Sean Heffley 12/11/2022

PHY 446

modeling the 1d time-independent Schrodinger equation

Quantum mechanics can be very unintuitive and difficult to explain concepts, such as why certain energy levels exist, outside of an academic environment with background understanding. Additionally, finding solutions to the schrodinger equation with different potential functions can rely on computationally finding solutions to transcendental functions. Numeric solutions are able to be executed in terms of other functions and can be done so by computers very quickly. As such, an interactive model with visual feedback was a major focus when designing this program. This also left the initial values, potential function, mass, and energy defined such that an algorithmic search could be relatively easily applied. Such a search may be manually undertaken in regards to energy in order to find bound energy states and see when the particle becomes unbounded.

The program is written in python and relies on the numpy, matplotlib, scipy, and decimal modules. It can be run from an IDE, command prompt, or by double clicking on it. If any modules fail to import a message will be printed “failed to import <module>” and the program will pause until the user closes it or enters an input to proceed anyways. The program uses 2 different solutions to a 1d time-independent schrodinger equation which may be independently toggled and controlled: The Numerov solution and a system of first order differential equations. Both of these methods rely on initial values located on the sliders to the right of the plot. There are two buttons located at the below and towards the right of the plot which display the normalization and toggle each solution. These toggles and initial value sliders are color coded to match the legend and solutions. The normalization, A, is displayed such that

$$\int_D \psi^2 dx = A^2, \quad (1)$$

for the currently plotted solution. When applied the normalization of the wave function is the inverse of A and the expected value to be displayed is 1. This makes the probability density over the domain, but not all space, unity.

There are a set of buttons along the bottom the the graph to toggle normalization, limiting the range, and plotting ψ^2 as probability density along with each solution.

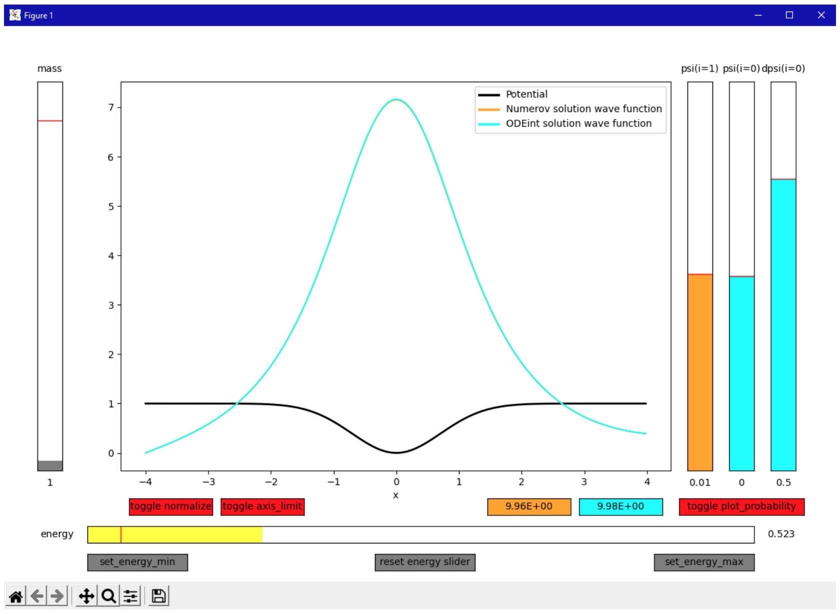

Fig.1 : program interface

Manually searching for a bound energy level can be done using the grey buttons below the energy slider. First an energy level must be identified, this can be seen around each energy level by moving the energy slider as the wave solution alternates from diverging from positive to negative. This can be seen with higher mass values as an additional line while limiting axes and not applying normalization as shown in Figure 2

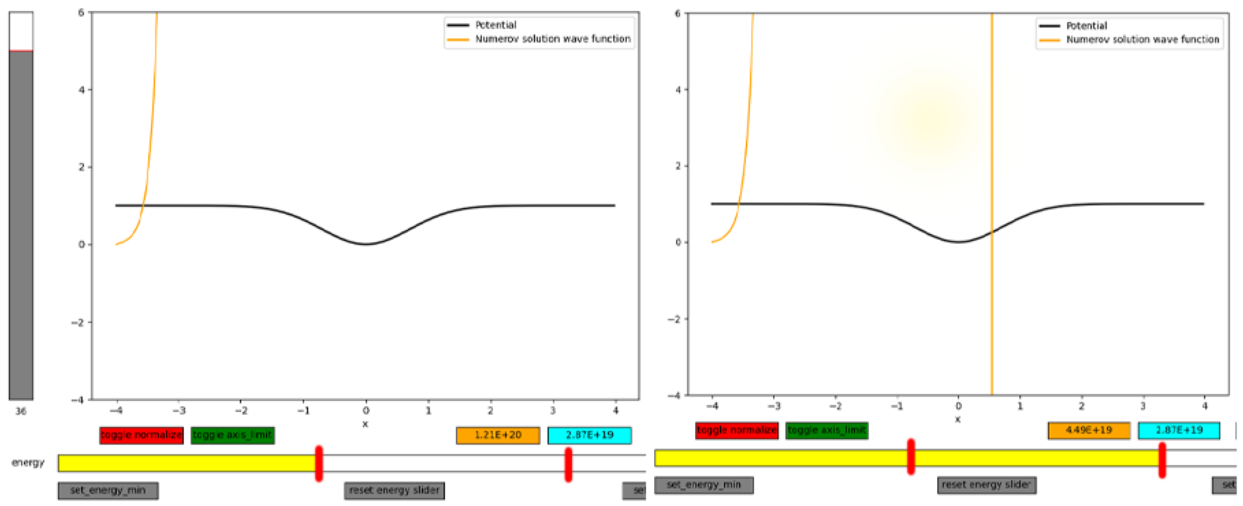

Fig.2 : identifying an energy value of interest

Where the energy level of interest lies somewhere between the 2 energy values highlighted in red. The min or max energy bound of the slider can be set to the value it currently is set at by clicking on the respective button thus increasing precision of the value set. I.e. if the energy value of interest is greater than the current value set the min energy bound and vice versa. The bounds of the slider can be reset using the reset energy slider button. After being repeated several times, particularly for lower energy levels, normalization can be turned back on and energy adjusted until the solution approaches 0 at both ends of the domain. At these points, interestingly, the form of the solution is insensitive to the initial values except for sign, but the normalization does change. An example of finding a bound energy state : https://youtu.be/zUalyMsnD7U

A word of caution, setting the initial values to 0 may give strange results especially pertaining to the ODE solution. Setting the min and max bounds on the energy slider to the same value prevents the energy from being changed and may freeze the program. Not-a-number values of normalization as well as overflow errors may result from radically changing the potential function i.e. attempting to model an approximation to the infinite square well.

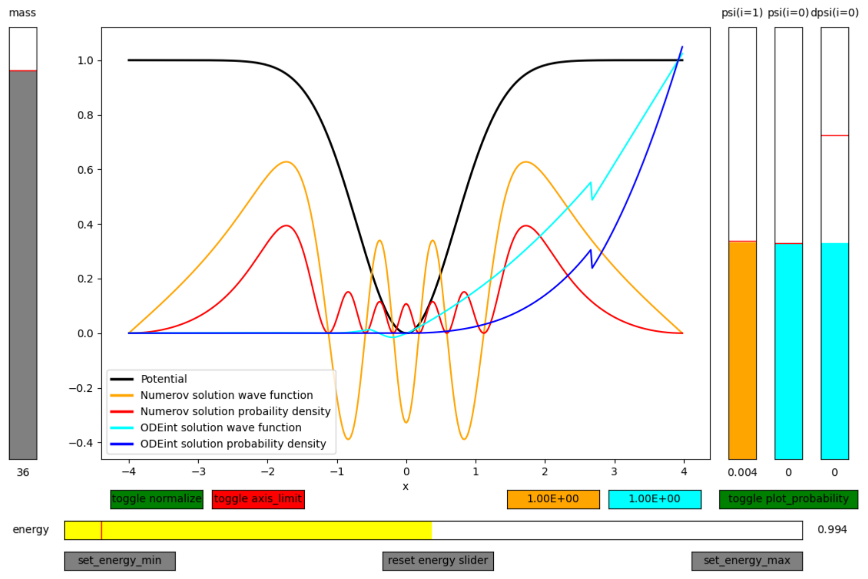

Fig. 3 erroneous solution from initial values of 0

Examples of select energies

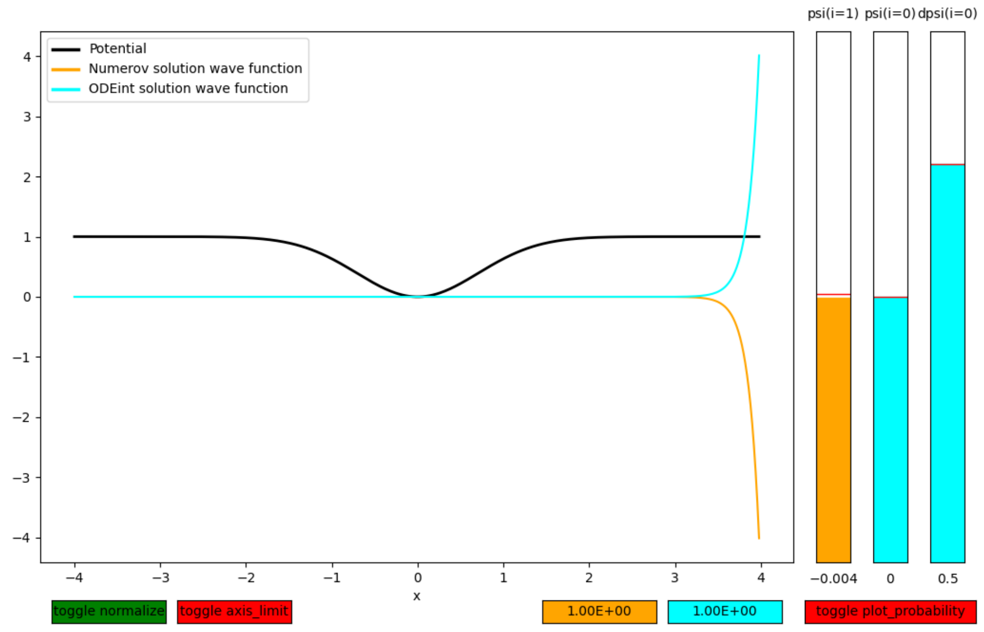

Fig. 2 E=0.1 M= 36

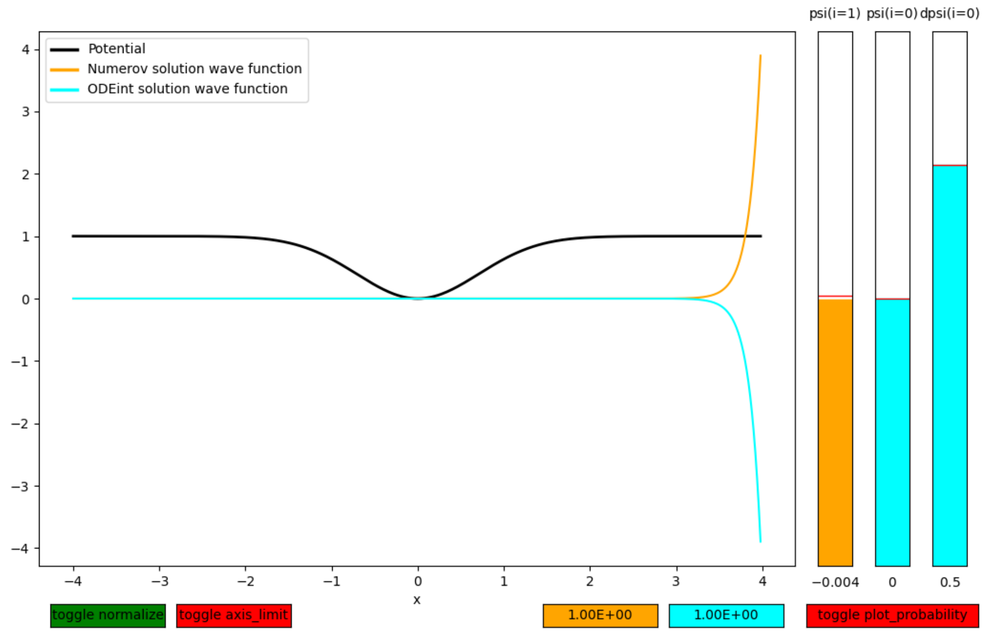

Fig. 3 E=0.2 M= 36

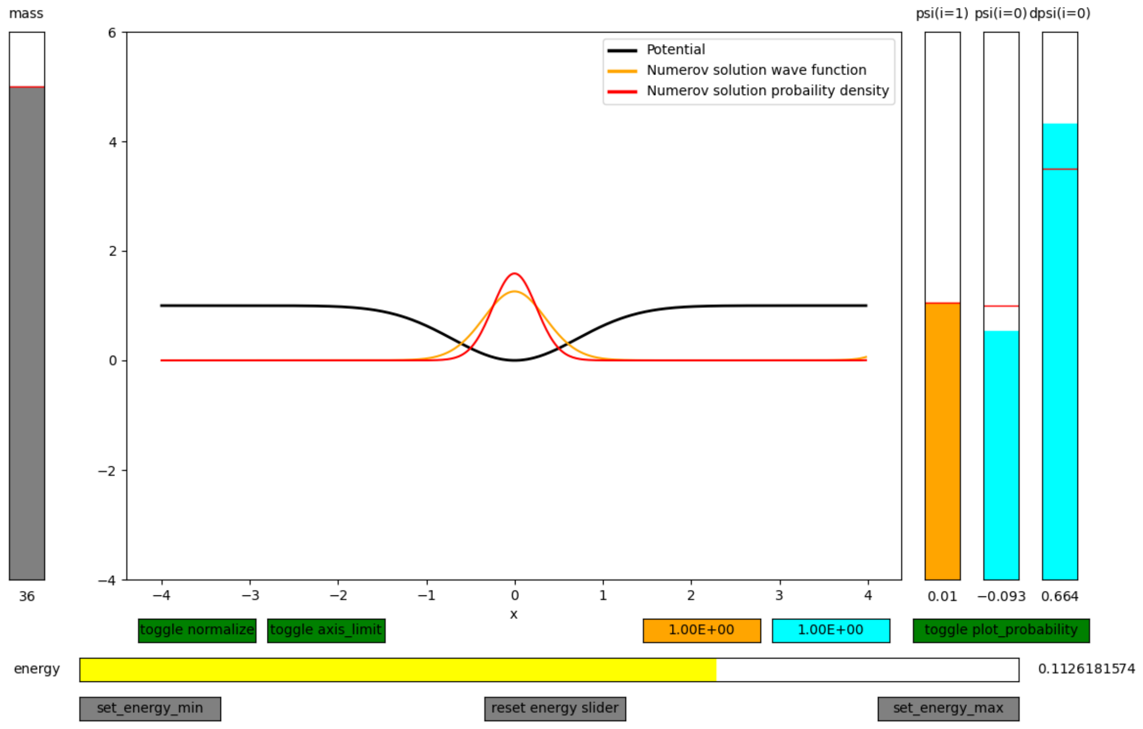

Fig.4 n=1 Numerov solution

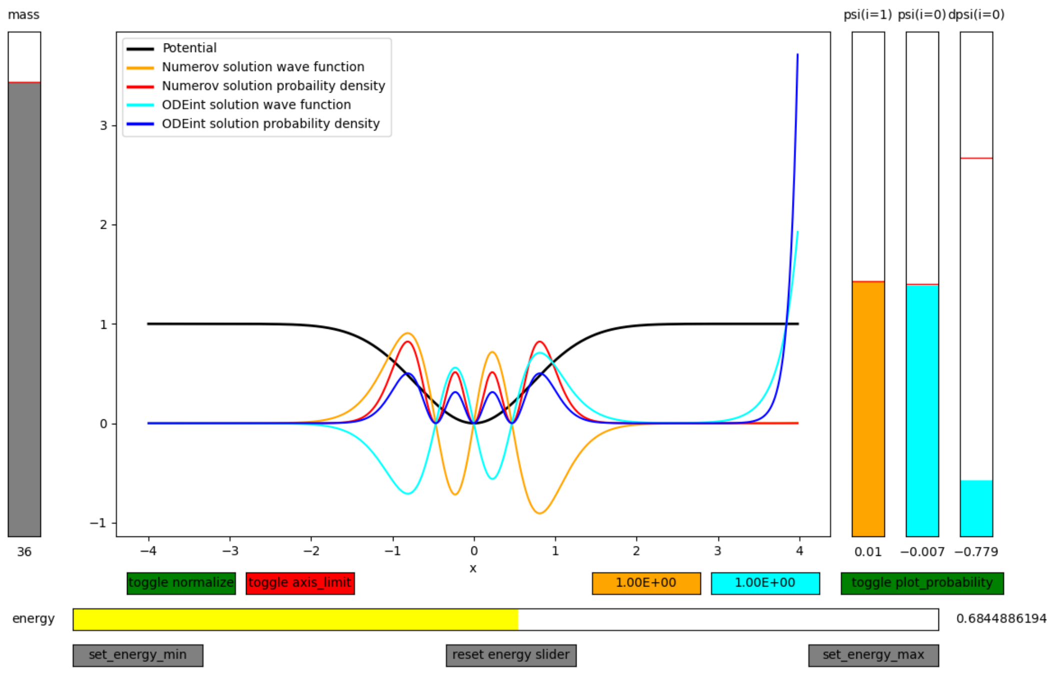

Fig. 5 n=4 both solutions shown

Further details

The starting values of energy, mass, potential function, domain, and initial values for different methods are defined near the top of the program and can easily be changed using an IDE or text editor if more precision is needed than matplotlib sliders provide. The potential function by default is a gaussian potential of depth 1. The selected initial values here give similar solutions and normalization, an algorithmic search through these values may need to be implemented for more accurate results.

The Numerov method numerically solves a differential equation of form

$$\frac{d^2y}{dx^2} + g(x)y(x) - s(x) = 0, \quad (2)$$

with continuous functions over a specific domain. The domain is discretized into evenly spaced points, and the taylor expansions of the current, next, and previous points are taken neglecting higher order terms. These can then be combined and solved for the next point of y(x) in the discrete domain using

$$y_{i+1} = \frac{(2y_i(1-\frac{5(\Delta x)^2}{12}g_i)-y_{i-1}(1+\frac{(\Delta x)^2}{12}g_{i-1})+\frac{(\Delta x)^2}{12}(S_{i+1}+10S_i+S_{i-1}))}{(1+\frac{(\Delta x)^2}{12}g_{i+1})}, \quad (3)$$

Where i and i±1 denote the current, previous, and next steps of the discrete domain and Δx is the step. Because the method uses current and past step of y in calculating the next the first 2 data points must be set manually.

The 1d time-independent schrodinger solution can be put in the form necessary for this method

$$\left[\frac{d^2\psi}{dx^2}\right] + 2m(E - U)[\psi] = 0, \quad (4)$$

where s(x) = 0 and g(x) = 2m(E-U). The first value of ψ is implicitly set as 0, under the assumption that the wave function approaches 0 far from a well of potential. The second value of ψ may be set using the first slider, this is the 'augment' parameter in the python function.

The second solution uses a system of first order differential equations to find a solution. The 1d time-independent schrodinger solution is represented as the system

$$\frac{d}{dx}(w_0) = w_1$$

$$\frac{d}{dx}(w_1) = -g(x)w_0$$

where w_n denotes $\frac{d^n\psi}{dx^n}$ and g(x) is the same as in the Numerov method. A python function is used to represent this system as a vector. Initial values are needed to find a solution again, these may be controlled with the rightmost 2 sliders. Taking the system, initial values, and an iterable domain the function scipy.odeint() returns a 2d array of x and ψ values.

Conclusion

The wave function diverging towards infinity so quickly at lower energy, and not near an allowed energy level, was surprising at first but not unexpected given more context. There are a number of points to address. These solutions diverging towards positive and negative infinity are not valid solutions to allowed energy states. The waves of unbounded solutions at greater energy values do not normalize properly as the normalizing algorithm assumes that the solution approaches 0 moving away from x=0. An algorithm searching for energy levels using these solutions could check for the sign of the solution changing, and search around each additional sign change for where the solution approaches 0 at both ends of the domain. Additionally an algorithm, and boundary conditions, to search through initial values will be needed for more accurate results. Near lower energy states the solution generally diverged much faster as the energy was changed, requiring far more iterations of reducing the range of the energy slider before the solution could be made to approach 0 at both ends of the domain. These states are more stable, perhaps the divergence and stability are related. Ultimately the interactivity of this program is not to the point I would like, searching through energy values may be enlightening but setting slider bounds may be confusing. It may be reasonably possible to find energy states when the mass or potential is changed and highlight them on the slider, as well as denote regularly spaced intervals, then dynamically change the scale to have greater precision around those energy levels.