University of Cape Town

PHY3004W

Y3 Physics Project

Computational Analysis of Numerov Shooting Methods for the Time Independent Schrödinger Equation

Author:
Zayd Pandit

Student Number:
PNDZAY001

September 2021

1 Abstract

Numerov's Method with a shooting algorithm is a popular technique for numerically solving the Schrödinger equation. We solve the one-dimensional Time Independent Schrödinger Equation with a Harmonic Potential using two very different Numerov shooting methods - the Method of Matching Derivatives and the so called 'Wag-the-Tail' Bisection method. We find that the Method of Matching Derivatives is more efficient and more reliably produces an accurate eigenstate wavefunction. We also find that the Wag-the-tail Bisection method is an extremely robust and accurate method, and far outperforms the Method of Matching Derivatives at determining the energy eigenvalues of our system to a high accuracy and precision.

Contents

2 Introduction and Theory

Numerov's Method is a numerical method used to solve second order ODE's with no first order term. It is a fourth-order linear multistep method [8].

Numerov's Method can be applied to equations of the form:

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

We will use Numerov's method to numerically integrate the 1-D Time Independent Schrödinger Equation (TISE), to extract the eigenstates and eigenvalues of our system. Solving for the eigenstates is of great importance as they provide a basis for the Hilbert space of our quantum system and their linear combination forms the general solution to the Time Dependent Schrödinger Equation (TDSE). The eigenvalues give us the spectrum of possible energies for the system [6].

The TISE can be rearranged into the form of 1 where we can see that the non-homogeneous term, $g(x)$, is zero:

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

$$\frac{d^2\psi}{dx^2} + \frac{2m}{\hbar^2}(E - V(x))\psi = 0 \tag{3}$$

If we discretise the wavefunction on our desired interval into equidistant points with separation h, we can numerically integrate forwards with the update scheme given by:

$$\left(1 + \frac{h^2}{12}f_{i+1}\right)\psi_{i+1} = 2\psi_i\left(1 - \frac{5h^2}{12}f_i\right) - \psi_{i-1}\left(1 + \frac{h^2}{12}f_{i-1}\right) \tag{4}$$

where $f_i = \frac{2m}{\hbar^2}(E - V(x_i))$. This numerical update scheme is derived in Appendix A

The potential, $V(x)$, we use throughout this investigation is that of the Quantum Harmonic Oscillator (QHO). The QHO is a system of extreme importance as any arbitrary potential can be approximated by a harmonic potential near a stable equilibrium point [6]. The QHO also has well known, exact analytical solutions, which allows us to create precise metrics for the accuracy and reliability of our numerical schemes. We can then, with much greater confidence, begin applying these numerical schemes to systems which do not have closed-form analytical solutions.

The Quantum Harmonic Oscillator (QHO) is the quantum analogue of the classical harmonic oscillator. It has the quadratic potential $V(x) = \frac{1}{2}m\omega^2x^2$ and discrete energy levels $E_n = \hbar\omega(n + \frac{1}{2})$, with $n = 0, 1, 2, ...$.

Through any one of a variety of methods, the eigenstates of the QHO can be found explicitly to be [6]:

$$\psi(x) = \frac{1}{\sqrt{2^n n!}} \left(\frac{m\omega}{\pi\hbar}\right)^{1/4} e^{-\frac{m\omega x^2}{2\hbar}} H_n\left(\sqrt{\frac{m\omega}{\hbar}}x\right) \tag{5}$$

where the H_n are the physicists' Hermite Polynomials [7]:

$$H_n(z) = (-1)^n e^{z^2} \frac{d^n}{dz^n}\left(e^{-z^2}\right) \tag{6}$$

The simplest way to implement the Hermite Polynomials in Python is to use the `scipy.special.hermite` functions [3].

For simplicity, we will set $\hbar = m = 1$. Our QHO potential will thus be described by the quadratic function $V(x) = \frac{1}{2}x^2$, and our spectrum of discrete energy eigenvalues is given by $E_n = n + \frac{1}{2}$.

3 The Numerov Shooting Method

3.1 Method of Matching Derivatives (MoMD)

According to the postulates of Quantum Mechanics the wavefunction should be continuous across the domain[6]. Furthermore, unless the potential contains an infinite discontinuity, the first derivative of the wavefunction should also be continuous [6]. The Method of Matching Derivatives leverages this constraint to construct an approximate wavefunction by optimizing the eigenstate energy such that the first derivative of the wavefunction is continuous. The algorithm goes as follows:

1. In a function which takes the energy (E) and the eigenstate number (n) as argument:
   (a) Determine the classical turning point[1] , $x = x_{TP}$, which is found by determining the index of the array of the total domain at which $|E - V_i|$ is minimised.
   (b) Define a sub-domain, a subset of the total domain, from $x = 0$ to one index beyond x_{TP} [2]
   (c) Use Numerov's method to integrate across this sub-domain, call it ψ_L, and have the function return it. For even n, use $\psi(0) = 1$ and $\psi(h) = 1 - \delta$. For odd n, use $\psi(0) = 0$ and $\psi(h) = \delta$, where δ is some small positive number.
2. In a function which takes the energy (E) and the eigenstate number (n) as argument:
   (a) Define a sub-domain, a subset of the total domain, from $x = b$ to one index less than x_{TP}. Here our domain endpoint b is an x value far[3] into the classically forbidden region ($V(x) > E$).
   (b) Use Numerov's method to integrate across this domain, call it ψ_R, and have the function return it. For even and odd n, use $\psi(b) = \delta$ and $\psi(b-h) = 2\times\delta$, where δ is some small positive number.
3. As a function of a specific E and n, determine ψ_L and ψ_R using the functions in Steps 1 and 2. Re-scale ψ_R such that $\psi_R(x_{TP}) = \psi_L(x_{TP})$. Re-scaling of the wavefunction is allowed as will be normalizing the wavefunction at a later stage.
4. Define a function (`Derivative Difference Function`) which evaluates the difference in the first derivatives of ψ_L and ψ_R using a first order central difference formula at the classical turning point as a function of the Energy, E.

[1]The 'classical turning point' is defined as the x position at which the energy E is equal to the Potential energy, $V(x)$

[3]The effect of changing the domain endpoint, b, will be investigated

5. The root finding method `scipy.optimize.fsolve` [2] is used to optimise the energy parameter in `Derivative Difference Function` such that the difference in first derivatives of ψ_L and ψ_R at the classical turning point is zero. If other arguments are present in `Derivative Difference Function`, a Lambda Function can be used to only pass the energy argument as a parameter for the root solving method to optimise.

6. We find the wavefunction with the energy eigenvalue calculated in Step 5 by using Numerov's Method to integrate across the whole domain. We then normalize to obtain our desired eigenstate.

3.2 Wag-the-Tail Bisection Method

In the classically forbidden region, $\{x \in [0, b] : V(x) > E\}$ [2], the eigenstate wavefunction of the QHO system should decay to zero [6]. Thus if the wavefunction obtained by numerically integrating from $x = 0$, becomes unstable quickly in this region and 'blows-up' exponentially to $+\infty$ or $-\infty$, we assume that it is not our desired eigenstate wavefunction. The better our approximation of the eigenvalue and eigenstate, the longer the domain of stability will be [9].

We expect that two wavefunctions corresponding to two energies bounding a single eigenvalue, should have opposite signs at $x = b$ [9]. Thus we can narrow our interval of eigenvalues using the familiar Bisection method root finding algorithm, which is simple and has guaranteed convergence (given that the initial lower and upper bounds bound a root) [5]. The Wag-the-Tail Bisection Method algorithm goes as follows:

1. In a function which takes energy (E) and the eigenstate number (n) as an argument, integrate from $x = 0$ to some domain endpoint $x = b$ far [3] outside the classically allowed region. We call this function `Numerov Integration Function` and call its output wavefunction ψ.

2. We construct a function which we call `Sign Function`, that takes energy as an argument, and then uses Numerov integration to integrate from $x = 0$ outwards to $x = b$. The function then returns the sign of $\psi(b)$. We need to make sure to catch any cases of the wavefunction blowing up to a value too large for the array element data type to handle before we have integrated all the way to $x = b$. This can be done by setting a large tolerance value (in our case 10^{12}) where, if ψ exceeds it we stop the integration and return the sign of the value of the last point of the wavefunction.

3. We start with initial bounds - a set of 2 energies, E_{up} and E_{lw}, bounding an eigenvalue. These two energies will correspond to two wavefunctions with different signs at $x = b$, but with the same number of nodes [9]. We then use the bisection method to narrow down the interval of the eigenvalue. To this end, we construct a recursive function, `Bisection Function`, which takes the two energy

bounds as arguments. The `Bisection Function` calls the `Sign Function` for energies E_{up}, E_{lw} and E_{mid}, where $E_{mid} = \frac{E_{up}+E_{lw}}{2}$.

(a) If $\text{sign}(E_{lw}) = \text{sign}(E_{up})$, our function raises an exception stating that E_{up} and E_{lw} do not bound an eigenvalue

(b) If $\text{sign}(E_{lw}) \neq \text{sign}(E_{up})$ and $\text{sign}(E_{lw}) = \text{sign}(E_{mid})$, we call the Bisection Function with E_{mid} replacing E_{lw}

(c) If $\text{sign}(E_{lw}) \neq \text{sign}(E_{up})$ and $\text{sign}(E_{up}) = \text{sign}(E_{mid})$, we call the Bisection Function with E_{mid} replacing E_{up}

We terminate the recursion when two consecutive iterates (the iterates being the middle value of our eigenvalue interval) have an absolute difference of at most 1.489102×10^{-8}, and then return the midpoint of the energy interval in the final iteration. [4]

4. We then use the value of the energy eigenvalue provided by the `Bisection Function`, plug that into the `Numerov Integration Function` and normalize the resulting wavefunction to obtain our eigenstate.

[2]The solutions to the TISE with a symmetric potential have a definite parity- this can be seen by reflecting $x \to -x$ and observing that $\psi(-x)$ satisfies the same equation as $\psi(x)$ up to a normalization factor A with $|A| = 1$. So we need only find a solution on the interval $x \in [0, b]$, and reflect the solution using $\psi(x) = \psi(-x)$ and $\psi(x) = -\psi(-x)$ for even and odd wavefunctions respectively.

[4]This termination method is not typical for Bisection methods, and was selected as it is the same criterion for termination that is used by `scipy.optimize.fsolve` [2] [4], which is implemented in the MoMD to find the eigenvalue.

4 Results and Analysis

4.1 Eigenstate Results

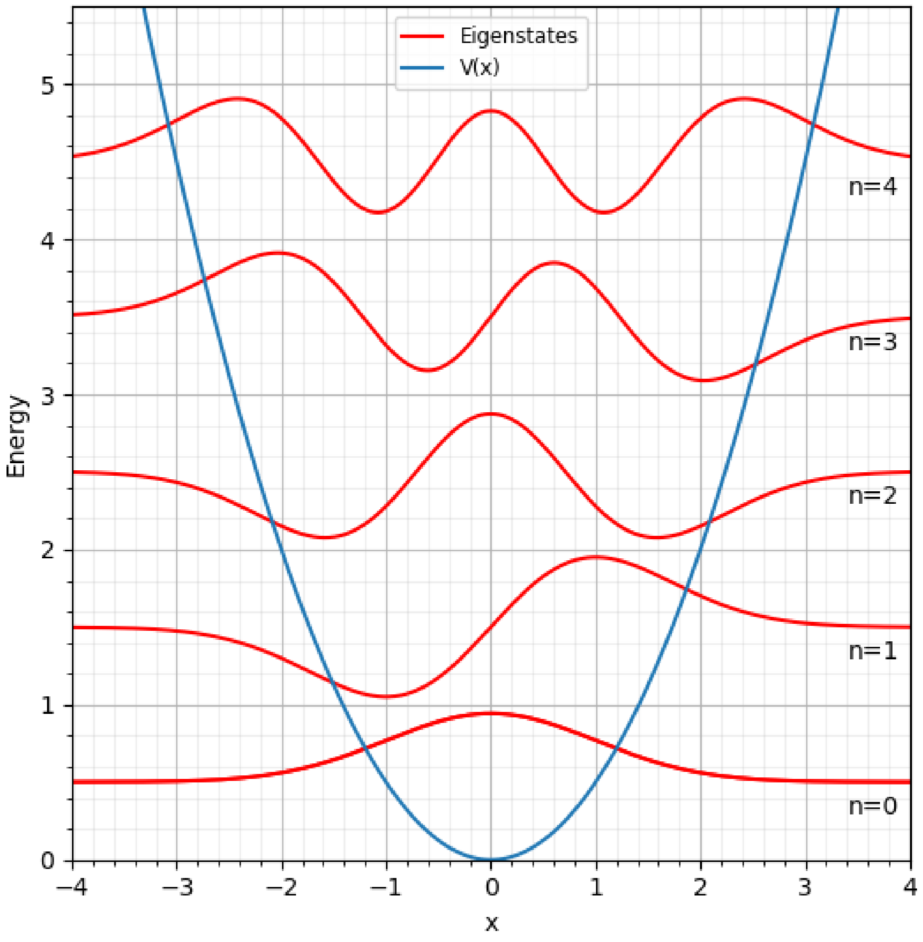

Figure 1: The first 5 eigenstates of the Quantum Harmonic Oscillator, as calculated by the Wag-the-Tail Bisection method. Each eigenstate has been plotted at a height corresponding to its associated eigenvalue. Note: In order to have the wavefunctions not overlap, they were all scaled by a factor of 0.7 when plotting, so they are not normalised relative to the scale given on the y-axis.

4.2 Running Time Analysis for Eigenvalue Determination

The running time of each algorithm is a vital metric of its efficiency. The running time is a proxy for the rate of convergence of an algorithm, as these should be roughly proportional. We use a single eigenstate, $n = 3$, fix the domain endpoint at $x = 6.0$ and fix the convergence tolerance at 1.489102×10^{-8}.

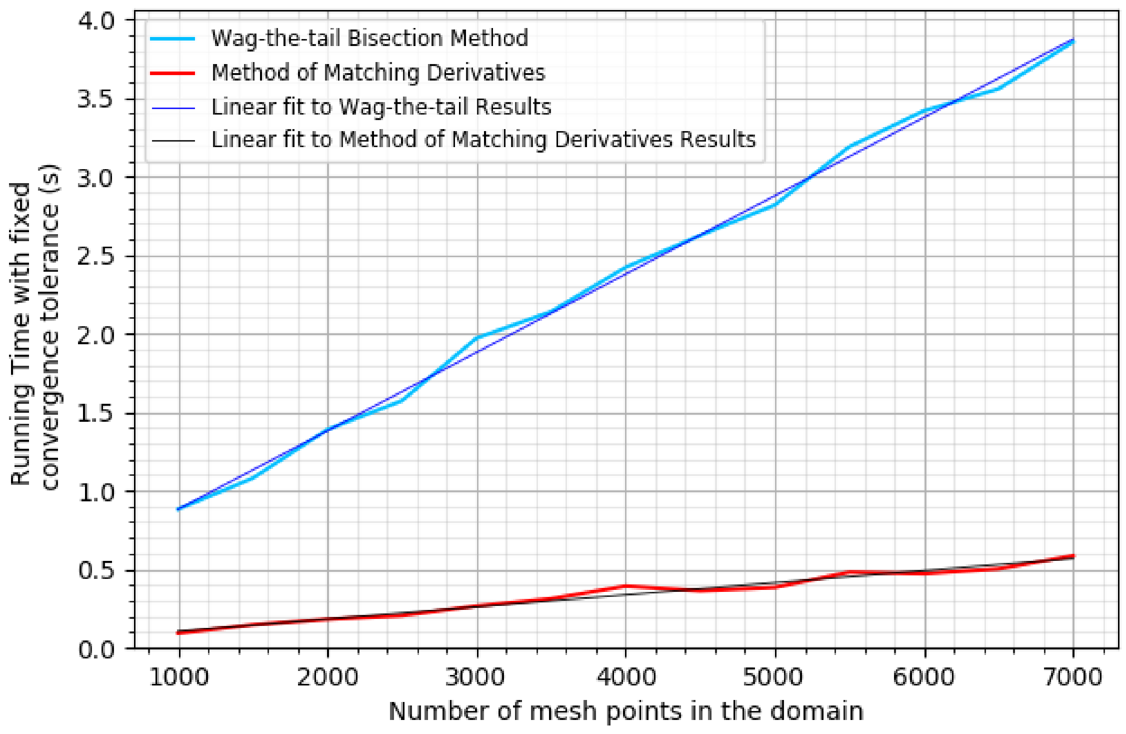

Figure 2: The running time of the Wag-the-Tail method and the MoMD for E_3 as a function of the number of mesh points in the domain. MoMD has an initial guess of 3.4 and Wag-the-Tail Bisection method is given initial bounds of $E_{lw} = 3.4$ and $E_{up} = 3.6$. The linear fit of the Bisection method results has a slope parameter $m = (477.7 \pm 9.0) \times 10^{-6}$ and a y-intercept parameter $c = (19.6 \pm 4.0) \times 10^{-3}$. The linear fit to the Method of Matching Derivatives results has a slope parameter $m = (71.9 \pm 1.8) \times 10^{-6}$ and a y-intercept parameter of $c = (16.2 \pm 8.1) \times 10^{-2}$

We can see that the relationship between number of mesh points and the running time is linear for both methods.

In the MoMD, we use `scipy.optimize.fsolve` to find the root of the `Derivative Difference Function` mentioned in Step 4 of the MoMD algorithm. `scipy.optimize.fsolve` uses a modified Powell method as its iterative optimisation algorithm [4] [1]. In the `Derivative Difference Function`, we use Numerov integration to find ψ_L and ψ_R

over a total of N mesh points. This means the running time per iteration (an iteration being a call of the `Derivative Difference Function`) will increase directly proportionally with N. Thus the linearity of the results suggests that the number of iterations to convergence is roughly constant for different values of N.

The Bisection method is known to converge linearly, but extremely slowly. This is confirmed in Fig 2. The convergence is linear because at each iteration the absolute error is halved [5]. The bisection is method has a really good Worst-case Complexity, but its Average-case Complexity (average performance over all possible inputs) is really bad as it will converge to a tolerance in the same time, irrespective of the inputs themselves - only the width of the initial interval contributes to the number of iterations to convergence [5]. Since the bisection method will be converging in the same number of iterations independent of N, the increase in running time comes from the fact that in each iteration, we have to do Numerov integration across N points 3 times, with energies E_{up}, E_{lw}, and E_{mid}.

Due to the fact that the Bisection method has to use Numerov integration over $3N$ points per iteration, while the MoMD uses Numerov integration over N points per iteration, the gradient of the Bisection method in Fig 2 being much steeper than that of the line for the MoMD is expected.

We can use the linear fits to extrapolate and determine the running time for higher numbers of mesh points.

4.3 Residual Analysis

We use a residual analysis as a metric for the accuracy of the numerical eigenstate wavefunction. The average residual per mesh point is calculated by taking the absolute value of the difference between the numerical eigenstate and the analytical solution at each mesh point, summing over all points, and dividing by the number of mesh points.

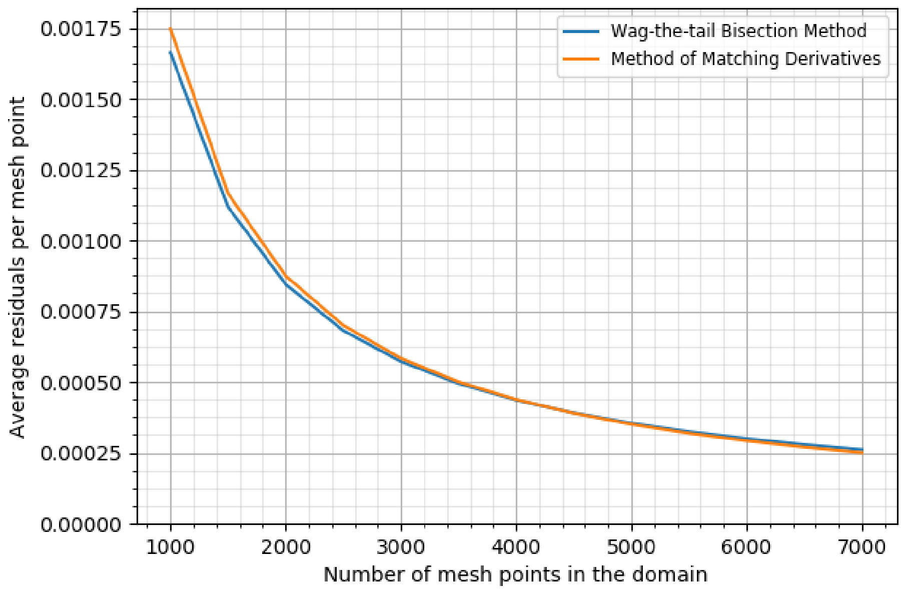

Figure 3: Average residuals per mesh point as a function of the number of mesh points - for both numerical methods. The domain endpoint is kept at a constant $x = 6.0$.

In Fig 3 we can see that both methods have near identical error in the numerical wavefunctions they produce - no matter the number of mesh points. Numerov's method has a truncation error of $O(h^6)$ and the interval size is given by $h = b/N$, so we expect the numerical error due to discretisation to decrease with an increase in N, and for the decrease in error to decrease as N increases. This expectation is indeed reflected in Fig 3.

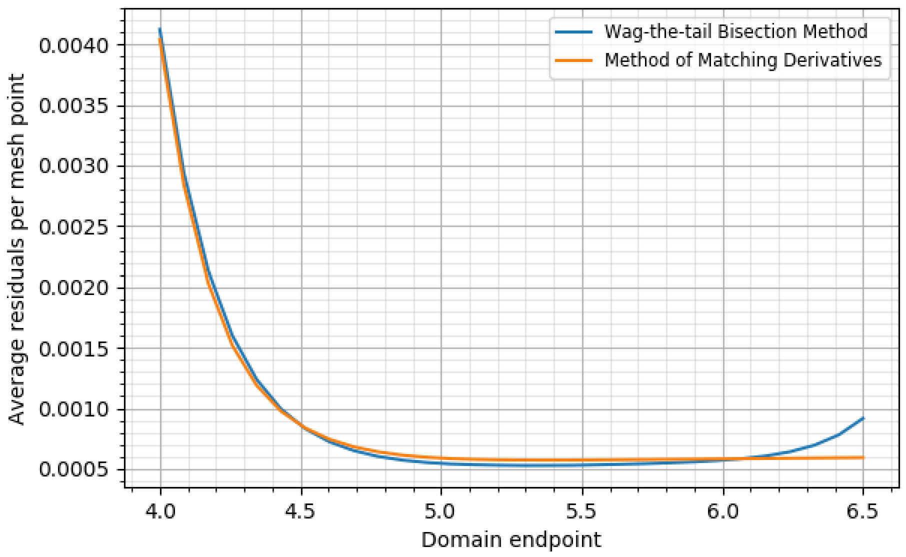

Figure 4: Average residuals per mesh point as a function of the domain endpoint, for both numerical methods. The number of mesh points is kept at a constant $N = 3000$

The wavefunction produced by the MoMD initially drastically increases in its agreement with the analytical solution as we increase our domain endpoint from 4.0, and then plateaus at an average residual per mesh point of 0.0006.

The Bisection method has an inherit discomfort with large values of the domain endpoint. In the MoMD, we fix the value of the wavefunction at $x = b$ (which gets normalised later), but in the Bisection method, we just integrate from $x = 0$ outwards - the numerical error incurred at each spatial step will compound and eventually lead to instability and divergence since the solutions to the TISE are exponential solutions for x where $E < V(x)$. The wavefunction for $n = 3$ and the domain endpoint $b = 6.5$ is shown in Appendix B - the 'tail' created by the instability near $x = 6.5$ is clearly visible.

On the domain $x \in [4.0, 6.0]$, both methods have nearly identical average residuals per mesh point. The Bisection method plateaus slightly lower than the MoMD, to about an average residual per mesh point of 0.00055. This makes sense as the Bisection method produces a more accurate eigenvalue than the MoMD (this is confirmed in Fig 5), which results in a slightly more accurate eigenstate wavefunction.

4.4 Eigenvalue Accuracy Analysis

One of the most important metrics of our numerical scheme that we need to consider is the accuracy with which it can calculate the energy eigenvalues. We will use the absolute difference between the numerical and analytical eigenvalue as our measure of eigenvalue accuracy. Recall that the analytical eigenvalue for $n = 3$ is $E_3 = 3.5$.

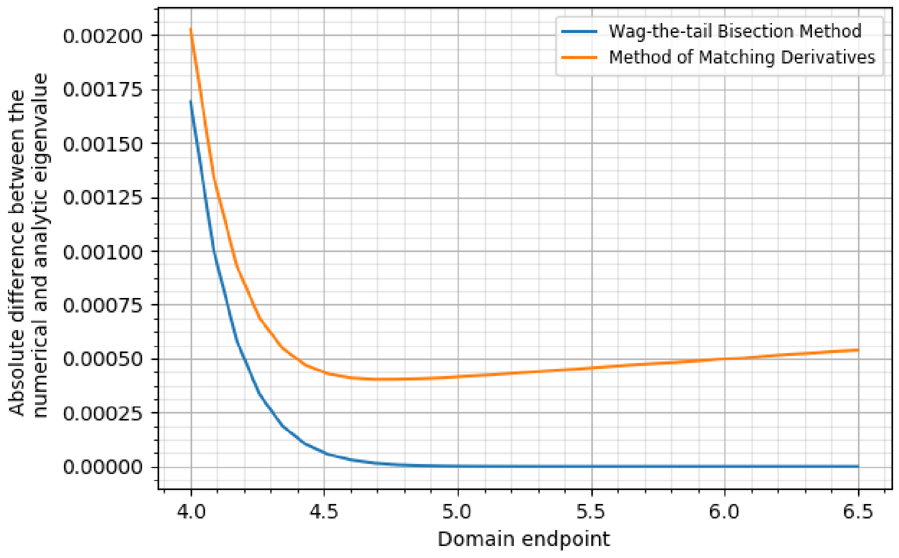

Figure 5: Eigenvalue accuracy as a function of domain endpoint. We set $n = 3$ and the number of mesh points $N = 3000$

In Fig 5 we see some really interesting and noteworthy behaviour of our numerical schemes. For the Method of Matching Derivatives, we observe that the accuracy decreases as the domain endpoint increases beyond some optimal value. The Wag-the-tail bisection method, however, just gets more accurate as you increase the domain endpoint up to an asymptote brought about by numerical error during discretization.

The existence of a minimum in the MoMD plot can be explained as follows: Increasing the domain endpoint from x just greater than x_{TP} increases the accuracy due to an improvement in the approximation of the wavefunction decay at infinity - when we integrate from the right to find ψ_R, we guess the first two points with some finite, non-zero values. At large values of the domain endpoint, the increase of b results in a much larger discretisation error of the wavefunction due a large portion of the wavefunction being integrated in the classically forbidden region (where the solutions take the form of exponentials, which can change rapidly). The increase in accuracy due to the improvement of the approximation of the endpoint will diminish with larger x. Meanwhile the decrease in accuracy due to discretisation over a larger interval (and a larger classically forbidden interval) will increase with larger x.

	Absolute difference between the numerical and analytic eigenvalue for E_3 (10^{-4})	
Number of mesh points	MoMD	Wag-the-Tail Bisection method
1000	14.6062	0.000119
2000	7.4316	0.000119
3000	4.9219	0.000119
4000	3.7137	0.000119
5000	2.9596	0.000119
6000	2.4907	0.000119
7000	2.1160	0.000119

Table 1: The error in the eigenvalue extended in both methods as a function of N

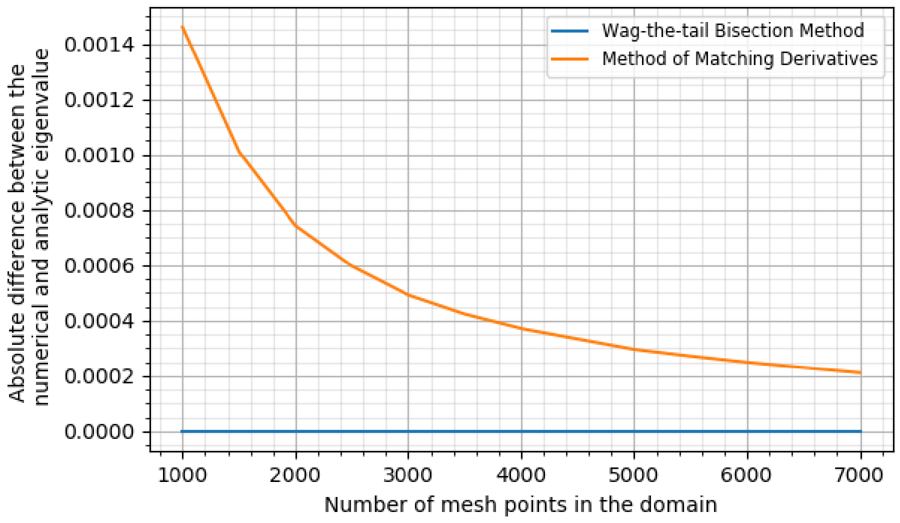

Figure 6: Eigenvalue accuracy as a function of the number of mesh points, for $n = 3$ and $b = 6.5$.

In Fig 6 we can clearly see the error of the eigenvalue decrease somewhat exponentially for the MoMD as we increase the number of points in the domain. In Table 1 we can see that the error from the Bisection method is in fact non-zero, it is just so small compared to the error of the MoMD that it appears to be zero in Fig 6.

The plot for the Bisection method in Fig 6, is not intuitive, but is in fact the expected result. Consider Step 3 in the algorithm, in which we recursively call the bisection function with a smaller and smaller interval bounding the eigenvalue. This bisection function terminates when consecutive guesses of the energy eigenvalue ($E_{mid} = \frac{E_{up}+E_{lw}}{2}$) are within a specific tolerance, `xtol`. In Fig 6, we start with initial bounds of 3.4 and 3.6. What determines how the interval will be divided in the next call of the function is the sign of the endpoint of the wavefunction with energies given by the bounds and the midpoint, E_{mid}. Only if the extra discretisation error incurred by reducing N will change the wavefunction enough to flip the sign of the wavefunction at the bounds, will the energy bounds at some iteration differ. Since, even for differing N, we will have identical sequences of energy bounds, the iteration at which the function terminates will be identical, and we end up with identical eigenvalues.

5 Conclusion

The Wag-the-Tail Bisection method is an extremely robust and accurate numerical scheme for finding the eigenvalues and eigenstates of the system. We find that its eigenstate wavefunctions increase in accuracy with an increase in the number of mesh points on the domain. Its eigenvalues however, remains constant with increasing N, due to the nature of the convergence criterion of the energy optimisation algorithm. Increasing the domain endpoint increases the accuracy of the eigenvalue. However, the wavefunction it produces for the eigenstate can be inaccurate near the edges if the domain end point is set too far outside the classically allowed region.

The Method of Matching Derivatives is an extremely simple and computationally efficient algorithm. The MoMD does not suffer from the same error in the eigenstate wavefunction of 'tails' at the domain endpoints that the Bisection method does. We found the accuracy of the eigenstate wavefunction produced by MoMD and Bisection method to be nearly identical, and that the error decreases steeply as we increase the number of mesh points. Thus, the Method of Matching Derivatives is a slightly more reliable method for finding an accurate numerical eigenstate wavefunction.

The Wag-the-Tail Method can calculate the eigenvalues to a far higher precision than the MoMD - in fact, it can be used to calculate the eigenvalue to any desired precision by changing the convergence tolerance. The error in the eigenvalue of the MoMD as a function of domain endpoint used has a minimum - an optimal point - which still has a much higher error than the Wag-the-Tail method eigenvalue with the same convergence tolerance. Thus, the Wag-the-Tail method is a more accurate and reliable method for determining the energy eigenvalues of the stationary states of our quantum system.

This investigation could be extended to compare the two Numerov Shooting methods for solving the TISE with arbitrary symmetric potentials.

References

[1] S Burton; Garbow; et al. *Documentation for MINPACK subroutine HYBRD*. URL: https://www.math.utah.edu/software/minpack/minpack/hybrd.html.

[2] The Scipy community. *Documentation for `scipy.optimize.fsolve`*. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fsolve.html.

[3] The Scipy community. *`scipy.special.hermite`*. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.hermite.html.

[4] The Scipy community. *Source Code for `scipy.optimize`*. URL: https://github.com/scipy/scipy/blob/v1.7.1/scipy/optimize/minpack.py#L46-L178.

[5] Richard Burden; Douglas Faires. "Numerical Analysis". In: PWS Publishers, 1985. Chap. 2.1: The Bisection Algorithm.

[6] David Griffiths. *Introduction to Quantum Mechanics (2nd Edition)*. Prentice Hall, 2004.

[7] Eric Weisstein. *Hermite Polynomial*. URL: https://mathworld.wolfram.com/HermitePolynomial.html.

[8] Qijing Zheng. *Numerov Algorithm*. URL: http://staff.ustc.edu.cn/~zqj/posts/Numerov-Algorithm/.

[9] Barton Zwiebach. *MiT 8.04: Quantum Physics I - Lecture 12.6: Energy eigenstates on a generic symmetric potential. Shooting method*. Spring 2016. URL: https://ocw.mit.edu/courses/physics/8-04-quantum-physics-i-spring-2016/video-lectures/part-2/.

Appendix

A: Derivation of the Numerov method update scheme

We can expand the function y(x) about the point $x = x_0$ to get:

$$y(x) = y(x_0) + (x - x_0)y'(x_0) + \frac{(x-x_0)^2}{2!}y''(x_0) + \frac{(x-x_0)^3}{3!}y'''(x_0) + \frac{(x-x_0)^4}{4!}y''''(x_0)$$

Substituting $x = x_0 + h$, and $x = x_0 - h$:

$$y_{n+1} = y_n + hy'_n + \frac{h^2}{2}y''_n + \frac{h^3}{3!}y'''_n + \frac{h^4}{4!}y^{(4)}_n + ... \tag{7}$$

$$y_{n-1} = y_n - hy'_n + \frac{h^2}{2}y''_n - \frac{h^3}{3!}y'''_n + \frac{h^4}{4!}y^{(4)}_n - ... \tag{8}$$

Adding 7 and 8 together and rearranging gives us a central difference equation for $y^{(4)}_n$ in 9. Truncating 7 and 8 to the second order, then adding and rearranging gives us the second order central difference equation, 10:

$$y^{(4)}_n = \frac{12}{h^4}(y_{n+1} + y_{n-1} - 2y_n - h^2y''_n) \tag{9}$$

$$y''_n = \frac{1}{h^2}(y_{n+1} - 2y_n + y_{n-1}) \tag{10}$$

The general form of the equations handled by the Numerov Method is given in 11. We can differentiate it twice with respect to x to get 12, then we can use 10 to rewrite the second derivatives on the RHS of 12 to get 13:

$$\frac{d^2y}{dx^2} + f(x)y = g(x) \tag{11}$$

$$\frac{d^4y}{dx^4} = \frac{d^2}{dx^2}(g(x) - f(x)y) \tag{12}$$

$$\frac{d^4y}{dx^4} = \frac{1}{h^2}(g_{n-1} - 2g_n + g_{n+1} - f_{n-1}y_{n-1} + 2f_ny_n - f_{n+1}y_{n+1}) \tag{13}$$

Equating 13 and 9 and multiplying through by $\frac{h^4}{12}$ gives:

$$y_{n+1} + y_{n-1} - 2y_n - h^2y''_n = \frac{h^2}{12}(g_{n+1} - 2g_n + g_{n-1}f_{n-1}y_{n-1} + 2f_ny_n - f_{n+1}y_{n+1}) \tag{14}$$

Lastly, we substitute 11 into y''_n in the above equation and rearrange such that all variables at step $n + 1$ are on the left hand side:

$$\left(1 + \frac{h^2}{12}f_{n+1}\right)y_{n+1} = y_n\left(2 - \frac{5h^2}{6}f_n\right) - y_{i-1}\left(1 + \frac{h^2}{12}f_{n-1}\right)\frac{h^2}{12}(g_{n-1} + 10g_n + g_{n+1}) \tag{15}$$

Now in application to the TISE, in the form 3, we notice that $g(x) = 0$, $f(x) = \frac{2m}{\hbar^2}(E - V(x))$ (which we'll label as f_i) and $y = \psi$. Giving us our final update equation, equation 4.

B: Eigenstate 3 calculated by the bisection method over domain with end-point $x = 6.5$

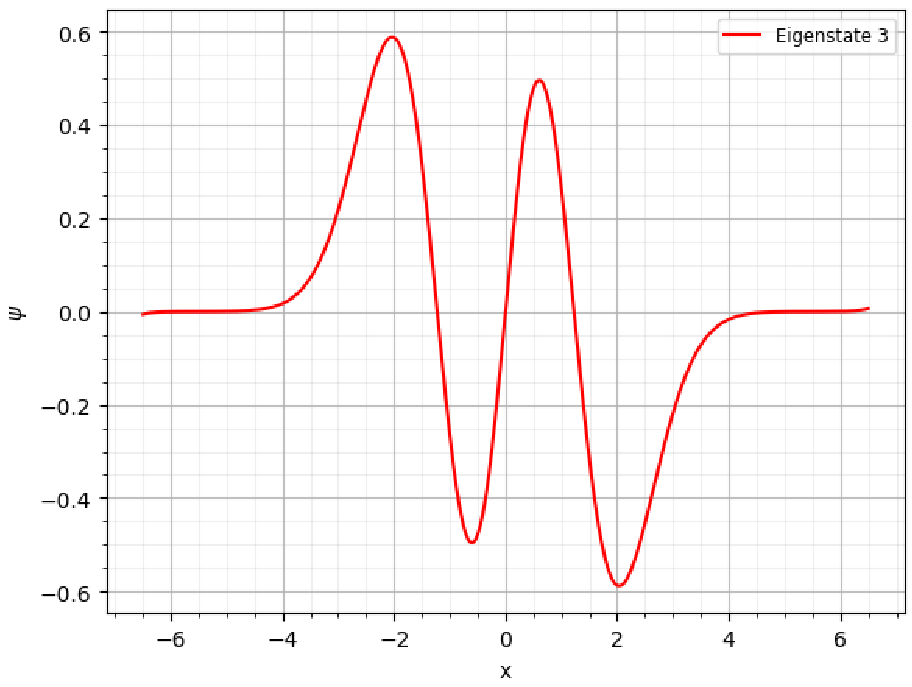

Figure 7: Eigenstate 3 of the QHO calculated by the wag-the-tail bisection method over domain $[-6.5, 6.5]$ with 4000 points