Regularization scheme dependence of the behavior of wavefunction

May 9, 2024

Abstract

Regularization scheme dependence of the behavior of wavefunction in the presence of a delta potential is explored in this project by regularizing the delta function to the form $\delta = v_0 exp(-\lambda\rho^2)$. The project analyzes a non-relativistic system involving two identical particles interacting with a delta potential. The energy eigen function of this system, written in spherical coordinates , consists of a radial part $R(r)$ that satisfies a differential equation subject to a delta potential. Numerical simulations for this system were performed using the Numerov method, the main objectives included determining binding energies, verifying the constancy of the regularization parameter BE = 2.2 MeV, and analyzing the convergence point (ρ_c), where the numerical solution converges with the $exp(-\gamma * \epsilon_b\rho)$, which is the Analytical solution for $\rho \to \infty$ (universal behavior). This is done initially for regularization parameters $v_0 = 974.443Mev, \lambda = 8fm^{-2}$ and the analysis is repeated on various pairs of $\{v_0, \lambda\}$. The convergence point ρ_c is identified for each regularization. Results highlight the sensitivity of the wavefunction's behavior to regularization schemes, depicted through graphs, offering insight insights into the quantum mechanical systems with delta potentials.

1 Introduction

Numerov's method is a standard numerical technique that leverages on numerical integration of wavefunction and the condition that wavefunction is both continuous and differentiable at any point ρ_m.

Regularization of delta functions is a technique used to handle the issue of infinities that arise in the calculation of physical quantities due to the mathematical nature of delta functions. One way of regularization involves modifying the delta function in a way that removes these infinities while preserving the physical properties of the system. This is the approach that is adopted in this project.

Delta potentials and delta functions play an important role in Quantum physics and related areas serving as initial approximations for modeling systems such as quantum wells and nuclear interactions. This project considers a non-relativistic, 3-d system of 2-identical particles interacting with delta potential and explores regularisations of the form $\delta = v_0 exp(-\lambda\rho^2)$.

2 Defining the system

A non-relativistic, 3-d system of 2-identical particles of mass m_0 interacting with contact potential is considered. Eigenvalue equation for this system in relative coordinates $\bar{r} = \bar{r_1} - \bar{r_2}$ with reduced mass $m = \frac{m_0}{2}$ is given by equation(1).

$$\frac{-\hbar^2}{2m}\nabla^2\psi(\bar{r}) + V(\bar{r})\psi(\bar{r}) = E\psi(\bar{r}) \tag{1}$$

The eigen function when written in spherical coordinates takes the form $\psi(r) = \sum_{l,\mathfrak{m}} R(r)Y^l_{\mathfrak{m}}(\theta,\phi)$ with radial distance r, polar angle θ, and azimuthal angle ϕ and the Azimuthal Quantum Number l and Magnetic Quantum Number $\mathfrak{m}$. The radial part $R(r)$ satisfies the equation(2).

$$\frac{-\hbar^2}{2m}\left(\frac{1}{r}\frac{d}{dr}r^2\frac{d}{dr} - \frac{l(l+1)}{r^2}\right)R(r) + V(r)R(r) = ER(r) \tag{2}$$

substituting $R(r) = \frac{u(r)}{r}$

$$\frac{-\hbar^2}{2m}\frac{d^2}{dr^2}u(r) + \left(V(r) + \frac{l(l+1)}{r^2}\frac{\hbar^2}{2m}\right)u(r) = Eu(r) \tag{3}$$

For the convenience of analysis we introduce a dimensionless varaiable ρ, given by $\rho = (1/\alpha)r$ and rewrite the equation(3) as equation(4).

$$\frac{-\hbar^2}{2m\alpha^2}\frac{d^2}{dr^2}u(\rho) + \left(V(\rho) + \frac{l(l+1)}{\rho^2}\frac{\hbar^2}{2m\alpha^2}\right)u(r) = Eu(\rho) \tag{4}$$

for contact potential $V(\rho) = -\delta(\rho)$ and after regularizing the delta potential as $\delta(\rho) = v_0 exp(-\lambda\rho^2)$ the equation(4) further simplifies to equation(5).

$$\frac{d^2}{d\rho^2}u(\rho) + k(\rho)u(\rho) = 0 \tag{5}$$

where,

$$k(\rho) = \gamma\left(f(\rho) - \frac{l(l+1)}{\gamma\rho^2} - \epsilon\right)$$

$$\gamma = \frac{2m\alpha^2 v_0}{\hbar^2}$$

$$f(\rho) = exp(-\lambda\rho^2)$$

$$\epsilon = |\frac{E}{v_0}|$$

3 Numerov's method for a specific regularization

The analytical solution does not exist for the equation(5); therefore, we resort to a Numerical technique known as Numeov's method. We conduct the analysis specifically for a regularization of form $\delta(\rho) = v_0 exp(-\lambda\rho^2)$ particularly for the case of $v_0 = 974.443$ MeV and $\lambda = 8\ fm^{-2}$.

3.1 Methodology

The first step involved discretizing the space with $h = 10^{-5}$as step size and $b = 100(b \to \infty)$. This converts the DE in equation(5) to a recursion relation that allows us to find the value of the wavefunction at the current location/step based on its values at the previous two steps.

Then a trail reduced energy $\left(\frac{E_t}{v_0}\right)$ is chosen between E_max and E_min. For this trail reduced energy the numerical integration of wavefunction was carried out from $\rho \to 0$ to the point ρ_m. This is carried out by a function named calculate_forward_wavefunction in our implementation and this is referred to as forward integration. The values at ρ_m and ρ_{m-1}were noted (see Appendix A). Similar numerical integration (backward) is carried out from $\rho = b$ to $\rho = \rho_m$ and values of the wavefunction at ρ_m and ρ_{m-1} are noted (see Appendix B).

If E_trial is the reduced binding energy $\left(\epsilon_b = \left(\frac{BE}{v_0}\right)\right)$then it must satisfy the condition that the wavefunction for an energy eigenstate should be both continuous and differentiable at any point(ρ_m) implying that logarithmic derivatives from both forward and backward integration at ρ_m should produce the same result. The value of logarithmic derivate at ρ_m for forward wavefunction can be found by using equation(6).

$$logder_f(\rho_c) = \left(\frac{\frac{d}{d\rho}\left(\psi_f(\rho)\right)}{\psi_f(\rho)}\bigg|_{\rho=\rho_m}\right) \tag{6}$$

Similarly, logarithmic derivative for backward wavefunction at ρ_m can be found by using equation(7).

$$logder_b(\rho_c) = \left(\frac{\frac{d}{d\rho}\left(\psi_b(\rho)\right)}{\psi_b(\rho)}\bigg|_{\rho=\rho_m}\right) \tag{7}$$

The above condition implies that ϵ_b can be found by determining E_trail for which the equation(8) is satisfied. To achieve this, a standard method for finding roots called the bisection method is implemented in the interval [E_min , E_max] (see Appendix D). The bisection method involves iteratively narrowing down the search interval by bisecting it into two sub-intervals and then selecting the sub-interval where the signs of $logder_f(\rho_c) - logder_b(\rho_c)$ at the endpoints differ, indicating the presence of a root.

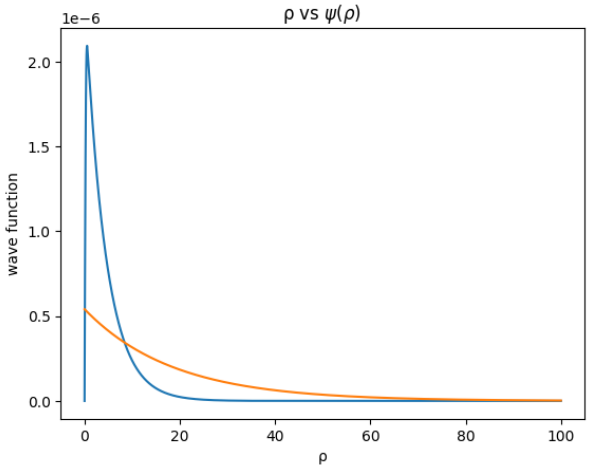
$$logder_f(\rho_c) - logder_b(\rho_c) \leq \epsilon_1; \epsilon_1 \to 0 \tag{8}$$

Finally, the point where the wavefunction tends to the universal form, termed as convergence point(ρ_c) in our implementation is found by imposing the criteria in equation(9), where $\epsilon_2 = 0.005$ in our implementation.

$$|(u(\rho_c)) - (exp(-\gamma * \epsilon_b \rho))| \leq \epsilon_2; \epsilon_2 \to 0 \tag{9}$$

3.2 Results

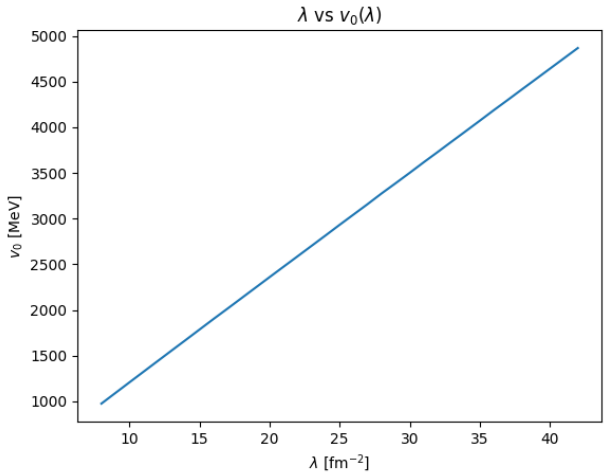
- The binding energy found : 2.23 MeV.
- The found point of convergence : 98.413



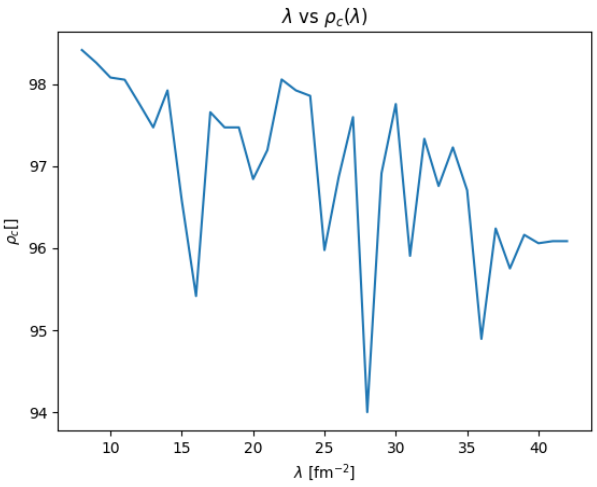
4 Finding the convergence points for different regularizations

4.1 The above analysis is Iteratively repeated for various pairs of $\{v_0[i], \lambda[i]\}$ and the point of convergence in each case is found.

- v_0=[974.443, 1090.781, 1206.874,1322.663, 1438.395, 1553.967, 1668.973, 1784.889, 1900.641, 2014.108, 2129.008, 2243.693, 2358.720, 2472.993, 2586.822, 2701.281, 2815.626, 2931.318, 3044.809, 3158.370, 3275.437, 3387.014, 3500.295, 3615.827, 3728.482, 3842.848, 3956.249, 4070.523, 4185.969, 4298.419, 4412.600, 4525.868, 4639.581, 4753.170, 4866.743] MeV
- λ = [8,9,10.....,43] fm^{-2}



4.2 Results



5 Conclusion

the project conducts a Numerical investigation into the behavior of wavefunction under different regularizations of form $\delta(\rho) = v_0 exp(-\lambda\rho^2)$ and presents the insights in the form of graphs.

6 Acknowledgements

I would like to express my sincere appreciation to my supervisor Dr. Johannes Kirscher whose support and expertise were invaluable to the success of this project.

7 Appendix

7.0.1 Appendix A

```
def calculate_wavefunction_forward(u_0, E_trail, ro_m, h, x, wavefunction_forward, lamda, gamma):
    # adding initial points in the wavefunction

    r_0 = 0
    wavefunction_current_1 = u_0
    wavefunction_current_2 = h * (l + 1) * (r_0**l) + wavefunction_current_1
    wavefunction_forward[0] = u_0
    wavefunction_forward[1] = wavefunction_current_2

    # important as x =2*h

    i = 2 * h     # error handling due to appending
    if len(wavefunction_forward) != 2 and len(x) != 2:
        wavefunction_forward = [u_0, wavefunction_current_2]
        x = [0, h]

    while i <= ro_m:

       v = calculate_reducedPotential(i, lamda)
        k = np.float128(gamma * (v - E_trail))
        # finding next wafunction
        wavefunction_current_3 = np.float128((-k * (h**2) + 2) * wavefunction_current_2
        - wavefunction_current_1)
```

```
        # updating for finding the next point wavefunction
        wavefunction_current_1 = wavefunction_current_2
        wavefunction_current_2 = (wavefunction_current_3)
        x.append(i)
        wavefunction_forward.append(wavefunction_current_2)
        if wavefunction_current_2 - wavefunction_current_1 < 0:
        i = i + h
return (x, wavefunction_forward, ro_m)
```

7.0.2 Appendix B

```
def claculate_wavefunction_backward(u_inf, E_trail, ro_m, b, h, y,
    wavefunction_backward, lamda, gamma ):
    u_inf = calculate_u_infinity(b, gamma, E_trail)
    wavefunction_current_1 = u_inf
    wavefunction_current_2 = np.float128(gamma * E_trail * h * np.exp(-gamma * E_trail * b) + u_inf)
    wavefunction_backward = [wavefunction_current_1, wavefunction_current_2]
    y = [b, b - h]
    j = b - 2 * h
    while j >= ro_m:
        v = calculate_reducedPotential(j, lamda)
        k = np.float128(gamma * (v - E_trail))
        wavefunction_current_3 = ((2 - (h**2) * k) * wavefunction_current_2
         -wavefunction_current_1)
        wavefunction_current_1 = wavefunction_current_2
        wavefunction_current_2 = wavefunction_current_3
        y.append(j)
        j = j - h
        wavefunction_backward.append(wavefunction_current_2)
return (y, wavefunction_backward)
```

7.0.3 Appendix C

```
def log_difference(E_trail,u_inf,b,h,y,wavefunction_backward,u_0,ro_m,
    x,wavefunction_forward,lamda,gamma):


    forward = calculate_wavefunction_forward(u_0, E_trail, ro_m, h, x,
        wavefunction_forward, lamda, gamma)
    ro_mid = forward[2]
    backward = claculate_wavefunction_backward(u_inf, E_trail, ro_mid, b, h, y,
       wavefunction_backward, lamda, gamma)
    #  finding logarithimic derivatives at matching point
    der_wavefunction_forward = (forward[1][-1] - forward[1][-2]) / h
    log_der_wavefunction_forward = der_wavefunction_forward / forward[1][-1]

    der_wavefunction_backward = (backward[1][-2] - backward[1][-1]) / h
    log_der_wavfunction_backward = der_wavefunction_backward / backward[1][-2]

return log_der_wavfunction_backward - log_der_wavefunction_forward
```

7.0.4 Appendix D

```
def log_difference(E_trail,u_inf,b,h,y,wavefunction_backward,u_0,ro_m,
    x,wavefunction_forward,lamda,gamma):


    forward = calculate_wavefunction_forward(u_0, E_trail, ro_m, h, x,
        wavefunction_forward, lamda, gamma)
    ro_mid = forward[2]
    backward = claculate_wavefunction_backward(u_inf, E_trail, ro_mid, b, h, y,
       wavefunction_backward, lamda, gamma)
    #  finding logarithimic derivatives at matching point
    der_wavefunction_forward = (forward[1][-1] - forward[1][-2]) / h
    log_der_wavefunction_forward = der_wavefunction_forward / forward[1][-1]

    der_wavefunction_backward = (backward[1][-2] - backward[1][-1]) / h
    log_der_wavfunction_backward = der_wavefunction_backward / backward[1][-2]

return log_der_wavfunction_backward - log_der_wavefunction_forward
```

7.0.5 Appendix E

```
 E_bound = bisect(log_difference,E_min,E_max,
       args=(u_inf,b,h,y,wavefunction_backward,
       u_0, ro_m,x,wavefunction_forward,lamda,gamma,),xtol=tolerance,)
```