# Simple solutions of Schrödinger's equation by numerical integration

WX Zhang

Hilary 2025

## 1 Abstract

The report investigates solutions to the time-indepndent Schrödinger's equation in the context of a standard quantum harmonic oscillator potential. Specifically, we compared analytic solutions in the form of Hermite polynomials with numerical ones using Numerov's integration algorithm, as well as additionally introducing a computational method for finding eigenvalues for the harmonic oscillator eigenvalue equation. The two solution initiatives, though differing in advantages, produced very similar results for small intervals.

## 2 Introduction

The time-independent Schrödinger's equation (TISE) is one of the defining partial differential equations in quantum mechanics, yet solving it is not always straightforward. The quantum harmonic oscillator is one such rare example where exact analytic solutions can be found with the methods of standard PDEs via orthogonal polynomials (i.e. Hermite polynomials). However, one can still apply Numerov's method as a computationally effective numerical algorithm in this case.

Structurally, to mainly compare and contrast the two methods, the report will start with theoretical discussions regarding how analytic and numerical solutions work in principles. Then some detail of plots will be discussed, which leads to a necessary involvement of an "eigenvalue-finding" algorithm. After analyzing several plots of specific "bound" integer eigenvalues, we end up with a final evaluation of Numerov's method in the context of solving quantum harmonic oscillation as well as solving PDEs in general.

## 3 Main Body

### 3.1 Analytic Solution - Hermite Polynomials

We first set up the problem clearly. Starting with TISE in it general form:

$$\left(-\frac{\hbar^2}{2m}\nabla^2 + V(\mathbf{r})\right)|\psi\rangle = E\,|\psi\rangle \tag{1}$$

Breaking it down into the 1D case at $x$ position basis, while the potential is $V = V(x) = \frac{1}{2}kx^2$:

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

To make computation simplified afterwards, we change variables conveniently, with $\hat{x} = (\frac{mk}{\hbar^2})^{1/4}x$ and $\hat{E} = \frac{2E}{\hbar}\sqrt{\frac{m}{k}}$, thus:

$$\frac{d^2\psi}{d\hat{x}^2} = \left(\hat{x}^2 - \hat{E}\right)\psi \tag{2}$$

which is now a non-linear ODE in $\hat{x}$. Generally, it is not guaranteed that one can have "bound solutions" where $\psi \to 0$ for $\hat{x} \to \pm\infty$. This is, however, possible, for certain integer eigenvalues of $\hat{E}$ for which (2) tends to the standard Hermite differential equation with another change of variables $\psi(\hat{x}) = \phi(\hat{x})\exp\left(-\hat{x}^2/2\right)$, and bound solutions exist. The general form of an eigenfunction solution to (2) is:

$$\psi_n(x) = H_n(\hat{x})e^{-\hat{x}^2/2} \tag{3}$$

where $H_n(x)$ are Hermite polynomials (still require normalisation) defined by:
(Rodrigues' formula)

$$H_n(x) = (-1)^n e^{x^2}\frac{d^n}{dx^n}\left(e^{-x^2}\right)$$

(Recurrence relation)

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x)$$

It can be shown that both are equivalent, and note that here $\hat{E}$ must only take specific eigenvalues $\hat{E_n}$ for the solution form (3) to be valid, namely:

$$\hat{E}_n = 2n + 1 \; (n \in N) \tag{4}$$

The general solution will thus be a superposition of various eigenstates, in the form of:

$$\psi(x,t) = \sum_{n=0}^{\infty} c_n\psi_n(x)e^{-iE_nt/\hbar} \tag{5}$$

where $c_n$ are coefficients determined by boundary conditions. Note that these are now in terms of $E_n$ and $x$, which need to be transformed back with just a factor of constant.

Thus concludes the analytic method of PDEs. However, it is also shown that generally one can use Ladder Operators to arrive at the conclusion in (4) without manipulations of orthogonal polynomials, and reproducing wavefunctions in the end if one prefers.

## 3.2 Numerical Solution - Numerov's Method

Numerov's method is based on recurrence relations as well as initial conditions of the wavefunction $\psi(0)$ and $\psi'(0)$.

We consider the general form:

$$\frac{d^2\psi_n}{d\hat{x}^2} = f(\hat{x})\psi_n \tag{6}$$

where clearly in the case of quantum harmonic oscillators, $f(\hat{x}) = \hat{x}^2 - \hat{E}_n$.

Considering a set of discrete points on the $\hat{x}$ axis with spacing $\delta$. According to Numerov's integration algorithm, the value of the $j$-th point will be:

$$\left(1-\frac{\delta^2}{12}f_{j+1}\right)\psi_{j+1}=\left(2+\frac{5}{6}\delta^2 f_j\right)\psi_j-\left(1-\frac{\delta^2}{12}f_{j-1}\right)\psi_{j-1} \tag{7}$$

where we specify $f_j = f(j\delta)$ and $\psi_j = \psi(j\delta)$, and $j$ start numbering from 1.

Consider solutions of even and odd cases (depend on $n$) separately. Starting from the origin:
Even Case

$$\psi(0)=1, \quad \frac{d\psi}{d\hat{x}}(0)=0$$

Odd Case

$$\psi(0)=0, \quad \frac{d\psi}{d\hat{x}}(0)=1$$

Whilst the initial condition $\psi(0)$ fixes the case $j = 0$, but we also need to find the case $j = 1$ to apply the form (7). To do that, we need the Taylor expansion about $\hat{x} = 0$:

$$\psi(d)=\psi(0)+\psi'(0)d+\frac{1}{2}\psi''(0)d^2+\frac{1}{6}\psi^{(3)}(0)d^3+\frac{1}{24}\psi^{(4)}(0)d^4+\cdots$$

This is not as useful, since the higher orders of derivatives are not known. However, they can be simplified to be only expressed in terms of known $\psi(0)$, $\psi'(0)$ and $f$.

At first glance, it is obvious by the relation (6) that the second-order derivative of any derivative of $\psi$ is simply the product of $f$ with $\psi$, meaning that if one vanishes, then all such terms separated by orders of 2 vanish. In simple terms, at $\hat{x} = 0$, for even cases: $\psi'(0) = 0$, all odd derivatives will vanish, so only even derivatives survive; for odd cases: $\psi(0) = 0$, all even derivatives vanish, and only odd derivatives survive.

Moreover, we re-express the quadratic term for even cases:

$$\psi''(0)=f(0)\psi(0)$$

Cubic term for odd cases:

$$\begin{aligned}\psi^{(3)}(0)&=\frac{d\psi''}{d\hat{x}}|_{x=0}=\frac{d}{d\hat{x}}(f(\hat{x})\psi(\hat{x}))|_{x=0}\\&=f(0)\psi'(0)+f'(0)\psi(0)\end{aligned}$$

Quartic term for even cases: (where we apply Leibnitz' rule of differentiation)

$$\begin{aligned}\psi^{(4)}(0)&=\frac{d^2\psi''}{d\hat{x}^2}|_{x=0}=\frac{d^2}{d\hat{x}^2}\left(f(\hat{x})\psi(\hat{x})\right)|_{x=0}\\&=f''(0)\psi(0)+2f'(0)\psi'(0)+f(0)\psi''(0)\\&=f''(0)\psi(0)+2f'(0)\psi'(0)+f^2(0)\psi(0)\end{aligned}$$

In the end, summarizing the Taylor expansion:

$$\psi_{even}(\delta)=\psi(0)+\frac{\delta^2}{2}f(0)\psi(0)+\frac{\delta^4}{24}\left(f''(0)\psi(0)+2f'(0)\psi'(0)+f^2(0)\psi(0)\right) \tag{8}$$

$$\psi_{odd}(\delta)=\delta\psi'(0)+\frac{\delta^3}{6}(f(0)\psi'(0)+f'(0)\psi(0)) \tag{9}$$

This form will be useful when encoding the function in Matlab in the next section.

### 3.3 Computation

We start with an upper bound of $\hat{x}_1 = 6$ with spacing $\delta = 0.05$, plotting both the analytic solution and the numerical solution for $n = 0$ ($\hat{E} = 1$) case:

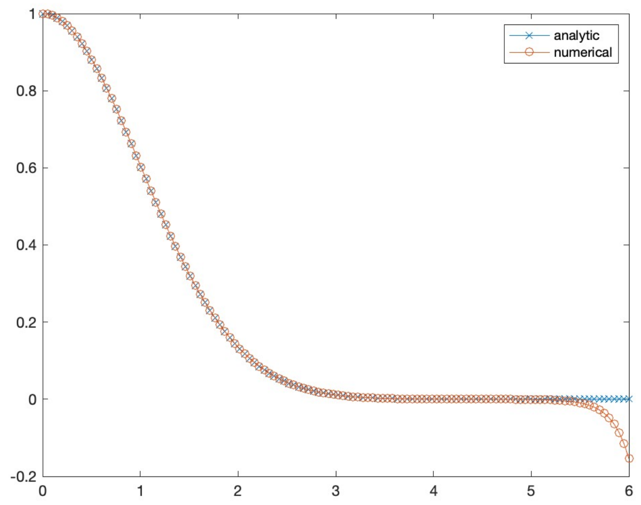

*Fig.1 Ground state* $\hat{E} = 1$

Then also consider the effects of varying spacing $\delta$:

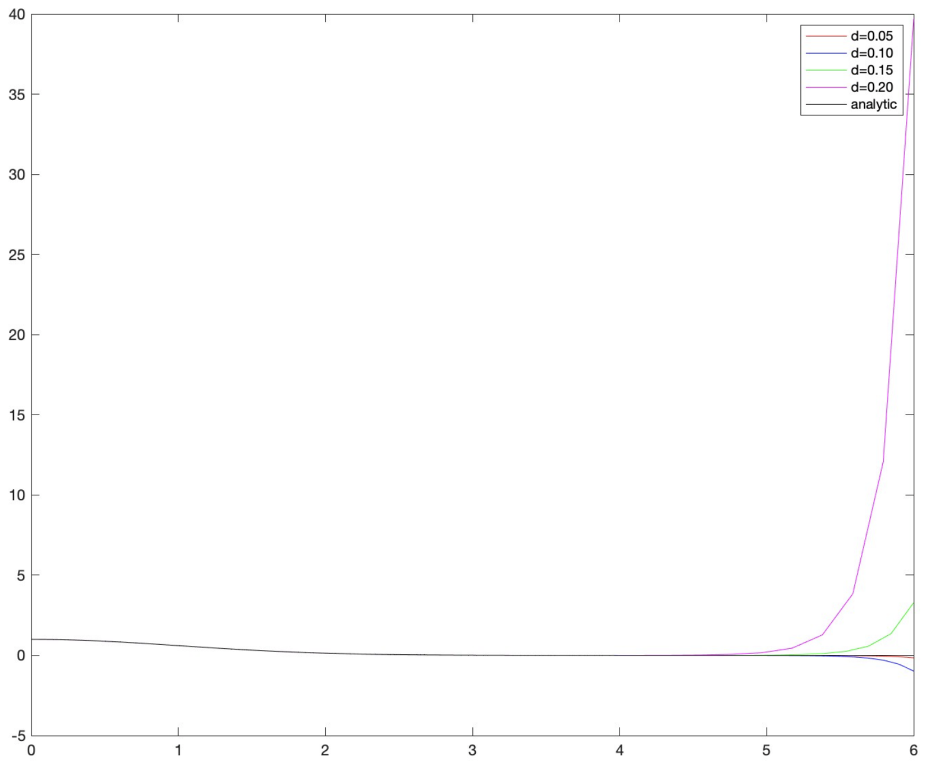

*Fig.2 Ground state with varying* $\delta$

Clearly, an increased value of spacing will lead to more deviations at larger values of $\hat{x} > 5$, as partly the truncation errors of ignoring higher order terms will be significant. However, choosing small $\delta$ also increases pressure on the computational power load.

Now consider "small" changes of $\hat{E}$ for $\hat{E} = 0.95, 1.00, 1.05$ respectively. One notes that the deviation is not insignificant for large values:

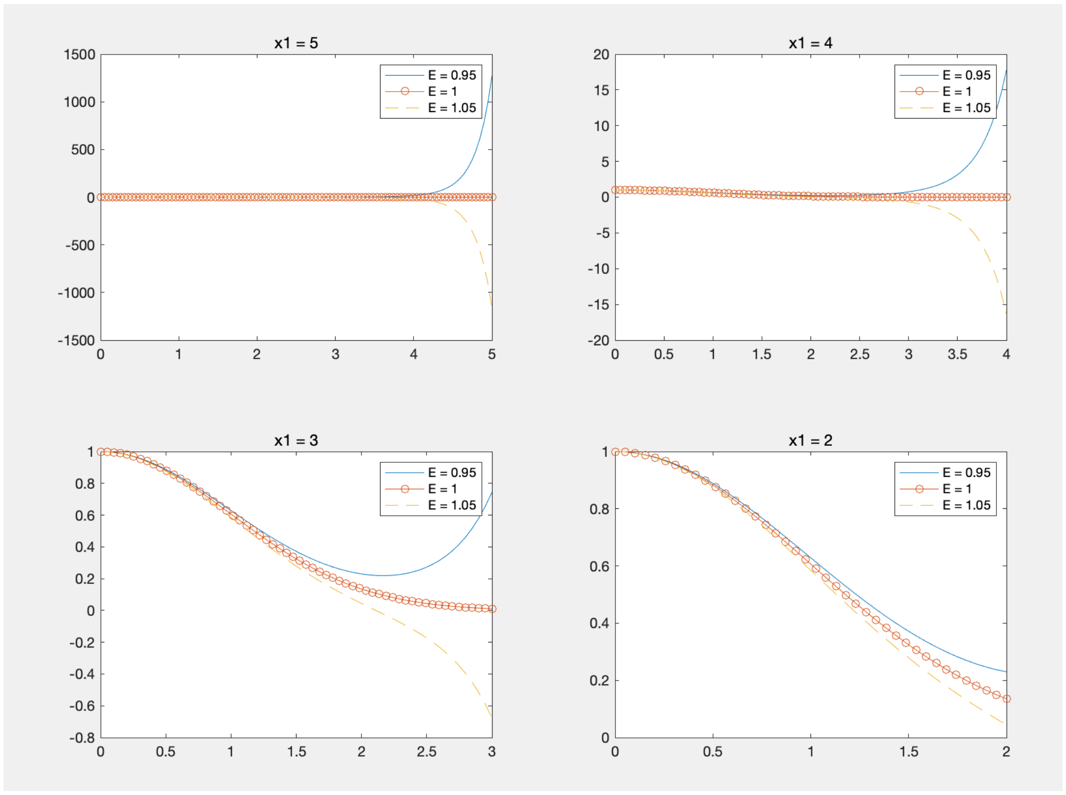

*Fig.3 Deviations for non-integer values of* $\hat{E}$

It might appears that the $\hat{E} = 1, \hat{x}_1 = 5$ case is just constant (get a little bit better for smaller $\hat{x}$), which is plainly misleading because its values are so negligible compared to the great perturbations for the other two non-integer cases. The numerical algorithm breaks down even when dealing with such seemingly "small" changes of $\hat{E}$.

## 3.4 Eigenvalue Finding

To fix the problem, we try to evaluate only at integer values exactly. The algorithm for finding such closest eigenvalues is included in the appendices, but nevertheless the method used is non-trivial.

The initiative involves for-loops, which distinguish between even and odd cases of $n$, and in each cycle one computes the value of wavefunction $\psi(\hat{x}_1)$ through Numerov's method using the value of $\hat{E}$ (set arbitrarily at the start, close to one integer eigenvalue). The algorithm then makes a change for the value of $\hat{E}$ in the direction where the value of $\psi(\hat{x}_1)$ decreases. This continues until $\hat{E}$ is very close to the true integer eigenvalue, and then we can round it to get the exact desired eigenvalue.

### 3.5 Several Plots

With the technique, to find eigenvalues correctly, we may compute a few values of $n$ to see the general patterns of solutions, in *Fig.4*:

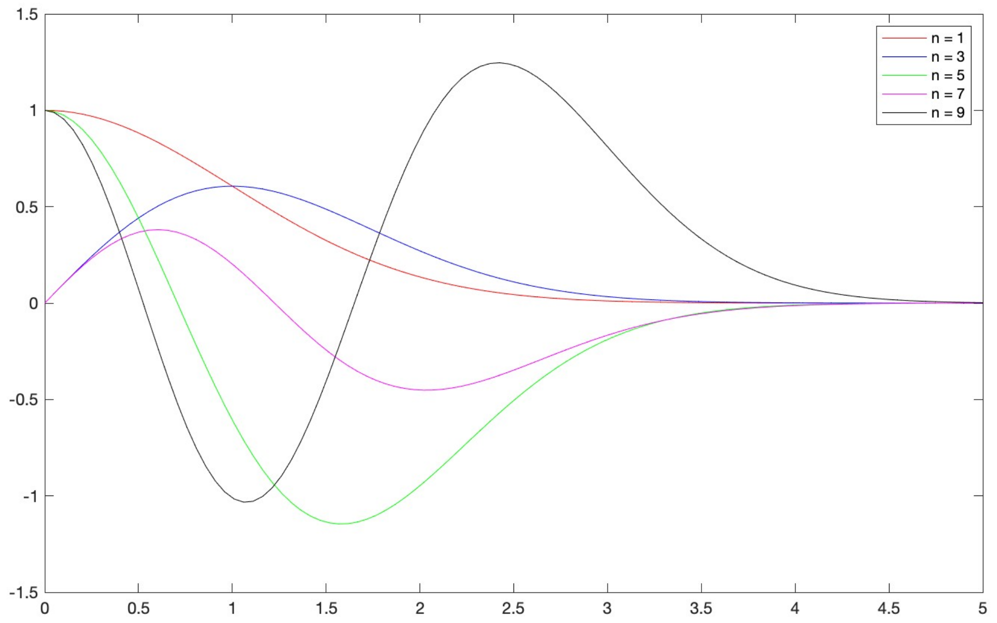

*Fig.4 Numerical solutions for various n*

If one has sufficient mathematical aptitude, this is recognisable as a product of Hermite polynomials with an exponentially decaying term, albeit "normalised" with different scalings. To see that comparison with the analytical solutions, considering the plots in *Fig.5*.

## 4 Conclusion

### 4.1 Results

Generally, with correct scaling in overall constants, the numerical solutions fit uniformly compared with the analytic solutions. There are some minor deviations at the large values but expected for truncation errors. Iterative calculations also imply that these errors will snowball and become uncontrolled eventually approaching a certain limiting value.

One more interesting result of observation is that for higher orders of $n$ and the same interval (limiting $\hat{x}_1$ kept constant), Numerov's numerical solutions seem to approach analytic solutions better than lower orders. This is possibly due to the fact that the absolute errors are much smaller because the expected value of $\psi$ is closer to 0, so the accumulation of errors is slower. i.e. This is in particular contrast to the ground state case where it is always positive and decreases rather slowly in a manner of "over-damped" motion instead of oscillating around 0, as exhibited by higher-order solutions which tend to resemble pure simple harmonic motion.

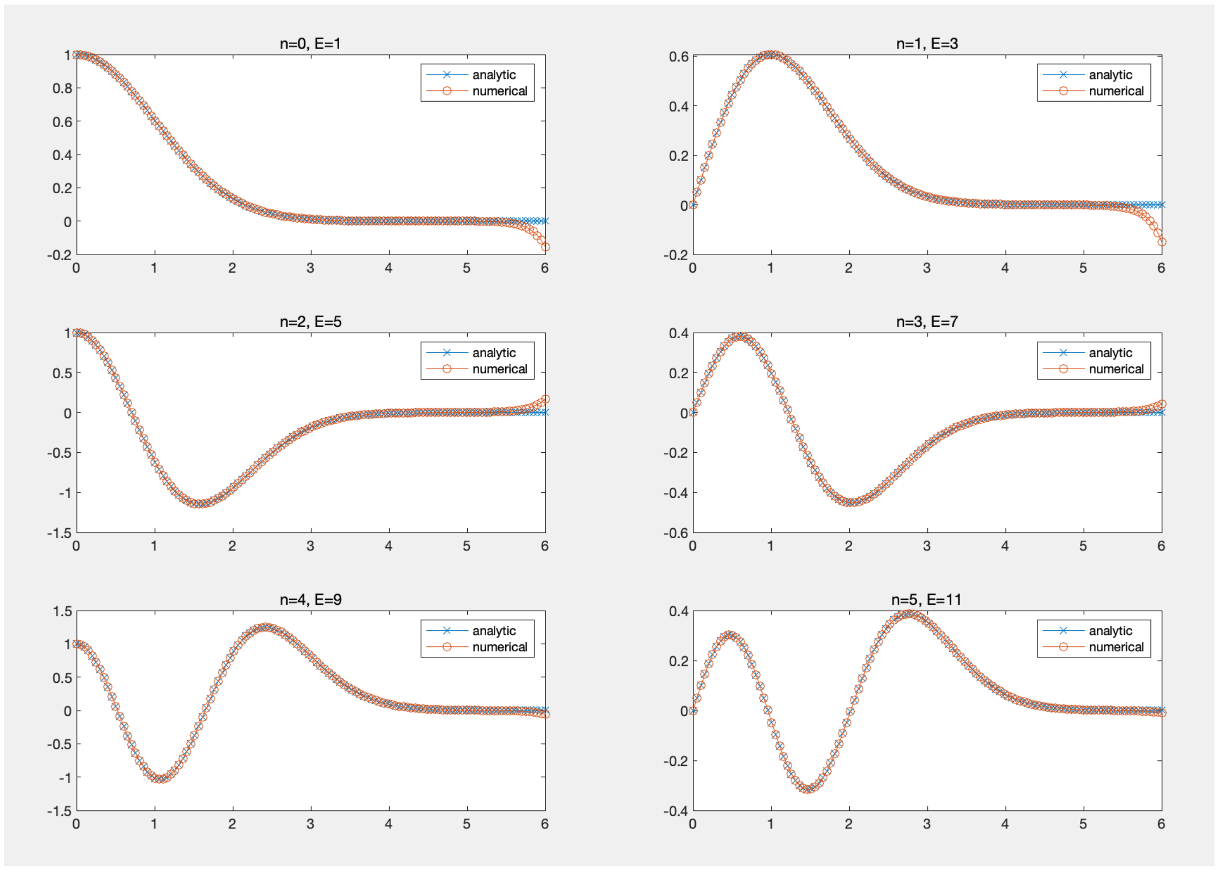

*Fig.5 Comparisons between numerical and analytic for different n*

## 4.2 Evaluation

To give a general evaluation of Numerov's integration algorithm, we consider the following aspects:

First, stability. When appropriately implemented with a small enough step size, the method is stable, particularly in cases like the quantum harmonic oscillator (exact eigenvalue case), where solutions are well-behaved and bounded.

Second, simplicity. The algorithm is based on simple recurrence relationship which is easy to implement and also computationally efficient, since the steps involved are not overcomplicated.

Third, deviations when eigenvalues are non-integers. This is an important issue, since the differential equation becomes unbounded, and Numerov's method will break down and result in exponentially exploding answers. This is why the introduction of an eigenvalue-finding algorithm is needed additionally.

Fourth, the method apparently fails for large values of $\hat{x}$ because truncation errors in Taylor expansions and errors with finite spacing become large, while the accumulated errors will eventually lead to exponential growth in deviations, which are no longer reliable.

## 5 Appendices

### 5.1 Functions

One for Numerov's algorithm, the other for eigenvalue finding.

```
function [psi] = solve_numerov(f, x, psi0, dpsi0)

% Author: Wanxiang Zhang (William) , Date: Jan 2025
% Solving d^2(psi)/dx^2 = f(x) psi from x0 to x1 with boundary condition
% psi(x0) = psi0 and d(psi(x0))/dx = dpsi0
% Input:
% * f: The function to be called on the right hand side of the equation (see hint
% section), receives x and returns the value of f(x).
% * x: array of the integration, the first element is the lower bound, x0, and the last
% element is the upper bound, x1.
% * psi0: the value of \psi at x0, i.e. \psi(x0)
% * dpsi0: the value of \psi derivative at x0, i.e. d\psi(x0)/dx
%
% Output:
% * psi: array of values of \psi with each element in the array corresponding to the same
% element in x

    psi_mat = linspace(0,0,size(x,2)); % Initialise a matrix for storage
    delta = (max(x)-min(x))/(size(x,2)-1); % Delta found
    f1 = diff(f,1); % First derivative of f
    f2 = diff(f,2); % Second derivative of f

    psi_mat(1,1) = psi0; %psi_0
    if psi0 == 1 & dpsi0 == 0; % even
        psi_mat(1,2) = psi0 + delta^2/2*f(0)*psi0 + delta^4/24*(f2(0)*psi0+2*f1(0)*dpsi0+f(0)^
    elseif psi0 == 0 & dpsi0 == 1;%odd
        psi_mat(1,2) = delta*dpsi0 + delta^3/6*(f(0)*dpsi0+f1(0)*psi0);
    end

    for j = 1:(size(x,2)-2) % update matrix entries
        num = (2+5/6*delta^2*f(j*delta))*psi_mat(1,j+1)-(1-delta^2/12*f((j-1)*delta))*psi_mat(
        den = (1-delta^2/12*f((j+1)*delta));
        psi_val = num/den;
        psi_mat(1,j+2) = psi_val;
    end

    psi = psi_mat;
end

function E = find_oscillator_eigenvalue(E0)
% Author: Wanxiang Zhang (William) , Date: Jan 2025
% Finding the eigenvalue for harmonic oscillator potential by iteration with starting
% value E0.
% Input:
%  E0: the initial guess of the eigenvalue
%
% Output:
%  E: the eigenvalue near the initial guess of the system with harmonic oscillator
% potential

    K = mod(E0,4); % Identify even or odd cases

    psi0_even = 1; % 1 for even, 0 for odd
    dpsi0_even = 0; % 0 for even, 1 for odd
    psi0_odd = 0;
    dpsi0_odd = 1;

    x1 = 5;
    delta = 0.05;
    depth = 0.25; % Fluctuation depth
    x = linspace(0, x1, x1/delta);

    for i = 0:5 % iterate 5 times is enough
        if K > 2 % odd
            syms y
            f(y) = y^2 - E0;
            psi1 = solve_numerov(f-depth, x, psi0_odd, dpsi0_odd);
            psi2 = solve_numerov(f+depth, x, psi0_odd, dpsi0_odd);
            if abs(psi1(x1/delta)) < abs(psi2(x1/delta))
                E0 = E0 + depth;
            else
                E0 = E0 - depth;
            end
        elseif K < 2 % even
            syms y
            f(y) = y^2 - E0;
            psi1 = solve_numerov(f-depth, x, psi0_even, dpsi0_even);
            psi2 = solve_numerov(f+depth, x, psi0_even, dpsi0_even);
            if abs(psi1(x1/delta)) < abs(psi2(x1/delta))
                E0 = E0 + depth;
            else
                E0 = E0 - depth;
            end
        end
    end

    E = round(E0); % output
end
```

## 5.2 Main Scripts

Mostly repetitive, for producing different graphs of comparisons.

```
% Main Script 1

syms x

% Input
V(x) = x^2; % Potential
E1 = 0.95; % Energy eigenvalue
E2 = 1; % Note: 1,5,9,... are even cases; 3,7,11... are odd cases
E3 = 1.05;
x1 = 5; % upper limit
delta = 0.15; % spacing

psi0 = 1; % 1 for even, 0 for odd
dpsi0 = 0; % 0 for even, 1 for odd

% Programme
f(x) = V(x) - E1;
g(x) = V(x) - E2;
h(x) = V(x) - E3;
x = linspace(0, x1, x1/delta);

psi1 = solve_numerov(f, x, psi0, dpsi0);
psi2 = solve_numerov(g, x, psi0, dpsi0);
psi3 = solve_numerov(h, x, psi0, dpsi0);

n = 0; % Ground State
H = hermiteH(n, x).*exp(-x.^2/2);  % Hermite polynomial of degree n in x %(up to constants)

plot(x,psi2,'-o'); % graph of numerical n=0 case alone

plot(x, psi1,'-',x,psi2,'-o',x,psi3,'--'); % graph of numerical E1, E2, E3 together
legend('E = 0.95', 'E = 1', 'E = 1.05')

% plot(x,H,'-x',x,psi2,'-o'); % comparison of analytic with numerical
% legend('analytic','numerical')

% Main Script 2 -> Generates plots for several energy orders

syms x

% Input
V(x) = x^2; % Potential
E1 = 1; % Energy eigenvalue
E2 = 3; % Note: 1,5,9,... are even cases; 3,7,11... are odd cases
E3 = 5;
```

```
E4 = 7;
E5 = 9;
x1 = 5; % upper limit
delta = 0.05; % spacing

psi0_even = 1; % 1 for even, 0 for odd
dpsi0_even = 0; % 0 for even, 1 for odd
psi0_odd = 0;
dpsi0_odd = 1;

% Programme
f(x) = V(x) - E1;
g(x) = V(x) - E2;
h(x) = V(x) - E3;
k(x) = V(x) - E4;
p(x) = V(x) - E5;
x = linspace(0, x1, x1/delta);

psi1 = solve_numerov(f, x, psi0_even, dpsi0_even);
psi2 = solve_numerov(g, x, psi0_odd, dpsi0_odd);
psi3 = solve_numerov(h, x, psi0_even, dpsi0_even);
psi4 = solve_numerov(k, x, psi0_odd, dpsi0_odd);
psi5 = solve_numerov(p, x, psi0_even, dpsi0_even);

plot(x, psi1,'r',x,psi2,'b',x,psi3,'g',x,psi4,'magenta',x,psi5,'black');

legend('n = 1', 'n = 3', 'n = 5', 'n = 7', 'n = 9')

% Main Script 3 -> Generates all comparisons of analytic with numerical
syms x

% Input
V(x) = x^2; % Potential
E1 = 1; % Energy eigenvalue
E2 = 3; % Note: 1,5,9,... are even cases; 3,7,11... are odd cases
E3 = 5;
E4 = 7;
E5 = 9;
E6 = 11;
x1 = 6; % upper limit
delta = 0.05; % spacing

psi0_even = 1; % 1 for even, 0 for odd
dpsi0_even = 0; % 0 for even, 1 for odd
psi0_odd = 0;
dpsi0_odd = 1;

% Programme
```

```
f(x) = V(x) - E1;
g(x) = V(x) - E2;
h(x) = V(x) - E3;
k(x) = V(x) - E4;
p(x) = V(x) - E5;
q(x) = V(x) - E6;
x = linspace(0, x1, x1/delta);

% Numerical
psi1 = solve_numerov(f, x, psi0_even, dpsi0_even);
psi2 = solve_numerov(g, x, psi0_odd, dpsi0_odd);
psi3 = solve_numerov(h, x, psi0_even, dpsi0_even);
psi4 = solve_numerov(k, x, psi0_odd, dpsi0_odd);
psi5 = solve_numerov(p, x, psi0_even, dpsi0_even);
psi6 = solve_numerov(q, x, psi0_odd, dpsi0_odd);

% Analytic ("normalised" to correspond with numerical solution)
H1 = hermiteH(0, x).*exp(-x.^2/2);
H2 = hermiteH(1, x).*exp(-x.^2/2)/2;
H3 = hermiteH(2, x).*exp(-x.^2/2)/(-2);
H4 = hermiteH(3, x).*exp(-x.^2/2)/(-12);
H5 = hermiteH(4, x).*exp(-x.^2/2)/12;
H6 = hermiteH(5, x).*exp(-x.^2/2)/120;

% Subplots for all cases

subplot(3,2,1)
plot(x,H1,'-x',x,psi1,'-o'); % comparison of analytic with numerical
legend('analytic','numerical')
title('n=0, E=1')

subplot(3,2,2)
plot(x,H2,'-x',x,psi2,'-o');
legend('analytic','numerical')
title('n=1, E=3')

subplot(3,2,3)
plot(x,H3,'-x',x,psi3,'-o');
legend('analytic','numerical')
title('n=2, E=5')

subplot(3,2,4)
plot(x,H4,'-x',x,psi4,'-o');
legend('analytic','numerical')
title('n=3, E=7')

subplot(3,2,5)
plot(x,H5,'-x',x,psi5,'-o');
```

```
legend('analytic','numerical')
title('n=4, E=9')

subplot(3,2,6)
plot(x,H6,'-x',x,psi6,'-o');
legend('analytic','numerical')
title('n=5, E=11')

% Main Script 4

% Different deltas (for ground state)

syms x

% Input
V(x) = x^2; % Potential
E = 1; % Note: 1,5,9,... are even cases; 3,7,11... are odd cases
x1 = 6; % upper limit
delta1 = 0.05; % spacing
delta2 = 0.10;
delta3 = 0.15;
delta4 = 0.20;

psi0 = 1; % 1 for even, 0 for odd
dpsi0 = 0; % 0 for even, 1 for odd

% Programme
f(x) = V(x) - E;
X1 = linspace(0, x1, x1/delta1);
X2 = linspace(0, x1, x1/delta2);
X3 = linspace(0, x1, x1/delta3);
X4 = linspace(0, x1, x1/delta4);

psi1 = solve_numerov(f, X1, psi0, dpsi0);
psi2 = solve_numerov(f, X2, psi0, dpsi0);
psi3 = solve_numerov(f, X3, psi0, dpsi0);
psi4 = solve_numerov(f, X4, psi0, dpsi0);

n = 0; % Ground State
H = hermiteH(n, X1).*exp(-X1.^2/2);  % Hermite polynomial of degree n in x %(up to constants)

% Comparison of different spacing with analytic
plot(X1, psi1,'r',X2,psi2,'b',X3,psi3,'g',X4,psi4,'magenta',X1,H,'black');
legend('d=0.05','d=0.10','d=0.15','d=0.20','analytic')

% Main Script 5 % Different upper limits

syms x
```

```
% Input
V(x) = x^2; % Potential
E1 = 0.95; % Energy eigenvalue
E2 = 1; % Note: 1,5,9,... are even cases; 3,7,11... are odd cases
E3 = 1.05;
x1 = 5; % upper limit
x2 = 4;
x3 = 3;
x4 = 2;
delta = 0.05; % spacing

psi0 = 1; % 1 for even, 0 for odd
dpsi0 = 0; % 0 for even, 1 for odd

% Programme
f(x) = V(x) - E1;
g(x) = V(x) - E2;
h(x) = V(x) - E3;
X1 = linspace(0, x1, x1/delta);
X2 = linspace(0, x2, x2/delta);
X3 = linspace(0, x3, x3/delta);
X4 = linspace(0, x4, x4/delta);

psi1 = solve_numerov(f, X1, psi0, dpsi0);
psi2 = solve_numerov(g, X1, psi0, dpsi0);
psi3 = solve_numerov(h, X1, psi0, dpsi0);

psi4 = solve_numerov(f, X2, psi0, dpsi0);
psi5 = solve_numerov(g, X2, psi0, dpsi0);
psi6 = solve_numerov(h, X2, psi0, dpsi0);

psi7 = solve_numerov(f, X3, psi0, dpsi0);
psi8 = solve_numerov(g, X3, psi0, dpsi0);
psi9 = solve_numerov(h, X3, psi0, dpsi0);

psi10 = solve_numerov(f, X4, psi0, dpsi0);
psi11 = solve_numerov(g, X4, psi0, dpsi0);
psi12 = solve_numerov(h, X4, psi0, dpsi0);

subplot(2,2,1)
plot(X1, psi1,'-',X1,psi2,'-o',X1,psi3,'--'); % graph of numerical E1, E2, E3 together
legend('E = 0.95', 'E = 1', 'E = 1.05')
title('x1 = 5')

subplot(2,2,2)
plot(X2, psi4,'-',X2,psi5,'-o',X2,psi6,'--'); % graph of numerical E1, E2, E3 together
legend('E = 0.95', 'E = 1', 'E = 1.05')
```

```
title('x1 = 4')

subplot(2,2,3)
plot(X3, psi7,'-',X3,psi8,'-o',X3,psi9,'--'); % graph of numerical E1, E2, E3 together
legend('E = 0.95', 'E = 1', 'E = 1.05')
title('x1 = 3')

subplot(2,2,4)
plot(X4, psi10,'-',X4,psi11,'-o',X4,psi12,'--'); % graph of numerical E1, E2, E3 together
legend('E = 0.95', 'E = 1', 'E = 1.05')
title('x1 = 2')

% plot(x,H,'-x',x,psi2,'-o'); % comparison of analytic with numerical
% legend('analytic','numerical')
```