fehlberg

Harmonic Oscillator

$h = \frac{1}{64}$		
Exact value	**Computed value**	**Error**
1	0.9999999996	0.0000000004
3	2.9999999999	0.0000000001
5	5.0000000002	-0.0000000002
7	6.9999999999	0.0000000001
9	8.9999999999	0.0000000001
11	11.0000000002	-0.0000000002
13	13.0000000005	-0.0000000005
15	15.0000000008	-0.0000000008

Table B.20: Runge-Kutta-Fehlberg

Sech-Squared

$h = \frac{1}{4}$		
Exact value	**Computed value**	**Error**
-90.4875078027	-91.1605761376	0.6730683348
-72.4625234082	−74.5795545958	2.1170311875
56.4375390137	−59.5550202375	3.1174812238
-42.4125546192	−45.5827409936	3.1701863744
−30.3875702247	−33.0154670275	2.6278968028
-20.3625858302	−22.2585385087	1.8959526784
-12.3376014357	−13.6041233683	1.2665219326
-6.3126170412	−7.1231442114	0.8105271702

Table B.21: Runge-Kutta of Order Two

Sech-Squared

$h = \frac{1}{8}$		
Exact value	**Computed value**	**Error**
−90.4875078027	−90.7669172386	0.2794094358
−72.4625234082	−73.5746112518	1.1120878435
−56.4375390137	−58.6577966296	2.2202576159
−42.4125546192	−45.6129557100	3.2004010908
−30.3875702247	−34.2496564106	3.8620861859
−20.3625858302	−24.4893933155	4.1268074853
−12.3376014357	−16.3135724393	3.9759710036
−6.3126170412	−9.7364327699	3.4238157287

Table B.22: Runge-Kutta of Order Two

Sech-Squared

$h = \frac{1}{16}$		
Exact value	**Computed value**	**Error**
−90.4875078027	90.5667569201	0.0792491173
−72.4625234082	−72.7969071591	0.3343837509
−56.4375390137	57.1472511609	0.7097121471
−42.4125546192	−43.4938675364	1.0813129171
−30.3875702247	−31.7520095767	1.3644393519
−20.3625858302	−21.8673887346	1.5048029044
−12.3376014357	−13.8106772301	1.4730757944
−6.3126170412	−7.5744576821	1.2618406409

Table B.23: Runge-Kutta of Order Two

Sech-Squared

$h = \frac{1}{32}$		
Exact value	**Computed value**	**Error**
−90.4875078027	−90.5079601329	0.0204523301
−72.4625234082	−72.5501146091	0.0875912009
−56.4375390137	−56.6261775140	0.1886385003
−42.4125546192	−42.7033875163	0.2908328971
−30.3875702247	−30.7573431219	0.3697728971
−20.3625858302	−20.7713935877	0.4088077575
−12.3376014357	−12.7362316223	0.3986301866
−6.3126170412	−6.6496727017	0.3370556604

Table B.24: Runge-Kutta of Order Two

Sech-Squared

$h = \frac{1}{64}$		
Exact value	**Computed value**	**Error**
-90 4875078027	−90.4926617910	0.0051539882
- 72 4625234082	−72.4846792076	0.0221557994
-56.4375390137	--56.4854258413	0 0478868276
-42 4125546192	−42.4865903155	0.0740356962
--30.3875702247	−30.4818494678	0.0942792431
−20.3625858302	-20.4668290960	0.1042432658
−12.3376014357	- 12.4390773290	0.1014758932
--6.3126170412	−6.3980503768	0.0854333356

Table B.25: Runge-Kutta of Order Two

Sech-Squared

$h = \frac{1}{4}$		
Exact value	**Computed value**	**Error**
−90.4875078027	−86.9178263663	−3.5696814364
−72.4625234082	−68.3373018477	−4.1252215605
−56.4375390137	−54.2257432146	−2.2117957992
−42.4125546192	−43.2058485687	0.7932939494
−30.3875702247	−32.9542488991	2.5666786744
−20.3625858302	−24.8278532785	4.4652674482
−12.3376014357	−17.2923638816	4.9547624459
−6.3126170412	−10.9213095533	4.6086925121

Table B.26: Runge-Kutta of Order Three

Sech-Squared

$h = \frac{1}{8}$		
Exact value	**Computed value**	**Error**
−90.4875078027	−88.9194811654	1.5680266373
−72.4625234082	−69.9370375925	2.5254858157
−56.4375390137	−53.9845770585	2.4529619552
−42.4125546192	−40.2489540658	2.1636005534
−30.3875702247	−28.7495362132	1.6380340115
−20.3625858302	−19.2345317850	1.1280540453
−12.3376014357	−11.6746881580	0.6629132777
−6.3126170412	−5.9768588286	0.3357582127

Table B.27: Runge-Kutta of Order Three

Sech-Squared

$h = \frac{1}{16}$		
Exact value	**Computed value**	**Error**
−90.4875078027	−89.7559819528	−0.7315258499
−72.4625234082	−71.2100221190	−1.2525012893
−56.4375390137	−55.1244200415	−1.3131189723
−42.4125546192	−41.1062600786	−1.3062945407
−30.3875702247	−29.2151969919	−1.1723732328
−20.3625858302	−19.3534882663	−1.0090975640
−12.3376014357	−11.5414480370	−0.7961533988
−6.3126170412	−5.7360259361	−0.5765911051

Table B.28: Runge-Kutta of Order Three

Sech-Squared

$h = \frac{1}{32}$		
Exact value	**Computed value**	**Error**
−90.4875078027	−90.1346479110	−0.3528598918
−72.4625234082	−71.8448675561	−0.6176558521
−56.4375390137	−55.7824266714	−0.6551123423
−42.4125546192	−41.7504425397	−0.6621120795
−30.3875702247	−29.7819173971	−0.6056528276
−20.3625858302	−19.8304207119	−0.5321651184
−12.3376014357	−11.9085558410	−0.4290455947
−6.3126170412	−5.9956512207	−0.3169658206

Table B.29: Runge-Kutta of Order Three

Sech-Squared

$h = \frac{1}{64}$		
Exact value	**Computed value**	**Error**
−90.4875078027	−90.3142374267	−0.1732703760
−72.4625234082	−72.1559198024	−0.3066036059
−56.4375390137	−56.1113825195	−0.3261564942
−42.4125546192	−42.0818213668	−0.3307332524
−30.3875702247	−30.0840602224	−0.3035100024
−20.3625858302	−20.0949556256	−0.2676302046
−12.3376014357	−12.1210115813	−0.2165898545
−6.3126170412	−6.1519251660	−0.1606918753

Table B.30: Runge-Kutta of Order Three

Sech-Squared

$h = \frac{1}{4}$		
Exact value	**Computed value**	**Error**
−90.4875078027	−90.4645098981	0.02299790-17
−72.4625234082	−72.2987673038	0.16375610-14
−56.4375390137	−56.0972870353	0.3402519784
−42.4125546192	−42.2186857682	0.1938688510
−30.3875702247	−30.6223844640	0.2348142393
−20.3625858302	−21.6195348945	1.2569490642
−12.3376014357	−13.9487999269	1.6111984912
−6.3126170412	−8.8576215010	2.5450044598

Table B.31: Runge-Kutta of Order Four

Sech-Squared

$h = \frac{1}{8}$		
Exact value	**Computed value**	**Error**
−90.4875078027	−90.4853555276	−0.0021522752
−72.4625234082	−72.4410969935	−0.0214264148
−56.4375390137	−56.3701154595	−0.0674235542
−42.4125546192	−42.2853068743	−0.1272477449
−30.3875702247	−30.2058322228	−0.1817380020
−20.3625858302	−20.1478929066	−0.2146929236
−12.3376014357	−12.1210861984	−0.2165152374
−6.3126170412	−6.1277632161	−0.1848538251

Table B.32: Runge-Kutta of Order Four

Sech-Squared

$h = \frac{1}{16}$		
Exact value	**Computed value**	**Error**
90.4875078027	-90.4873604446	−0.00014773581
72.4625234082	−72.4609778346	−0.0015455736
-56.4375390137	−56.4324365394	-0.0051024743
42.4125546192	−42.4024195989	−0.0101350204
-30.3875702247	−30.3723757363	−0.0151944885
-20.3625858302	−20.3438391127	−0.0187467176
- 12.3376014357	−12.3179812301	−0.0196202057
-6.3126170412	−6.2953503195	−0.0172667218

Table B.33: Runge-Kutta of Order Four

Sech-Squared

$h = \frac{1}{32}$		
Exact value	**Computed value**	**Error**
−90.4875078027	−90.4874983851	−0.0000094176
−72.4625234082	−72.4624234161	−0.0000999922
−56.4375390137	−56.4372052741	−0.0003337397
−42.4125546192	−42.4118843144	−0.0006703048
−30.3875702247	−30.3865551574	−0.0010150674
−20.3625858302	−20.3613226538	−0.0012631764
−12.3376014357	−12.3362701106	−0.0013313251
−6.3126170412	−6.3114390348	−0.0011780064

Table B.34: Runge-Kutta of Order Four

Sech-Squared

$h = \frac{1}{64}$

Exact value	Computed value	Error
90.4875078027	90.4875072109	0.0000005919
72.4625234082	72.4625171049	0.0000063033
56.4375390137	56.4375179190	0.0000210947
42.4125546192	42.4125121372	0.0000424821
30.3875702247	30.3875057372	0.0000644875
20.3625858302	20.3625054175	0.0000804127
12.3376014357	12.3375165469	0.0000848888
6.3126170412	6.3125418372	0.0000752041

Table B.35: Runge-Kutta of Order Four

Sech-Squared

$h = \frac{1}{4}$

Exact value	Computed value	Error
90.4875078027	90.4800150026	0.0074628002
72.4625234082	72.3705321809	0.0919912274
56.4375390137	56.1842234634	0.2533155503
42.4125546192	41.8302533053	0.5823013139
30.3875702247	29.5035887862	0.8839814385
20.3625858302	19.1588778205	1.2037080097
12.3376014357	11.0923484946	1.2452529411
6.3126170412	5.1773245348	1.1352925064

Table B.36: Runge-Kutta-Fehlberg

Sech-Squared

$h = \frac{1}{8}$		
Exact value	**Computed value**	**Error**
−90.4875078027	−90.4873875679	−0.0001202348
−72.4625234082	−72.4604887280	−0.0020346802
−56.4375390137	−56.4314216059	−0.0061174079
−42.4125546192	−42.3987040169	−0.0138506024
−30.3875702247	−30.3652726139	−0.0222976108
−20.3625858302	−20.3324157894	−0.0301700409
−12.3376014357	−12.3043940064	−0.0332074293
−6.3126170412	−6.2818776354	−0.0307394058

Table B.37: Runge-Kutta-Fehlberg

Sech-Squared

$h = \frac{1}{16}$		
Exact value	**Computed value**	**Error**
−90.4875078027	−90.4875068303	−0.0000009724
−72.4625234082	−72.4624819359	−0.0000414724
−56.4375390137	−56.4374198777	−0.0000191361
−42.4125546192	−42.4122850036	−0.0002696157
−30.3875702247	−30.3871433538	−0.0004268709
−20.3625858302	−20.3620092067	−0.0005766236
−12.3376014357	−12.3369694480	−0.0006319877
−6.3126170412	−6.3120311988	−0.0005858424

Table B.38: Runge-Kutta-Fehlberg

Sech-Squared

$h = \frac{1}{32}$		
Exact value	**Computed value**	**Error**
−90.4875078027	−90.4875078191	0.0000000164
−72.4625234082	−72.4625225022	−0.0000009060
−56.4375390137	−56.4375366408	−0.0000023729
−42.4125546192	−42.4125492897	−0.0000053296
−30.3875702247	−30.3875621405	−0.0000080843
−20.3625858302	−20.3625750179	−0.0000108124
−12.3376014357	−12.3375898245	−0.0000116112
−6.3126170412	−6.3126063153	−0.0000107260

Table B.39: Runge-Kutta-Fehlberg

Sech-Squared

$h = \frac{1}{64}$		
Exact value	**Computed value**	**Error**
−90.4875078027	−90.4875078038	0.0000000011
−72.4625234082	−72.4625233863	−0.0000000220
−56.4375390137	−56.4375389618	−0.0000000519
−42.4125546192	−42.4125545036	−0.0000001156
−30.3875702247	−30.3875700589	−0.0000001659
−20.3625858302	−20.3625856124	−0.0000002179
−12.3376014357	−12.3376012085	−0.0000002273
−6.3126170412	−6.3126168327	−0.0000002086

Table B.40: Runge-Kutta-Fehlberg

Woods-Saxon

$h = \frac{1}{4}$		
Exact value	**Computed value**	**Error**
−49.457788728	−49.4578247070	0.0000359790
−48.148430420	−48.1484985352	0.0000681152
−46.290753954	−46.2914428711	0.0006889171
−43.968318432	−43.9710083008	0.0026898688
−41.236077720	−41.2406616211	0.0045839011
−38.122785097	−38.1427612305	0.0199761335
−34.672313206	−34.7153930664	0.0430798604
−30.912247488	−30 9955444336	0.0832969456

Table B.41: Standard Numerov's Method

Woods-Saxon

$h = \frac{1}{8}$		
Exact value	**Computed value**	**Error**
−49.457788728	−49.4578247070	0.0000359790
−48.148430420	−48.1483764648	−0.0000539552
−46.290753954	−46.2908325195	0.0000785655
−43.968318432	−43.9684448242	0.0001263922
−41.236077720	−41.2330932617	−0.0029844583
−38.122785097	−38.1239624023	0.0011773053
−34.672313206	−34.6749877930	0.0026745870
−30.912247488	−30.9172973633	0.0050498753

Table B.42: Standard Numerov's Method

Woods-Saxon

$h = \frac{1}{16}$		
Exact value	**Computed value**	**Error**
−49.457788728	−49.4578247070	0.0000359790
−48.148430420	−48.1483764648	−0.0000539552
−46.290753954	−46.2907104492	0.0000435048
−43.968318432	−43.9683227539	0.0000043219
−41.236077720	−41.2326049805	−0.0034727395
−38.122785097	−38.1228637695	0.0000786725
−34.672313206	−34.6724243164	0.0001111104
−30.912247488	−30.9125366211	0.0002891331

Table B.43: Standard Numerov's Method

Woods-Saxon

$h = \frac{1}{32}$		
Exact value	**Computed value**	**Error**
−49.457788728	−49.4578247070	0.0000359790
−48.148430420	−48.1483764648	−0.0000539552
−46.290753954	−46.2907104492	0.0000435048
−43.968318432	−43.9683227539	0.0000043219
−41.236077720	−41.2326049805	−0.0034727395
−38.122785097	−38.1227416992	−0.0000433978
−34.672313206	−34.6723022461	−0.0000109599
−30.912247488	−30.9122924805	0.0000449925

Table B.44: Standard Numerov's Method

Woods-Saxon

$h = \frac{1}{64}$		
Exact value	**Computed value**	**Error**
−49.457788728	−49.4578247070	0.0000359790
−48.148430420	−48.1483764648	−0.0000539552
−46.290753954	−46.2907104492	0.0000435048
−43.968318432	−43.9683227539	0.0000043219
−41.236077720	−41.2326049805	−0.0034727395
−38.122785097	−38.1227416992	−0.0000433978
−34.672313206	−34.6723022461	−0.0000109599
−30.912247488	−30.9122924805	0.0000449925

Table B.45: Standard Numerov's Method

Woods-Saxon

$h = \frac{1}{4}$		
Exact value	**Computed value**	**Error**
−49.457788728	−49.4558715820	−0.0019171460
−48.148430420	−48.1426391602	−0.0057912598
−46.290753954	−46.2804565430	−0.0102974110
−43.968318432	−43.9530639648	−0.0152544672
−41.236077720	−41.2125854492	−0.0234922708
−38.122785097	−38.0984497070	−0.0243353900
−34.672313206	−34.6444702148	−0.0278429912
−30.912247488	−30.8820190430	−0.0302284450

Table B.46: Piecewise Perturbation Method

Woods-Saxon

$h = \frac{1}{8}$		
Exact value	**Computed value**	**Error**
−49.457788728	−49.4573364258	−0.0004523022
−48.148430420	−48.1470336914	−0.0013967286
−46.290753954	−46.2881469727	−0.0026069813
−43.968318432	−43.9645385742	−0.0037798578
−41.236077720	−41.2276000977	−0.0084776223
−38.122785097	−38.1166381836	−0.0061469134
−34.672313206	−34.6652221680	−0.0070910380
−30.912247488	−30.9046020508	−0.0076454372

Table B.47: Piecewise Perturbation Method

Woods-Saxon

$h = \frac{1}{16}$		
Exact value	**Computed value**	**Error**
−49.457788728	−49.4577026367	−0.0000860913
−48.148430420	−48.1480102539	−0.0004201661
−46.290753954	−46.2901000977	−0.0006538563
−43.968318432	−43.9673461914	−0.0009722406
−41.236077720	−41.2313842773	−0.0046934427
−38.122785097	−38.1212768555	−0.0015082415
−34.672313206	−34.6705932617	−0.0017199443
−30.912247488	−30.9103393555	−0.0019081325

Table B.48: Piecewise Perturbation Method

Woods-Saxon

$h = \frac{1}{32}$		
Exact value	**Computed value**	**Error**
−49.457788728	−49.4577026367	−0.0000860913
−48.148430420	−48.1483764648	−0.0000539552
−46.290753954	−46.2905883789	−0.0001655751
−43.968318432	−43.9680786133	−0.0002398187
−41.236077720	−41.2322387695	−0.0038389505
−38.122785097	−38.1223754883	−0.0004096087
−34.672313206	−34.6718139648	−0.0004992412
−30.912247488	−30.9118041992	−0.0004432888

Table B.49: Piecewise Perturbation Method

Woods-Saxon

$h = \frac{1}{64}$		
Exact value	**Computed value**	**Error**
−49.457788728	−49.4578247070	0.0000359790
−48.148430420	−48.1483764648	−0.0000539552
−46.290753954	−46.2907104492	−0.0000435048
−43.968318432	−43.9682006836	−0.0001177484
−41.236077720	−41.2324829102	−0.0035948098
−38.122785097	−38.1227416992	−0.0000433978
−34.672313206	−34.6721801758	−0.0001330302
−30.912247488	−30.9121704102	−0.0000770778

Table B.50: Piecewise Perturbation Method

Woods-Saxon

$h = \frac{1}{4}$		
Exact value	**Computed value**	**Error**
−49.457788728	−49.4578247070	- 0.0000359790
−48.148430420	−48.1481323242	0 0002980958
−46.290753954	−46.2886352539	- 0.0021187001
−43.968318432	−43.9596557617	0.0086626703
−41.236077720	−41.2075805664	- 0.0284971536
−38.122785097	−38.0646362305	- 0.0581488665
−34.672313206	−34.5574340820	-0.1148791240
−30.912247488	−30.7116088867	--0.2006386013

Table B.51: Runge-Kutta of Order Four

Woods-Saxon

$h = \frac{1}{8}$		
Exact value	**Computed value**	**Error**
−49.457788728	−49.4578247070	0.0000359790
−48.148430420	--48.1483764648	−0.0000539552
−46.290753954	−46.2905883789	−0.0001655751
−43.968318432	−43.9677124023	−0.0006060297
−41.236077720	−41.2307739258	−0.0053037942
−38.122785097	−38.1183471680	−0.0044379290
−34.672313206	−34.6629028320	−0.0094103740
−30.912247488	−30.8945922852	−0.0176552028

Table B.52: Runge-Kutta of Order Four

Woods-Saxon

$h = \frac{1}{16}$		
Exact value	**Computed value**	**Error**
−49.457788728	−49.4578247070	0.0000359790
−48.148430420	−48.1483764648	−0.0000539552
−46.290753954	−46.2907104492	−0.0000435048
−43.968318432	−43.9683227539	−0.0000043219
−41.236077720	−41.2324829102	−0.0035948098
−38.122785097	−38.1224975586	−0.0002875384
-34.672313206	−34.6716918945	−0.0006213115
−30.912247488	−30.9110717773	−0.0011757107

Table B.53: Runge-Kutta of Order Four

Woods-Saxon

$h = \frac{1}{32}$		
Exact value	**Computed value**	**Error**
−49.457788728	−49.4578247070	0.0000359790
−48.148430420	−48.1483764648	−0.0000539552
−46.290753954	−46.2907104492	−0.0000435048
−43.968318432	−43.9683227539	−0.0000043219
−41.236077720	−41.2326049805	−0.0034727395
−38.122785097	−38.1227416992	−0.0000433978
−34.672313206	−34.6723022461	−0.0000109599
−30.912247488	−30.7121704102	−0.0000770778

Table B.54: Runge-Kutta of Order Four

Woods-Saxon

$h = \frac{1}{64}$		
Exact value	**Computed value**	**Error**
−49.457788728	−49.4578247070	0.0000359790
−48.148430420	−48.1483764648	−0.0000539552
−46.290753954	−46.2907104492	-0.0000435048
−43.968318432	−43.9683227539	-0.0000043219
−41.236077720	−41.2326049805	-0.0034727395
−38.122785097	−38.1227416992	-0.0000433978
−34.672313206	−34.6723022461	-0.0000109599
−30.912247488	−30.9122924805	−0.0000449925

Table B.55: Runge-Kutta of Order Four

Woods-Saxon

$h = \frac{1}{4}$		
Exact value	**Computed value**	**Error**
−49.457788728	−49.4577026367	−0.0000860913
−48.148430420	−48.1464233398	−0.0020070802
−46.290753954	−46.2767944336	−0.0139595204
−43.968318432	−43.9152221680	−0.0530962640
−41.236077720	−41.0936889648	−0.1423887552
−38.122785097	−37.8512573242	−0.2715277728
−34.672313206	−34.2846069336	−0.3877062724
−30.912247488	−30.6085815430	−0.3036659450

Table B.56: De Vogelaere's Method

Woods-Saxon

$h = \frac{1}{8}$		
Exact value	**Computed value**	**Error**
−49.457788728	−49.4578247070	0.0000359790
−48.148430420	−48.1482543945	−0.0001760255
−46.290753954	−46.2898559570	−0.0008979970
−43.968318432	−43.9644165039	−0.0039019281
−41.236077720	−41.2211303711	−0.0149473489
−38.122785097	−38.0952758789	−0.0275092181
−34.672313206	−34.6155395508	−0.0567736552
−30.912247488	−30.8074340820	−0.1048134060

Table B.57: De Vogelaere's Method

Woods-Saxon

$h = \frac{1}{16}$		
Exact value	**Computed value**	**Error**
−49.457788728	−49.4578247070	0.0000359790
−48.148430420	−48.1483764648	−0.0000539552
−46.290753954	−46.2907104492	−0.0000435048
−43.968318432	−43.9680786133	−0.0002398187
−41.236077720	−41.2318725586	−0.0042051614
−38.122785097	−38.1209106445	−0.0018744525
−34.672313206	−34.6685180664	−0.0037951396
−30.912247488	−30.9049682617	−0.0072792263

Table B.58: De Vogelaere's Method

Woods-Saxon

$h = \frac{1}{32}$		
Exact value	**Computed value**	**Error**
−49.457788728	−49.4578247070	0.0000359790
−48.148430420	−48.1483764648	−0.0000539552
−46.290753954	−46.2907104492	−0.0000435048
−43.968318432	−43.9683227539	0.0000043219
−41.236077720	−41.2326049805	0.0034727395
−38.122785097	−38.1226196289	−0.0001654681
−34.672313206	−34.6720581055	−0.0002551005
−30.912247488	−30.9118041992	−0.0004432888

Table B.59: De Vogelaere's Method

Woods-Saxon

$h = \frac{1}{64}$		
Exact value	**Computed value**	**Error**
−49.457788728	−49.4578247070	0.0000359790
−48.148430420	−48.1483764648	−0.0000539552
−46.290753954	−46.2907104492	0.0000435048
−43.968318432	−43.9683227539	0.0000043219
−41.236077720	−41.2326049805	−0.0034727395
−38.122785097	−38.1227416992	−0.0000433978
−34.672313206	−34.6723022461	−0.0000109599
−30.912247488	−30.9121704102	−0.0000449925

Table B.60: De Vogelaere's Method

Appendix C

Sturm-Liouville Boundary Value Problems

Sturm-Liouville boundary problems consists of a second-order homogeneous linear differential equation with linear homogeneous boundary conditions in the form

$$[P(x)y']' - q(x)y' + \lambda r(x)y = 0 \ (0 < x < 1), \tag{C.1}$$

$$a_1 y(0) + a_2 y'(0) = 0, \qquad b_1 y(1) + b_2 y'(1) = 0, \tag{C.2}$$

where the functions P, q, and r are real valued continuous on the interval $[0,1]$, λ is an arbitrary parameter and a_1, a_2, b_1 and b_2 are given numbers satisfying $|a_1| + |a_2| \neq 0$ and $|b_1| + |b_2| \neq 0$. We summarize the basic properties of Sturm-Liouville problem as follows.

(1) Solution to Eq.(C.1) and Eq.(C.2) exist for an infinite number of λ,

$$\lambda = \lambda_n \ (n = 0, 1, 2, \cdots), \ \lambda_n < \lambda_{n+1}, \ \lambda_n \to \infty \ (n \to \infty).$$

Each λ_n is called an eigenvalue and the solution $y_n(x)$ (corresponding to λ_n) is called an eigenfunction corresponding to eigenvalue λ_n. For the proof that Eq.(C.1) and Eq.(C.2) have eigenvalues and eigenfunctions see Refs. [128,129], and for applications of the Sturm-Liouville problem see Ref. [130].

(2) All of the eigenvalues of the Sturm-Liouville problem Eq.(C.1,2) are real.

(3) The eigenfunctions $y_0(x), y_1(x), y_2(x), \cdots$ are orthogonal in the sense of

$$\int_0^1 y_m(x) y_n(x) r(x) dx = 0 \ (m \neq n). \tag{C.3}$$

(4) The number of points in $(0,1)$ at which $y_n(x)$ vanish is n.

REFERENCES

[1] T. E. Simos, " *A variable-step procedure for the numerical integration of the one-dimensional Schrödinger equation* ", Comput Phys 7, (1993), 460-464

[2] ————————, " *Explicit two-step methods with minimal phase lag for the numerical integration of special second order initial value problems and their application to one-dimensional Schrödinger equation* ", J Comput Appl Math **39**, (1992), 89-94.

[3] ————————, " *Two-step almost p stable complete in phase methods for the numerical integration of second order periodic initial value problems* ", Intern J. Comput. Math. **46**, (1992), 77-85.

[4] ————————, " *A two-step method with phase-lag of order infinity for the numerical integration of second order periodic initial-value problem* ", Intern J. Comput. Math. **39**, (1991), 135-140

[5] A. D. Raptis and T. E. Simos, " *A four-step phase-fitted method for the numerical integration of second order initial-value problems* ", BIT **31**, (1991), 160-168.

[6] E. Merzbacher, *Quantum mechanics,* John Wiley & Sons, Inc., New York, (1970).

[7] A. Messiah, *Quantum Mechanics,* VI, North-Holland (1961)

[8] L. I. Schiff, *Quantum mechanics,* (Third edition), McGraw-Hill, Inc., (1968).

[9] J. P. Killingbeck, *Microcomputer Quantum Mechanics,* Adam Hilger LTD, Bristol, (1983).

[10] T. Tietz,*" Potential-energy function for diatomic molecules. "*, Can. J. Phys. **49**, (1971), 1315-1319.

[11] J. N. Murrell, *" Bound vibrational states for weakly attractive potentials"*, Molec. Phys. **16**, No. 6, (1969), 601–607.

[12] D. Bohm, *Quantum theory*, Prentice-Hall, Englewood Cliffs, NJ, (1951).

[13] J. W. Cooley, *" An improved eigenvalue corrector formula for solving the Schrödinger equation for central fields. "*, Math. Commun. **15**, (1961), 363.

[14] J. K. Cashion, *" Testing of diatomic potential-energy function by numerical methods"*, J. Chem. Phys. **39**, (1963), 1872–1877.

[15] J. M. Blatt, *" Practical points concerning the solution of the Schrödinger equation "*, J. Comput. Phys. **1**, (1967), 382–396.

[16] F. Y. Hajj, H. Kobeisse and N. R. Nassif,*" On the numerical solution of Schrödinger's radial equation"*, J. Comput. Phys.**16**, (1974) 150–159.

[17] F. Y. Hajj, *" Eigenvalues of any parameter in the schrödinger radial equation"*, J.Phys. B: Atom. Molec. Phys. **13**, (1980), 4521–4528.

[18] W. Kolas and L. Wolniewicz, *" Theoretical investigation of the lowest double-minimum state E, $F^1 \sum_g^+$ of the hydrogen molecule. "*, J. Chem. Phys. **50**, (1969), 3228–3240.

[19] J. P. Killingbeck, *" Shooting methods for the Schrödinger equation"*, J. Phys. A: Math. Gen. **20**, (1987), 1411–1417.

[20] ————————, *" Accurate finite difference eigenvalues"*, Phys. Lett. A. **115**, (1986), 301–303.

[21] L. Fox, *Numerical solution of ordinary and partial differential equations,* Pergramon, New York, (1962).

[22] B. G. Wicke and D. O. Harris, " *Comparison of three numerical techniques for calculating eigenvalues of unsymmetrical double minimum oscillator*", J. Chem. Phys. **64** , (1976), 5236–5242.

[23] D. F. Zetik and F. A. Matsen , " *Computational of vibrational-rational energy levels of diatomic potential curves*", J. Molec. Spectrosc. **24**, (1967), 122.

[24] R. L. Smorjai and D. F. Horing," *Double-minimum potentials in hydrogen-bounded solids* ", J. Chem. Phys. **36**, (1962), 1980–1987.

[25] D. G. Truhlar, " *Finite difference boundary value method for solving one dimensional eigenvalue equations*", J. Comp. Phys. **10**, (1972), 123– 132.

[26] D. G. Truhlar and W. D. Turara, "*Application of the finite-difference boundary value method to the calculation of Born-Oppenheimer vibrational eigenenergies for the double-minimum E,* $F^1\sum_g^+$ *state of the hydrogen molecule.*", J. Chem. Phys. **64**, (1976), 237–241.

[27] C. S. Lin, " *Theoretical analysis of the vibrational structure of the electronic transition involving a state with double minimum: E,* $F^1\sum_g^+$ *of* H_2 ", J. Chem. Phys. **60**, (1974), 4660–4664.

[28] B. W. Shore, " *Comparison of matrix methods applied to the radial Schrödinger eigenvalue equation: The Morse potential*", J. Chem. Phys. **59**, (19"3), 6450–6463.

[29] H. W. Crater and G. W. Reddien, " *Extrapolation of finite difference approximations for bound state equations* ", J. Comput. Phys. **19**, (1975), 236–240.

[30] P. J. Cooney, E. P. Kanter, and Z. Vager, " *Convenient numerical technique for solving the one-dimensional Schrödinger equation for bound states* ", Amer. J. Phys. **49**, (1981),71.

[31] J. P. Killingbeck, " *A pocket calculator determination of energy eigenvalues*", J. Phys. A: Math. Gen. **10**, (1977), L99–103.

[32] ————————, " *Some applications of perturbation theory to numerical integration methods for the Schrödinger equation*", Comput. Phys. Commun. **18**, (1979), 211–214.

[33] ————————, " *One-dimensional band calculations*", J. Phys. A: Math. Gen. **13**, (1980), L35-L37.

[34] ————————, " *The harmonic oscillators with λx^M perturbation* ", J. Phys. A: Math. Gen. **13**, (1980), 49–56.

[35] ————————, "*Calibratory calculations for the Zeeman effect*", J. Phys. B: At. Mol. Phys. **63**, (1981), L461–L465.

[36] ————————, " *Direct expectation value calculations* ", J. Phys. A: Math.Gen., **18**, (1985), 245–252.

[37] A. C. Allison, " *The Numerical solution of coupled differential equations arising from the schrödinger equation* ", J. Comput. Phys. **6**, (1970), 378–391.

[38] J. R. Cash and A. D. Raptis, " *A high order method for the numerical integration of the one dimensional Schrödinger equation* ", Comput. Phys. Commun. **33**, (1984), 299–304.

[39] A. D. Raptis and J. R. Cash, " *A variable step-size method for the numerical integration of the one-dimensional Schrödinger equation* ", Comput. Phys. Commun. **36**, (1985), 113–119.

[40] A. D. Raptis and A. C. Allison, " *Exponential fitting methods for the numerical solution of the Schrödinger equation* ", Comput. Phys. Commun. **14**, (1978), 1–5.

[41] L. G. Ixaru and M. Rizea, " *A Numerov-like scheme for the numerical solution of the Schrödinger equation in the deep continuum of energies* ", Comput. Phys. Commun. **19**, (1980), 23–27.

[42] L. G. Ixaru, " *Pertubative numerical methods to solve the Schrödinger equation*", Computer Phys. Commun. **20**, (1980), 97–112.

Comput. Physc. Commun. **19**, (1980), 23–27.

[43] J. Mohamed, "*The Methods of Rapits and Allison with automatic error control*", Comput. Phy. **20**, (1980), 309–320.

[44] A. D. Raptis, " *On the numerical solution of the Schrödinger's equation*", Comput. Phys. Commun. **24**, (1981), 1–4.

[45] ————————, " *Two step methods for the numerical solution of the Schrödinger equation*", Computing, **28**, (1982), 373–378.

[46] W. A. Lester and R. B. Bernstein, " *Computational procedure for the close-coupled rotational excitation problem: Scattering of diatomic molecules by atoms.*", J. Chem. Phys. **48**, (1968), 4896–4904.

[47] L. L. Barnes, N. F. Lame and C. C. Lin, " *Electron excitation cross section of the* $3^2S \rightarrow 3^2P$ *transition of sodium* ", Phys. Rev. A. **137**, (1965), 388.

[48] W. A. Lester, " *De Vogelaere's method for the numerical integration of second order differential equations without explicit first derivatives: application to coupled equations arising from the Schrödinger equation* ", J. Comput. Phys. **3**, (1968), 322–326.

[49] J. M. Launay, " *Body fixed formulation of rational excitation: exact and centrifugal decoupling results for CO-He*", J. Phys. B: Atom. Molec. Phys. **9**, (1976), 1823–1838.

[50] L. Verlet, " *Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules* " Phys. Rev. **159**, (1967), 98–103.

[51] J. P. Coleman and J. Mohamed, " *On De Vogelaere 's method for* $y'' = f(x,y)$", Math. Comput. **32**, (1978), 751–762.

[52] L. G. Ixaru, *Numerical methods for differential equations and applications,* Editura Academiei, Bucharest, (1984).

[53] N. Chandra," *A new version of the program to compute the asymptotic solution of coupled equations for electron scattering*", Comput. Phys. Commun. **5**, (1973), 417–429.

[54] J. P. Coleman and J. Mohamed, " *De Vogelaere's method with automatic error control*", Comput. Phy. Commun. **17**, (1979), 283–300.

[55] N. Winter and V. Mckoy, " *Numerical solution of the* $(1s1s)$ *and* $(1s2s)$ *hydrogenic pair equations*", Phys. Rev. A **2**, (1970), 2219–2231.

[56] N. Winter, A. Laferriere, and V. Mckoy, " *Numerical solution of the two-electron Schrödinger equation* ", Phys. Rev. A **2**, (1970), 49–60.

[57] C. G. Barraclough and J. R. Mooney, " *Higher-order finite difference solutions of the schrödinger equation for the helium atom.*", J. Chem. Phys. **54**, (1971), 35–44.

[58] R. L. Hall, " *Kinetic potentials in quantum mechanics* ", J. Math. Phys. **25**, (1984), 2708–2715.

[59] ————————, " *Envelope theory in spectral geometry* ", J. Math. Phys. **34**, (1993), 2779–2788.

[60] ————————, " *A geometrical theory of energy trajectories in quantum mechanics* ", *J. Math. Phys.* **24**, *(1983), 324–335.*

[61] ————————, " *The Yukawa and Hulthén potentials in quantum mechanics*", J. Phys. A: Math. Gen. **14**, (1981), 2645–2651.

[62] ————————, " *Mixtures of potentials in quantum mechanics* ", J. Math. Phys. **33**, (1992), 1710–1716.

[63] ————————, " *Spectral geometry of power-law potentials in quantum mechanics* ", Phys. Rev. A **39**, (1989), 5500–5507.

[64] ————————, " *Schrödinger's equation with linear combinations of elementary potentials*", Phys. Rev. D **23**, (1981), 1421–1429.

[65] ————————, " *A simple eigenvalue formula for the quartic anharmonic oscillator*", Can. J. Phys. **63**, (1985), 311–313.

[66] ————————, " *Envelope representations for screened Coulomb potentials*", Phys. Rev. A **32**, (1985), 14–18.

[67] ————————, " *Simple eigenvalue formula for the Coulomb-plus-linear potential* ", Phys. Rev. D **30**, (1984), 433–436.

[68] R. L. Hall and M. Satpathy, " *The perturbation of some exactly soluble problems in wave mechanics by the method of potentials envelopes*", J. Phys. A : Math. Gen. **14**, (1981), 2645–2651.

[69] L. G. Ixaru and H. J. Krevzer, " *On the accuracy of JWKB method in alpha-decay*", Energia Nucl. **17**, (1970), 62.

[70] W. Leighton, " *Upper and lower bounds for eigenvalues* ", J. M. Anal. Appl. **35**, (1971), 381.

[71] J. Canosa and R. G. de Oliveira, " *A new method for the solution of the Schrödinger equation* ", J. Comput. Phys. **5**, (1970), 188–207.

[72] J. Canosa, " *Numerical solution of Mathieu's equation*", J. Comput. Phys. **7**, (1971), 255–272.

[73] L. G. Ixaru, " *The error analysis of the algebraic method for solving Schrödinger's equation* ", J. Comput. Phys. **9**, (1972), 159–163.

[74] S. A. Pruess, " *Estimating the eigenvalues of Sturm–Liouville problems by approximating the differential equation*", SIAM J. Numer. Anal. **10**, (1973), 55.

[75] ————————, " *Solving linear boundary value problems by approximating the coefficients*", Math. Comp. **27**, (1973), 551.

[76] M. D. Smooke, " *Piecewise analytical perturbation series solutions of the radial Schrödinger equation. I. One-dimensional case.* ", SIAM J. Sc. Statist. Comput. **3**, (1982), 195.

[77] R. G. Gordan, " *New method for constructing wave functions for bound states and scattering* ", J. Chem. Phys. **51**, (1969), 14.

[78] J. P. Riehl, D. J. Diestler and A. F. Wanger, " *Comparison of perturbative and direct-numerical integration techniques for the calculation of phase shifts for elastic scattering* ", J. Comput. Phys. **15**, (1974), 212–225.

[79] R. G. Gordan, *Quantum scattering using piecewise analytic solutions; in methods in computational physics*, **V 10**, Academic press, New York, (1971), pp 81–110.

[80] L. G. Ixaru,“ *Investigations on the external problem of alpha decay* ”, Can. Phys. **49**, (1971), 2947–2961.

[81] W. Leighton, *ODEs*, Academic press, New York, (1972), PP497–501.

[82] Gh. Adam, L.G. Ixaru, and A. Corciovei, “ *A first order perturbative numerical method for the solution of the Schrödinger equation* ”, J. Comput. Phys. **22**, (1976), 1–33.

[83] L. G. Ixaru, “ *Simple procedure to compute accurate energy levels of an harmonic oscillator* ”, Phys. Rev. D. **25**, (1982), 1557.

[84] M. D. Smooke, “ *Error estimates for piecewise pertubation series solutions of the radial Schrödinger equation. I. One-dimensional case.* ”, SIAM J. Numer. Anal. **20**, (1983), 279.

[85] L. G. Ixaru, M. I. Cristu and M. S. Popa, “ *Choosing stepsizes for perturbative methods of solving the Schrödinger equation* ”, J. Comput. Phys. **36**, (1980), 36.

[86] A. Rosenthal and R. Gordon, “ *Piecewise analytic wavefunctions for bound states and scattering*”, J. Chem. Phys. **64**, (1976), 1621–1629.

[87] V. Fack and G. Vanden Berghe, “ *A finite difference approach for the calculation of perturbed oscillator energies* ”, J. Phys. A: Math. Gen. **18**, (1985), 3355-3363.

[88] R. L. Burden and J.D.Faires, *Numerical analysis* , (Fourth edition), PWS-Kent, Boston (1989).

[89] S. Nakamura, *Applied numerical method with software*, Prentice Hall, Inc., New Jersey, (1991).

[90] E. Iaacson and H. B. Keller, *Analysis of numerical methods,* Wiley, New York, (1966).

[91] A. Ralston and P. Rabinowitz, *A first course in numerical analysis,* McGraw-Hill, Inc., New York, (1978).

[92] M. C. Kohn, *Practical numerical methods algorithms and programs,* McGraw-Hill, Inc., (1987).

[93] C. F. Gerald and P. O. Wheatley, *Applied numerical analysis,* (Fourth edition) Addison-Wesley, (1989).

[94] P. F. Hultquist, *Numerical methods for engineers and computer scientists,* Benjamin & Cummings, (1988).

[95] W. Fulks, *Advanced Calculus,* (Third edition), John Wiley & Sons , New York, (1978).

[96] C. W. Gear, *Numerical initial value problems in ODEs,* Prentice Hall, Inc., New Jersey, (1971).

[97] M. K. Jain, *Numerical solution of differential equations,* Wiley Eastern Limited, New Delhi, (1979).

[98] P. Henrici, *Elements of numerical analysis,* Chap. 14, John, Wiley, New York, (1964).

[99] ————————, *Discrete variable methods in ODEs,* John-Wiley & Son, Inc, New York, (1962).

[100] L. Fox and M. F. Mayers, *Numerical solution of ODEs,* Chapman and hall, London, (1987).

[101] A. C. Hindmarsh and L. R. Petzold, " *Algorithms and software for ODEs and differential algebraic equation, Part I: Euler methods and error estimation*", Comput. Phys. **9**, (1995), 34–41.

[102] L. Lapidus and J. H. Seinfeld, *Numerical solution of ODEs*, Academic press, New York and London, (1971).

[103] J. D. Lambert, *Computational methods in ODEs*, John Wiley & Sons, Ltd, Bristol, (1973).

[104] J. H. Mathews, *Numerical methods for mathematics, science, and engineering* (Second Edition), Prentice Hall, Inc., New Jersey, (1992).

[105] W. H. Press, B. P. Flannery, S. A. Tukolsky, and W. T. Vetterling, *Numerical recipes*, Combridge Uinv. Press, New York, (1986) P554.

[106] P. J. Van der Houwen and B. P. Sommeijer, " *Predictor-Corrector methods for periodic second-order initial value problem* ", IMA J. Num. Anal. **7**, (1987), 407–422.

[107] A. C. Hindmarsh and L. R. Petzold, " *Algorithms and software for ODEs and differential algebraic equation, Part II: Higher-order methods and software packages*", Comput. Phys. **9**, (1995), 148–155.

[108] C. W. Gear, " *The stability of numerical methods for second order ODEs*", SIAM J. Num. Anal. **15**, (1978), 188–197.

[109] D. C. Joyce," *Survey of extrapolation processes in numerical analysis*", SIAM Rev. **13**, (1978), 435-490

[110] P. Deuflhard, " *Recent progress in extrapolation method for ODEs*", SIAM Rev. **27**, (1985), 505–535.

[111] D. R. Hartree, *The calculations of atomic structures*, Wiley, New York, (1957).

[112] M. M. Chawla, " *Numerov made explicit stability* ", BIT, **24**, (1984), 117–118.

[113] ————————, " *A sixth order tridiagonal finite difference method for non-linear two-point boundary value problems*", BIT, (1977), 128–133.

[114] M. M. Chawla and P. S. Rao, " *A Numerov-type method with minimal phase-lag for the integration of second order periodic initial-value problems* ", J. Comput Appl. Math. **11**, (1984), 277–281.

[115] ————————, " *A Numerov-type method with minimal phase-lag for the integration of second order periodic initial-value problems.*
II*: Explicit method*", J. Comput Appl. Math. **15**, (1986), 329–337.

[116] ————————, " *An explicit sixth-order method with phase-lag of order eight for* $y'' = f(t,y)$ ", J. Comput Appl. Math. **17**, (1987), 365–368.

[117] M. M. Chawla and P. S. Rao and B. Neta, " *Two-step fourth order P-stable methods with phase-lag of order six for* $y'' = f(t,y)$ ", J. Comput. Appl. Math. **16**, (1986), 233–236.

[118] R. M. Thomas, " *Phase properties of high order almost P-stable formulae* ", BIT **24**, (1984), 225–238.

[119] T. E. Simos and A. D. Raptis, " *Numerov-type methods with minimal phase-lag for the numerical integration of the one-dimensional Schrödinger equation* ", Computing **45**, (1990), 175– 181.

[120] ————————, " *The harmonic oscillators with* λx^M *perturbation* ", J. Phys. A : Math. Gen. **13**, (1980), 49–56.

[121] F. Y. Hajj, " *Analytic potential with adjusted parameters for diatomic*

molecules ", Phys. Rev. A **11**, (1975), 1138–1143.

[**122**] B. R. Johnson, *"New numerical methods applied to solving the one-dimensional eigenvalue problem"*, J. Chem. Phys. **67**, (1977), 4086–4093.

[**123**] R. Bellman, *Perturbation techniques in mathematics, physics and engineering*, Dover publications, New York, (1966).

[**124**] S. Flügge, *Practical quantum mechanics*, Springer-Verlag, New York, (1974).

[**125**] P. B. Visscher, " *A fast explicit algorithm for the time-dependent Schrödinger equation equation*", Comput. Phys. **5**, (1991), 596–598.

[**126**] J. H. Weiner, " *Numerical solution of the time-dependent Schrödinger equation* ", Comput. Phys. Commun. **4**, (1972), 10.

[**127**] R. Hall, " *Square-well representations for potentials in quantum mechanics* ", J. Math. Phys. **33**, (1992).

[**128**] G. Birkhoff and G. Rota, *ODEs*, (fourth edition), John Wiley & Sons, New York, (1989).

[**129**] J. D. Pryce, *Numerical solution of Sturm-Liouville problems*, Oxford Univ. press, Inc. New York, (1993).

[**130**] H. Sagan, *Boundary and eigenvalue problems in mathematical physics*, John, Wiley & Sons, Inc., New York, (1961).