# CO22: Solutions of Schrödinger's Equation by Numerical Integration

A. Kulanthaivelu
University College

## 1 Abstract

The aim of the experiment was to determine the solution to the time-independent Schrödinger equation for the quantum harmonic oscillator. We first solve the equation numerically using Numerov's method and compare the solution to the analytic solution finding the numerical solution to follow the analytic one very closely, although differently scaled. We also find the numerical solution fails at large values of $\hat{x}$ but this can be compensated for by using a smaller spacing term. We then use a similar program to determine the energy eigenvalues for the $n^{\text{th}}$ eigenstate using an interval bisection method to arrive at an answer which is correct to at least four decimal places when using a trial value within 2.0 of the eigenvalue and other suitable input parameters.

## 2 Introduction

The time-independent Schrödinger equation in one dimension has the form;

$$\left(-\frac{\hbar^2}{2m}\frac{d^2}{dx^2}+V(x)\right)\Psi = E\Psi \tag{1}$$

We will be considering the case where

$$V(x) = \frac{1}{2}kx^2 \tag{2}$$

i.e the simple harmonic oscillator, where $k$ represents the spring constant. In this case analytic solutions to the Schrödinger equation exist which we can compare with the numerical solution.

For convenience we use the dimensionless variables

$$\hat{x} = \left(\frac{mk}{\hbar^2}\right)^{\frac{1}{4}}x \text{ and } \hat{E} = \frac{2}{\hbar}\sqrt{\frac{m}{k}}E \tag{3}$$

This gives us an equation of the form

$$\frac{d^2\Psi}{dx^2} = (\hat{V}(\hat{x}) - \hat{E})\Psi \tag{4}$$

which can be written as

$$\frac{d^2\Psi}{dx^2} = f(\hat{x})\Psi \text{ where } f(\hat{x}) = \hat{x}^2 - \hat{E} \tag{5}$$

For a physically acceptable solution to exist, i.e. one where $\int_{-\infty}^{+\infty}|\Psi|^2 dx = 1$ and $\Psi(x) \to 0$ as $\hat{x} \to \pm\infty$, the energy must have specific values which are the eigenvalues of the equation.

Analytic solutions to this equation are of the form

$$\Psi_n(\hat{x}) = H_n e^{-\frac{\hat{x}^2}{2}} \tag{6}$$

with each $n$ corresponding to a different eigenstate and $H_n$ being the Hermite polynomials, which satisfy the recurrence relation

$$H_{n+1} = 2\hat{x}H_n - 2H_{n-1} \quad H_1 = 2\hat{x},\ H_0 = 1 \tag{7}$$

The eigenvalues satisfy

$$\hat{E}_n = 2n + 1 \quad n \in \mathbb{Z}^+, 0 \tag{8}$$

We first integrate the Schrödinger equation in the form of equation (5) using the Numerov method and then compute the eigenvalues by using trial value suitably close to the eigenvalue and an interval bisection method to work down to the eigenvalue.

# 3 Solution

### Part 1 – Integrating the equation with the Numerov method

We use Numerov's algorithm which can be stated as, for a grid of points with spacing δ, the value of Ψ at the $(n+1)^{\text{th}}$ grid point can be approximately written with the recurrence relation

$$\left(1 - \frac{\delta^2}{12} f_{n+1}\right)\Psi_{n+1} \approx \left(2 + \frac{5\delta^2}{6} f_n\right)\Psi_n - \left(1 - \frac{\delta^2}{12} f_{n-1}\right)\Psi_{n-1} \tag{9}$$

where $f_n$ and $\Psi_n$ are the values of $f$ and Ψ at $\hat{x} = n\delta$. This only applies to equations in the form of (4). The derivation of this equation is included in the appendix.

We require two points to start the integration. Since the potential is symmetric solutions will fall into odd and even categories. Starting from $\hat{x} = 0$ we can use the boundary conditions

$$\begin{aligned} &\text{Even: } \Psi(0) = 1;\ \frac{d\Psi}{d\hat{x}} = 0 \\ &\text{Odd: } \Psi(0) = 0;\ \frac{d\Psi}{d\hat{x}} = 1 \end{aligned} \tag{10}$$

We can get a second point by using the Taylor expansion of $\Psi(\hat{x})$ to order four;

$$\Psi(\delta) = \Psi(0) + \Psi'(0)\delta + \frac{1}{2!}\Psi''(0)\delta^2 + \frac{1}{3!}\Psi'''(0)\delta^3 + \frac{1}{4!}\Psi''''(0)\delta^4 + \cdots \tag{11}$$

Since $\Psi'' = f\Psi$ we have, by differentiating

$$\begin{aligned} \Psi''' &= f'\Psi + f\Psi' \\ \Psi'''' &= f''\Psi + 2f'\Psi' + f\Psi'' \end{aligned} \tag{12}$$

Combining these with the boundary conditions and using the facts

$$f(\hat{x}) = \hat{x}^2 - \hat{E} \tag{13}$$

$$f'(\hat{x}) = 2\hat{x}$$
$$f''(\hat{x}) = 2$$

This leads to:

$$\text{Even: } \Psi(\delta) = \Psi(0) + \frac{\delta^2}{2} f(0)\Psi(0) + \frac{\delta^4}{24}\Big\{f''(0)\Psi(0) + 2f'(0)\Psi(0) + \big(f(0)\big)^2\Psi(0)\Big\} + \cdots$$
$$\text{Odd: } \Psi(\delta) = \delta\Psi'(0) + \frac{\delta^3}{6}\{f(0)\Psi'(0) + f'(0)\Psi(0)\} + \cdots \tag{14}$$

Substituting the values of these functions in we get approximations for $\Psi(\delta)$ for each case being

$$\text{Even: } \Psi(\delta) \approx 1 - \frac{\delta^2}{2}\hat{E} + \frac{\delta^4}{24}\{2 + \hat{E}^2\}$$
$$\text{Odd: } \Psi(\delta) \approx \delta - \frac{\delta^3}{6}\hat{E} \tag{15}$$

With these values we use the Numerov algorithm to integrate the equation from $\hat{x} = 0$ to some arbitrary value $\hat{x}_1$ for a given value of $\hat{E}, \delta$ and $n$ where $n$ determines whether the solution is even or odd.

### Part 2 – Determining the Eigenvalues

The dependence on the solution on the energy value was observed and it was noted that slightly below the correct eigenvalue the plot goes off to plus or minus infinity and slightly above the same correct eigenvalue the plot goes off to infinity once again but with the opposite sign.
To determine the eigenvalue $\hat{E}$ we use a form of interval bisection. We first input a trial value $\hat{E}_1$, reasonably close to the eigenvalue and use this to form two other numbers $\hat{E}_2 = \hat{E}_1 + 1$ and $\hat{E}_3 = \hat{E}_1 - 1$. We then use Numerov's method to find the values of $\Psi(5)$ with $\hat{E}_2$ and $\hat{E}_3$ as inputs for the energy. From the previous assumption we know that at least one of the pairs $\{\Psi(\hat{x}_1, \hat{E}_1)\Psi(\hat{x}_1, \hat{E}_2)\}$ and $\{\Psi(\hat{x}_1, \hat{E}_1)\Psi(\hat{x}_1, \hat{E}_3)\}$, has a value less than zero, provided that the true value for the energy lies in between. If the true eigenvalue doesn't lie within the interval between the two energies two more values are chosen, spaced further apart. Here $\Psi(\hat{x}_1, E)$ represents our numerical solution to the Schrodinger equation at the upper integration limit $\hat{x}_1$ for a given energy. We then take the mean of these two values and, checking which of the products of $\Psi$ with this new value and the values from which this energy is formed is less than zero. After determining which energy value results in a values $\Psi$ with opposite signs we repeat the process it until the difference between the two energies becomes very small, at which point the eigenvalue should be determined.

## 4 Results

### Part 1 – Integrating the equation with Numerov's method

The numerical results were taken and plotted against the analytic solution.
The numerical solution had to be scaled in order for it to be compared to the analytic solution since, while our numerical solution is a solution to the differential equation, it was generally a factor smaller in amplitude than the analytic solution. This is because the underlying differential equation we integrated was homogeneous, so a solution can be scaled by an arbitrary factor and still be a solution.
For large values of $\hat{x}$ our numerical solution differed significantly from the analytic solution, possibly due to accumulation of precision errors due to very large dividing term when calculating the $\Psi_{n+1}$ value and from the approximations made using throughout our solution. Using smaller values of $\delta$ suppressed the increasing $f_{n+1}$ term giving solutions which don't deviate from the analytic solution until much greater values of $\hat{x}$. For consistency results for the first four eigenstates are shown; a result with

eigenvalues slightly deviated from the proper eigenvalues is shown and, as an example, an unscaled result is shown, showing the deviation from the analytic results at large values of $\hat{x}$, and a corresponding unscaled results where a smaller value of $\delta$ is shown to show how it improves the accuracy of the numerical solution on the same interval.

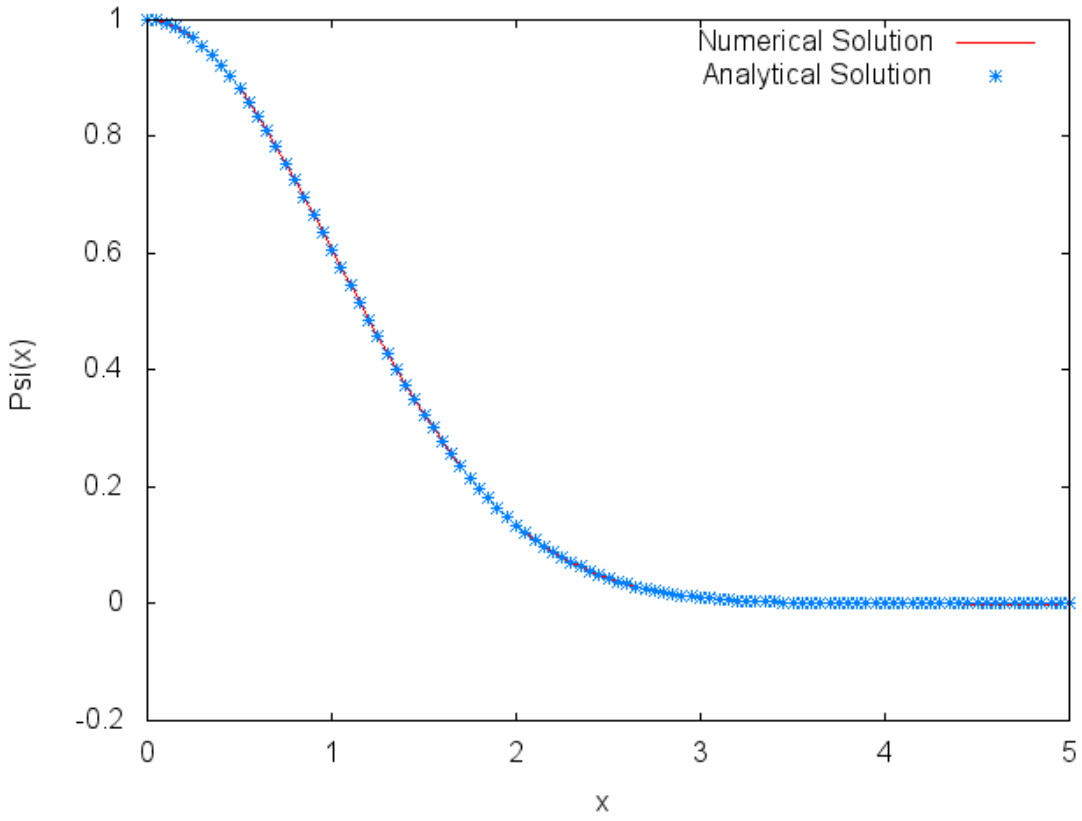


Figure 1: Plot of the numerical and analytic solution corresponding to $\hat{E} = 1$

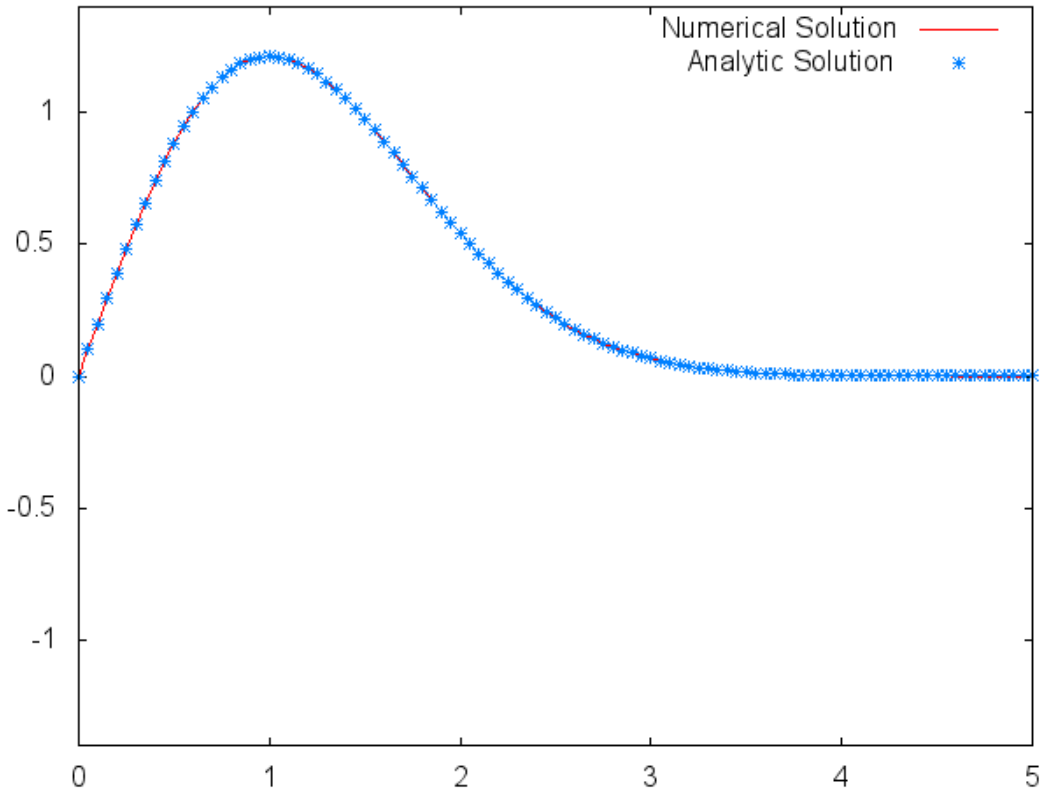


Figure 2: Plot of the numerical and analytic solution corresponding to $\hat{E} = 3$

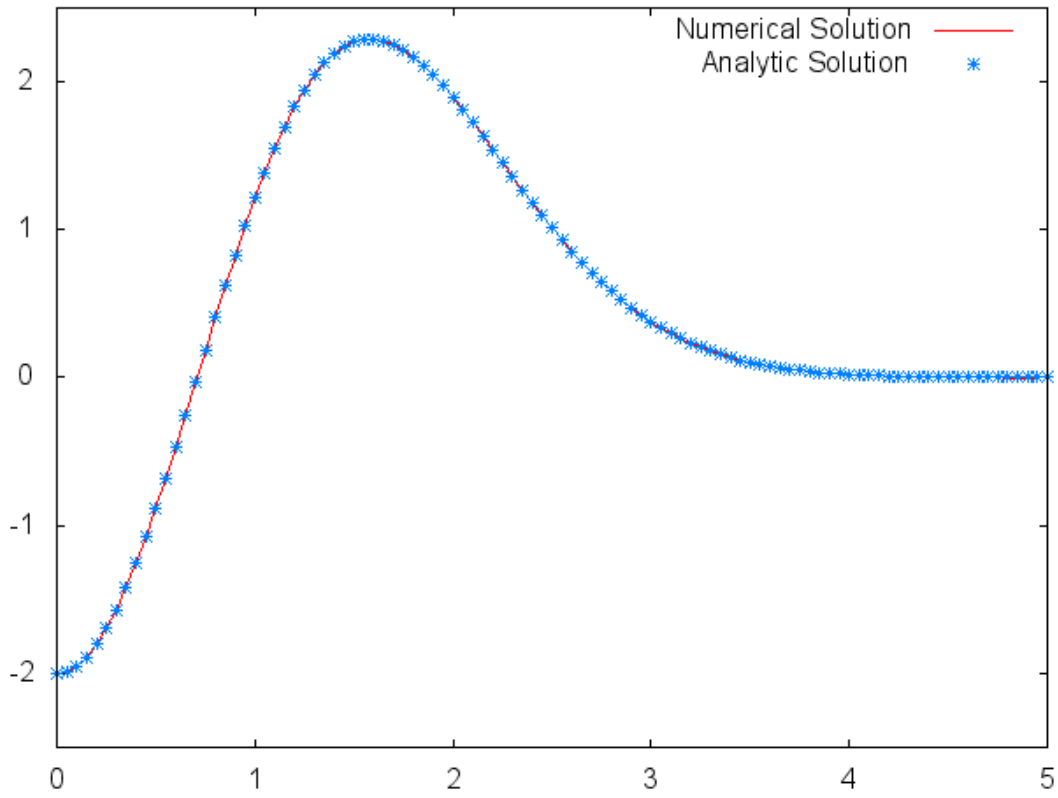


Figure 3: Plot of the numerical and analytic solution corresponding to $\hat{E} = 5$

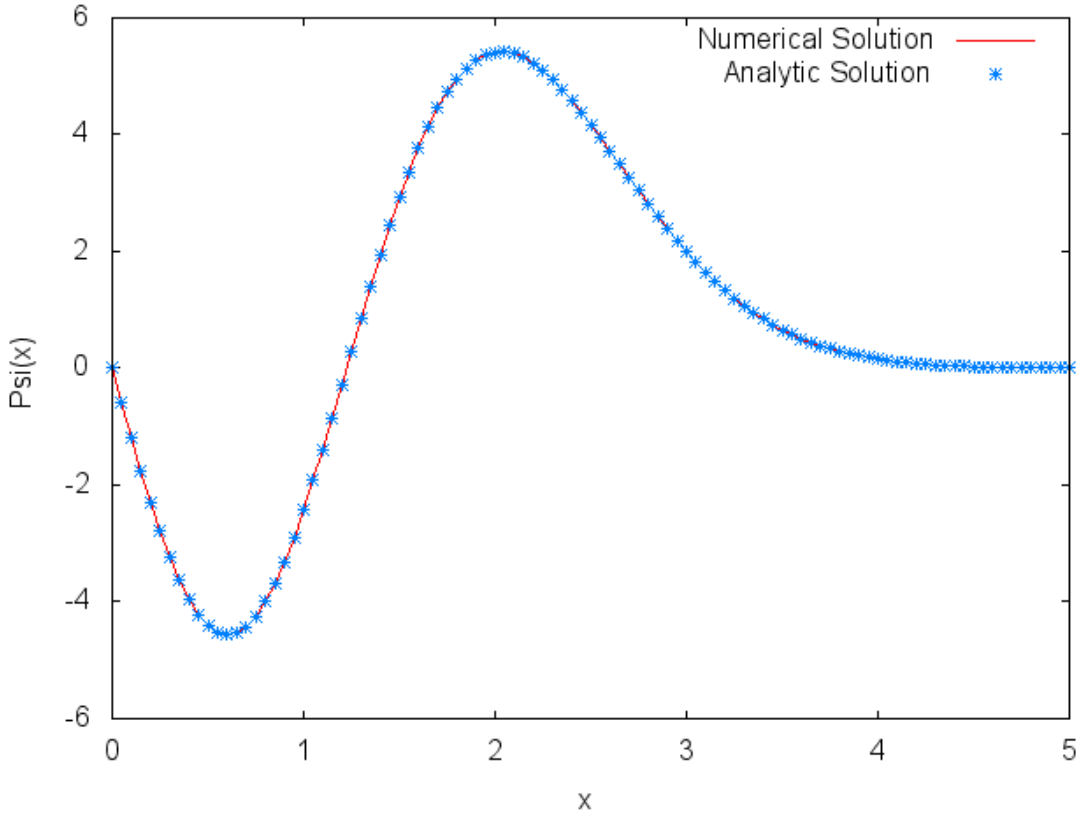


Figure 4: Plot of the numerical and analytic solution corresponding to $\hat{E} = 7$

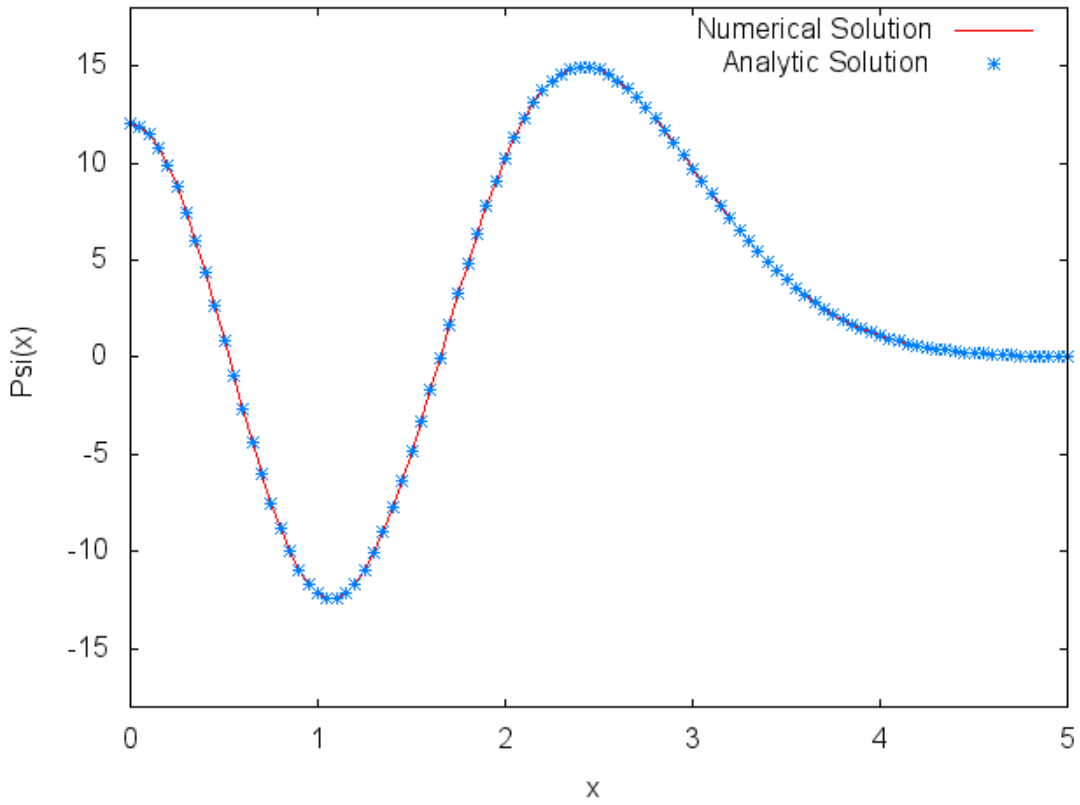


Figure 5: Plot of the numerical and analytic solution corresponding to $\hat{E} = 9$

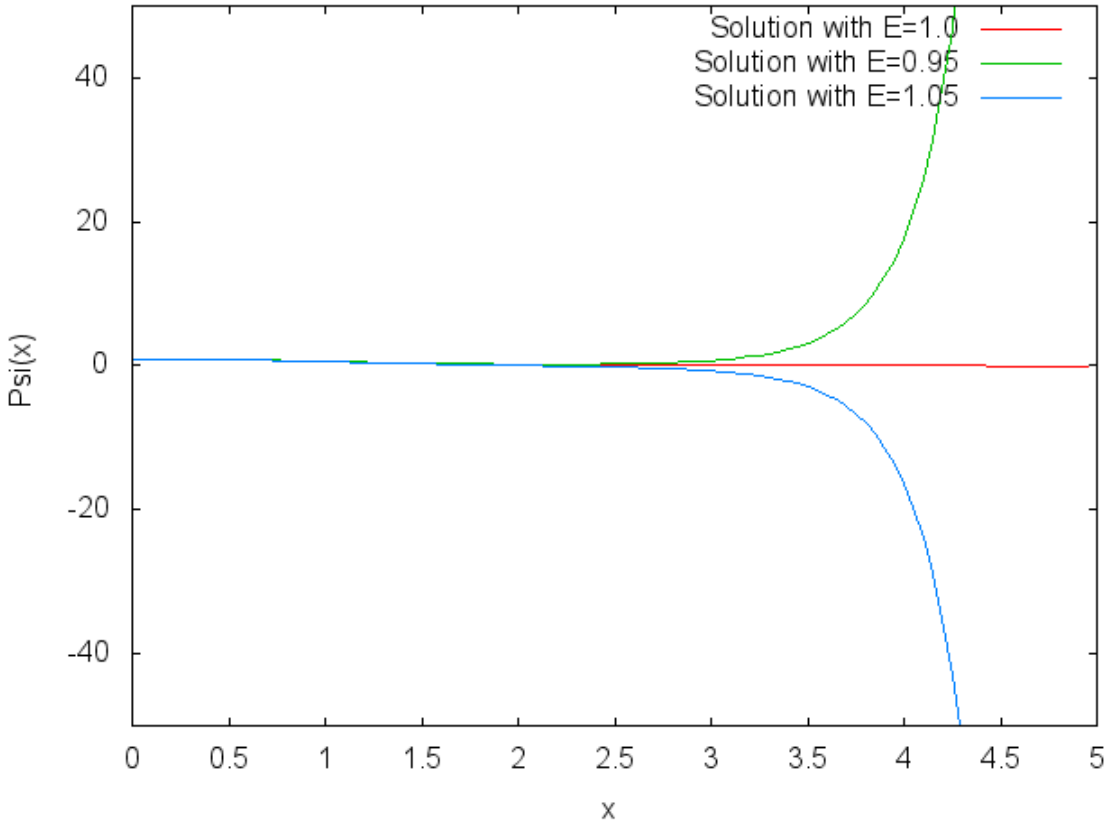


Figure 6: Plot of the numerical solutions corresponding to $\hat{E} = 0.95$ ,1.0 and 1.05

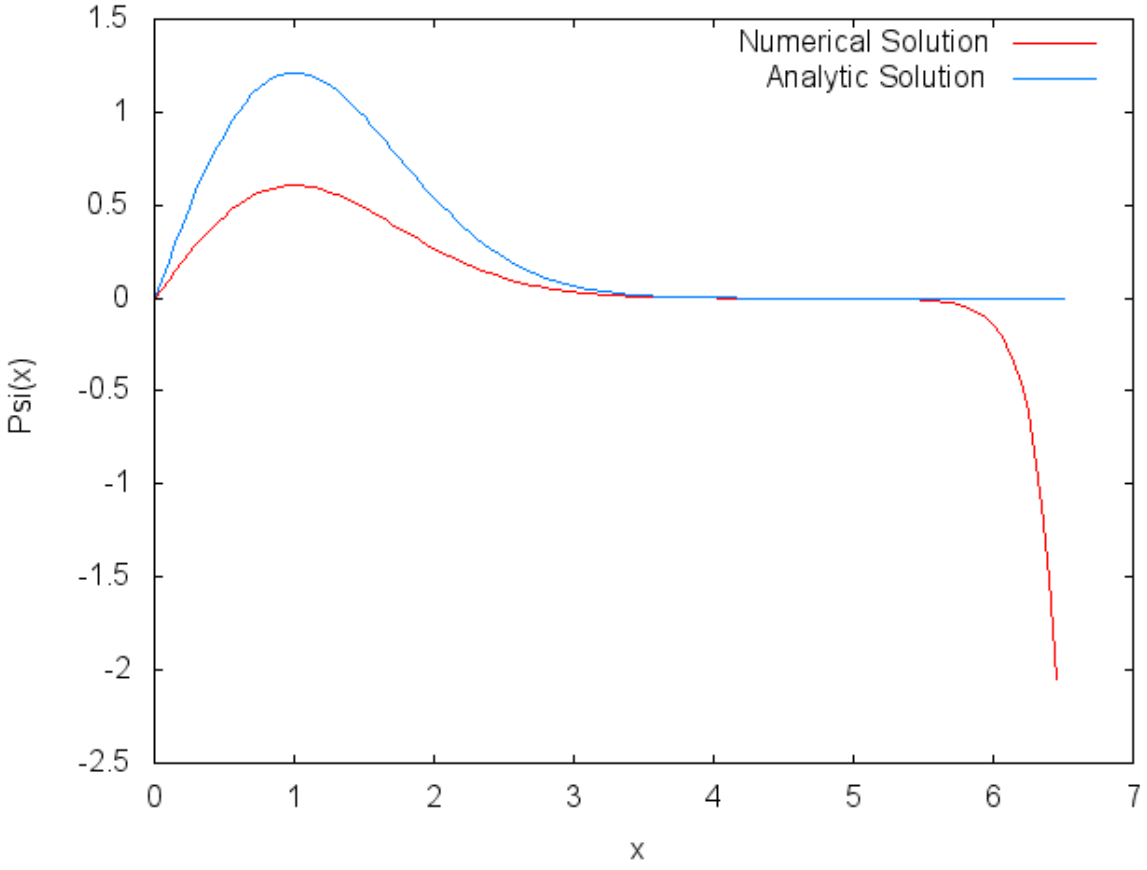


Figure 7: Plot of the unscaled numerical solution and analytic solution from 0 to $\hat{x}_1$=6.5 with $\delta = 0.05$ corresponding to $\hat{E} = 3$

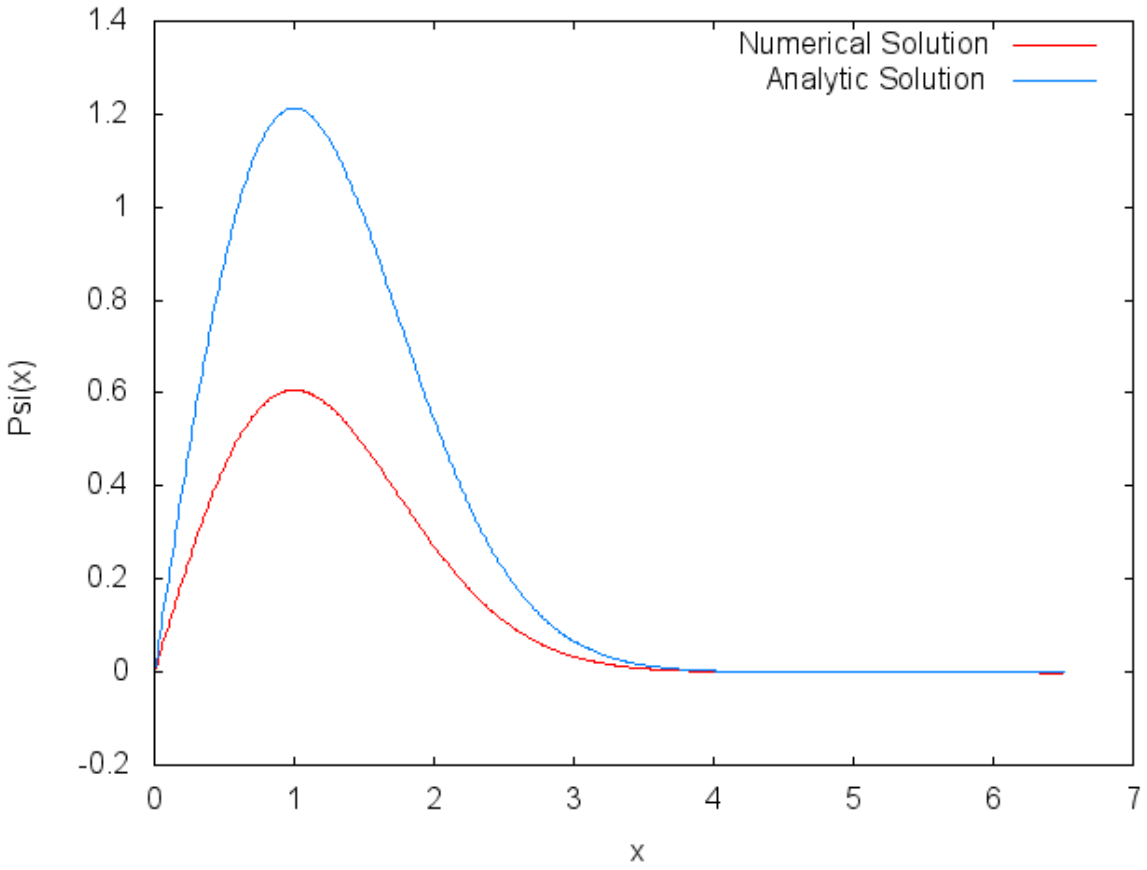


Figure 8: Plot of unscaled numerical solution against analytic solution from 0 to $\hat{x}_1$=6.5 with $\delta = 0.01$ corresponding to $\hat{E} = 3$

## Part 2 – Determining the Eigenvalues

We ran the devised program several times with the results depicted in the following tables. The range of trial values over which the program works is shown in the trial value column; within this range the program converged onto the output value displayed in the proceeding column. The first four tables show the results for smaller spacings and larger upper limits. The combination of smaller spacings and larger upper limits seems to improve the accuracy of the output value. The last table show the results when using a spacing of 0.01 and an upper limit of 6 for higher eigenstates, showing the reduced accuracy in output value.

| ***Eigenstate*** $(n)$ | ***Spacing*** $(\delta)$ | ***Upper limit*** $(\hat{x}_1)$ | ***Trial Value*** $(\hat{E}_1)$ | ***Output Value*** $(\hat{E}_f)$ | ***Actual Value*** $(\hat{E})$ |
|---|---|---|---|---|---|
| 0 | 0.05 | 5 | $-1.0 < \hat{E}_1 < 3.0$ | 1.000000 | 1 |
| 1 | 0.05 | 5 | $1.0 < \hat{E}_1 < 5.0$ | 3.000000 | 3 |
| 2 | 0.05 | 5 | $3.0 < \hat{E}_1 < 7.0$ | 4.999999 | 5 |
| 3 | 0.05 | 5 | $5.0 < \hat{E}_1 < 9.0$ | 7.000001 | 7 |
| 4 | 0.05 | 5 | $7.0 < \hat{E}_1 < 11.0$ | 9.000031 | 9 |

Figure 9: Table showing calculate eigenvalues with $\delta = 0.05$ and $\hat{x}_1 = 5$ for a range of trial values

| ***Eigenstate*** $(n)$ | ***Spacing*** $(\delta)$ | ***Upper limit*** $(\hat{x}_1)$ | ***Trial Value*** $(\hat{E}_1)$ | ***Output Value*** $(\hat{E}_f)$ | ***Actual Value*** $(\hat{E})$ |
|---|---|---|---|---|---|
| 0 | 0.01 | 5 | $-1.0 < \hat{E}_1 < 3.0$ | 1.000000 | 1 |
| 1 | 0.01 | 5 | $1.0 < \hat{E}_1 < 5.0$ | 3.000000 | 3 |
| 2 | 0.01 | 5 | $3.0 < \hat{E}_1 < 7.0$ | 5.000000 | 5 |
| 3 | 0.01 | 5 | $5.0 < \hat{E}_1 < 9.0$ | 7.000003 | 7 |
| 4 | 0.01 | 5 | $7.0 < \hat{E}_1 < 11.0$ | 9.000027 | 9 |

Figure 10: Table showing calculate eigenvalues with $\delta = 0.01$ and $\hat{x}_1 = 5$ for a range of trial values

| ***Eigenstate*** $(n)$ | ***Spacing*** $(\delta)$ | ***Upper limit*** $(\hat{x}_1)$ | ***Trial Value*** $(\hat{E}_1)$ | ***Output Value*** $(\hat{E}_f)$ | ***Actual Value*** $(\hat{E})$ |
|---|---|---|---|---|---|
| 0 | 0.05 | 6 | $-1.0 < \hat{E}_1 < 3.0$ | 1.000000 | 1 |
| 1 | 0.05 | 6 | $1.0 < \hat{E}_1 < 5.0$ | 3.000000 | 3 |
| 2 | 0.05 | 6 | $3.0 < \hat{E}_1 < 7.0$ | 4.999999 | 5 |
| 3 | 0.05 | 6 | $5.0 < \hat{E}_1 < 9.0$ | 6.999997 | 7 |
| 4 | 0.05 | 6 | $7.0 < \hat{E}_1 < 11.0$ | 8.999993 | 9 |

Figure 11: Table showing calculate eigenvalues with $\delta = 0.05$ and $\hat{x}_1 = 6$ for a range of trial values

| ***Eigenstate*** $(n)$ | ***Spacing*** $(\delta)$ | ***Upper limit*** $(\hat{x}_1)$ | ***Trial Value*** $(\hat{E}_1)$ | ***Output Value*** $(\hat{E}_f)$ | ***Actual Value*** $(\hat{E})$ |
|---|---|---|---|---|---|
| 0 | 0.01 | 6 | $-1.0 < \hat{E}_1 < 3.0$ | 1.000000 | 1 |
| 1 | 0.01 | 6 | $1.0 < \hat{E}_1 < 5.0$ | 3.000000 | 3 |
| 2 | 0.01 | 6 | $3.0 < \hat{E}_1 < 7.0$ | 5.000000 | 5 |
| 3 | 0.01 | 6 | $5.0 < \hat{E}_1 < 9.0$ | 7.000000 | 7 |
| 4 | 0.01 | 6 | $7.0 < \hat{E}_1 < 11.0$ | 9.000000 | 9 |

Figure 12: Table showing calculate eigenvalues with $\delta = 0.01$ and $\hat{x}_1 = 6$ for a range of trial values

| ***Eigenstate*** $(n)$ | ***Spacing*** $(\delta)$ | ***Upper limit*** $(\hat{x}_1)$ | ***Trial Value*** $(\hat{E}_1)$ | ***Output Value*** $(\hat{E}_f)$ | ***Actual Value*** $(\hat{E})$ |
|---|---|---|---|---|---|
| 5 | 0.01 | 6 | $9.0 < \hat{E}_1 < 13.0$ | 11.000000 | 11 |
| 6 | 0.01 | 6 | $11.0 < \hat{E}_1 < 15.0$ | 13.000000 | 13 |
| 7 | 0.01 | 6 | $13.0 < \hat{E}_1 < 17.0$ | 15.000003 | 15 |
| 8 | 0.01 | 6 | $15.0 < \hat{E}_1 < 19.0$ | 17.000019 | 17 |
| 9 | 0.01 | 6 | $17.0 < \hat{E}_1 < 21.0$ | 19.000108 | 19 |
| 10 | 0.01 | 6 | $19.0 < \hat{E}_1 < 23.0$ | 21.000535 | 21 |

Figure 13: Table showing calculate eigenvalues with $\delta = 0.01$ and $\hat{x}_1 = 6$ for a range of trial values

# 5 Conclusion

Using the two computer models we computed the solution to the time independent Schrodinger equation for the quantum harmonic oscillator and the eigenvalues of the solution for the first five eigenstates. We found them to correspond closely to the analytic solution to within an arbitrary scale factor, given suitable boundary conditions at the origin which were determined by whether the function was even or odd, but deviated from the analytic solution for large values of $\hat{x}$. We also found that we could compensate for this by using a smaller interval spacing ($\delta$), giving us even more accurate solutions.
We found that the dependence on the solution on the energy value was such that slightly above or below the correct eigenvalue the solution did not converge as $\hat{x} \to +\infty$ used this to construct an interval bisection method to compute an approximation for the correct eigenvalue. This approximation was found to be quite accurate but deviated slightly for higher eigenstates. Again we found that we could improve accuracy by using a smaller interval spacing and also a larger upper limit ($\hat{x}_1$).

The algorithm with which we integrated the equation has several advantages:

- Numerov's method is a higher order method than other such methods for solving differential equations such as the improved Euler method (order 2) and the Euler method (order 1), with a lower global error.
- The Numerov method has a local error of $\mathcal{O}(h^6)$ with only one calculation of $f(\hat{x})$ per step compared to the 4th order Runge-Kutta method which requires the evaluation of 6 functions in each step to obtain the same local accuracy. The Numerov method therefore provides a far better solution to the problem, being more accurate with a lower running time, than a fourth order Runge-Kutta approximation.
- Furthermore the Numerov method can be solved backwards or forwards with a local error of $\mathcal{O}(h^6)$ which is one order more accurate than the Runge-Kutta method which could be used to integrate the problem as two coupled first order O.D.E.s.

On the other hand the algorithm comes with a few disadvantages:

- The problem with the Numerov method, whether we integrate forwards or backwards, is that we need two starting values either $\Psi_n$ and $\Psi_{n-1}$ to forward integrate or $\Psi_n$ and $\Psi_{n+1}$ to backward integrate, where we usually only have one initial value $\Psi_0$. To calculate $\Psi_2$ with an accuracy of $\mathcal{O}(h^6)$ we need a value of $\Psi_1$ with at least that accuracy, however, since the global error of the method is $\mathcal{O}(h^5)$ and we calculate $\Psi_1$ only once we can use a method that is $\mathcal{O}(h^5)$ accurate to calculate $\Psi_1$ and still retain our global accuracy.
- In our case the Numerov method failed for large $\hat{x}$, causing solutions that deviated considerably from the analytic solution due to accumulation of precision errors.
- Finally, the method is quite inflexible in that it requires the differential equation to be in a specific form, and so can only be applied to a specific class of equations.

Nevertheless, in terms of our problem, the Numerov method was an efficient and effective algorithm for integrating the Schrodinger equation, producing highly accurate results within the interval we chose which closely followed the true values. Even problems with accuracy can be compensated for by a suitable choice of input parameters in return for longer computation.

# References

[1] *CO22 Practical Script,* Oxford Physics, 2009

[2] A. O'Hare, *Numerical Methods for Physicists,* Oxford Physics, 2005

# Appendix A Source Code for Part 1

The source code for the first part of the problem (where we numerically integrate the Schrödinger integration and plot it against the analytic solution) is:

```
/*****************************************************************************
*  CO22 Solutions of Schrödinger's Equation by Numerical Integration - Part 1
*  A program designed to integrate the time-independent Schrödinger using
*  Numerov's algorithm for a given value of E, the energy, x1, the limit of the
*  integration, n, which corresponds to the relevant eigenstate and delta, the
*  width spacing used in the program.
*****************************************************************************/

#include <math.h>
#include <stdio.h>
#include <stdlib.h>

/*Function used to determine the value of psi_d when psi is even*/
double even_psi(double d, double E)
{   return (1.0 - (((d*d)*(E))/2.0) + ((d*d*d*d)*(2.0 + (E*E)))/24.0);
}

/*Function used to determine the value of psi_d when psi is odd*/
double odd_psi(double d, double E)
{   return (d - ((E*d*d*d)/6.0));
}

/*Function used to evaluate f(x) at some point x=N*delta*/
double f(double N, double d, double E)
{ return (N*N*d*d -E);}

/*Function used to evaluate the nth Hermite polynomial using a recurrence loop*/
double H(int n, double x)
{    if(n==0)
     return 1;
     else if (n==1)
     return 2.0*x;
     else return (2.0*x*H(n-1,x) - 2.0*(n-1)*H(n-2,x));
     }

int main()
{   double psi0, Dpsi, psi_d, psi_i, psi_j, psi_a, psi, psi_1, psi_2;
    double x, x1, x0;
    double d, E;        /*Variables used in the main function*/
    int n, N;

    FILE *fout;

    printf("Enter a value for the delta, x1, n and E\n");
    scanf("%lf %lf %d %lf", &d, &x1, &n, &E);

    if ((fout=fopen("HP.txt","w"))==NULL)
    { printf("Cannot open %s\n", "HP.txt");
    exit(EXIT_FAILURE);}

 /*The values of the analytic solution are calculated for each x from 0
   to x1 with a width spacing of delta between*/
    for(x=0.0; x<=x1; x+=d)
    { psi_a=H(n, x)*exp(-(x*x)/2.0);
      fprintf(fout,"%f\t %f\n", x, psi_a);
        }
   if ((fout=fopen("Schrd.txt","w"))==NULL)
    { printf("Cannot open %s\n", "Schrd.txt");
```

```
    exit(EXIT_FAILURE);}

   /*The integer n is checked to see whether it is odd or even. Since n is an
     int variable, if it is odd, dividing it by two will return an int and so
     multiplying it then by two will return a number not equal to n. If n is
     even, after diving it by two and then multiplying it by two the answer will
     be n. Once n has been determined to be either even or odd the corresponding
     value of psi0 and Dpsi is assigned and psi_d is calculted*/
   if ( (n/2)*2 == n )
     {x0=0.0;
      psi0=1.0;
      Dpsi=0.0;
      psi_d=even_psi(d, E);
      fprintf(fout,"%f\t %f\n", x0, psi0);
      fprintf(fout,"%f\t %f\n", d, psi_d); }
   else
     {x0=0.0;
      psi0=0.0;
      Dpsi=1.0;
      psi_d=odd_psi(d, E);
      fprintf(fout,"%f\t %f\n", x0, psi0);
      fprintf(fout,"%f\t %f\n", d, psi_d);}

   psi_j=psi0;
   psi_i=psi_d;

  /*Each value of psi is determined using Numerov's algorithm from x=2*delta to
    x=x1 with each point being a multiple, N, of delta.*/
  for(N=2; N<=(x1/d); N+=1)
   {
    psi=((((24.0+10.0*d*d*f(N-1, d, E))/(12.0-d*d*f(N, d ,E)))*psi_i
      - ((12.0-d*d*f(N-2, d, E))/(12.0-d*d*f(N, d, E)))*psi_j));

   fprintf(fout,"%f\t %f\n", N*d, psi);

   psi_1=psi_i;        /*Extra variables psi_1 and psi_2 are used as a mediator
   psi_2=psi_j;          between the old values of psi_i and psi_j to make the
                         association clearer and avoid confusion*/
   psi_j=psi_1;
   psi_i=psi;
    }

   fclose(fout);

   exit(0);

}
```

## Appendix B Source Code for Part 2

The source code for the second part of the problem (where we try to compute the energy eigenvalue corresponding to a particular eigenstate by starting from an eigenvalue close to the actual value) is:

```
/*****************************************************************************
*  CO22 Solutions of Schrödinger's Equation by Numerical Integration - Part 2
*  A program designed to find the energy eigenvalues of an arbitrary eigenstate
*  of the time-independent Schrödinger equation given an input value reasonably
*  close to the actual value.
*****************************************************************************/

#include <math.h>
#include <stdio.h>
#include <stdlib.h>
```

```
/*Function used to determine the value of psi_d when psi is even*/
double even_psi(double d, double E)
{   return (1- (((d*d)*(E))/2) + ((d*d*d*d)*(2 + (E*E)))/24);
}

/*Function used to determine the value of psi_d when psi is odd*/
double odd_psi(double d, double E)
{   return (d - (E*d*d*d)/6);
}

/*Function used to evaluate f(x) at some point x=N*delta*/
double f(double N, double d, double E)
{ return (N*N*d*d -E);}

/*Function used to evaluate the value of psi at the upper limit of integration*/
double psi(double d, double x1, int n, double E)
{   double psi0;
    double Dpsi;
    double psi_d;
    double x, x0;
    double psi_i;
    double psi_j;
    double psi;
    double psi_a;
    double psi_1, psi_2;
    double N;

/*Determines whether n is odd or even and returns psi_d*/
   if ( (n/2)*2 == n )
     {x0=0.0;
      psi0=1.0;
      Dpsi=0.0;
      psi_d=even_psi(d, E);
      }
   else
     {x0=0.0;
      psi0=0.0;
      Dpsi=1.0;
      psi_d=odd_psi(d, E);
     }

   psi_i=psi0;
   psi_j=psi_d;

/*Runs Numerov's algorithm to numerically integrate psi up to the upper limit*/
  for(N=1.0; N<=(x1/d); N+=1.0)
   {
    psi=((24+10*d*d*f(N-1, d, E))/(12-d*d*f(N, d ,E)))*psi_i
      - ((12-d*d*f(N-2, d, E))/(12-d*d*f(N, d, E)))*psi_j;

   psi_1=psi_i;
   psi_2=psi_j;

   psi_j=psi_1;
   psi_i=psi;
    }

   return psi;}

int main()
```

```
{
    double E1, E2, E3, k, psi1, psi2, psi3, d, x1;
    int n;

/*Relevant values, delta, x1, n, E1 and k, needed for the program are input*/
    printf("Enter a value for delta, x1, n, E1\n");
    scanf("%lf %lf %d %lf", &d, &x1, &n, &E1);

/*Two trial values for the energy are assigned*/
    E2=E1+1.0;
    E3=E1-1.0;

/*The values of psi at the upper limit for each energy value is computed*/
    psi1=psi(d, x1, n, E1);
    psi2=psi(d, x1, n, E2);
    psi3=psi(d, x1, n, E3);

/*If the first two trial values generate values of psi with the same sign then
  two more trial values are computed, slightly further apart*/
    if ((psi1*psi2<0 && psi1*psi3<0) || (psi1*psi2>0 && psi1*psi3>0))
    {  E2=E2+1.0;
       E3=E3-1.0;
       psi1=psi(d, x1, n, E1);
       psi2=psi(d, x1, n, E2);
       psi3=psi(d, x1, n, E3);
}

/*The sign of the product of the value of Psi with each eigenvalue at the
  upper limit is determined and if it is negative then the corresponding
  eigenvalue is chosen and the average is formed*/
    if (psi1*psi2<0)
    {E2=E2;}
    else if (psi1*psi3<0)
    {E2=E3;}

    E3=(E1+E2)/2;

/*The process of interval bisection is repeated until the sepeartion between
  the two values is smaller than 0.00000001 and the final value for the
  computed eigenvalue is printed*/
    while(fabs(E2-E1)>0.00000001)
    { psi1=psi(d, x1, n, E1);
      psi2=psi(d, x1, n, E2);
      psi3=psi(d, x1, n, E3);

      if(psi1*psi3<0)
      {E2=E3;}
      else if(psi2*psi3<0)
      {E1=E2;
       E2=E3;}

      E3=(E1+E2)/2;}

    printf("The eigenvalue is approximately %f\n", E3);

    exit(0);

}
```

## Appendix C Derivation of the Numerov Equation

The Numerov method is a method used for solving equations of the form

$$\frac{d^2y}{dx^2} + f(x)y = g(x) \tag{C.1}$$

By taking the Taylor series to the fourth power of $\frac{d(y+h)}{dx} = f(y+h)$ and $\frac{d(y-h)}{dx} = f(y-h)$

$$\begin{aligned} f(y+h) &\approx f(y) + hf'(y) + \frac{h^2}{2}f''(y) + \frac{h^3}{6}f'''(y) + \frac{h^4}{24}f''''(y) \\ f(y-h) &\approx f(y) - hf'(y) + \frac{h^2}{2}f''(y) - \frac{h^3}{6}f'''(y) + \frac{h^4}{24}f''''(y) \end{aligned} \tag{C.2}$$

And adding them we get

$$f(y+h) + f(y-h) - 2f(y) = h^2f''(y) + \frac{h^4}{12}f''''(y) \tag{C.3}$$

Rearranging this we get

$$\frac{y_{n+1} - 2y_n + y_{n-1}}{h^2} = y_n'' + \frac{h^2}{12}y_n'''' \tag{C.4}$$

Where $f(y+h) = y_{n+1}$, $f(y-h) = y_{n-1}$ and $f(y) = y_n$
We can differentiate equation C.1 twice to get an expression for $y_n''''$

$$y_n'''' = \frac{d^2}{dx^2}(g(x) - f(x)y) \tag{C.5}$$

Using this and the centred difference approximation for the second derivative

$$f_n'' = \frac{f_{n+1} - 2f_n + f_{n+1}}{h^2} \tag{C.6}$$

We get

$$y_n'''' = \frac{g(x)_{n+1} - 2g(x)_n + g(x)_{n-1}}{h^2} - \frac{(f(x)y)_{n+1} - 2(f(x)y) + (f(x)y)_{n-1}}{h^2} \tag{C.7}$$

Which we can substitute into C.4 to get the Numerov equation

$$\left(1 + \frac{h^2}{12}f_{n+1}\right)y_{n+1} = \left(2 + \frac{5h^2}{6}f_n\right)y_n - \left(1 + \frac{h^2}{12}f_{n-1}\right)y_{n-1} - \frac{h^2}{12}(g_{n-1} - 10g_n + g_{n-1}) \tag{C.8}$$

If $g(x) = 0$ this reduces to

$$\left(1 + \frac{h^2}{12}f_{n+1}\right)y_{n+1} = \left(2 + \frac{5h^2}{6}f_n\right)y_n - \left(1 + \frac{h^2}{12}f_{n-1}\right)y_{n-1} \tag{C.9}$$