

Python code and analysis

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 a = 3.0          # the half width of the well in angstroms
5 m = 1.0          # mass of electron in 1 me units
6 V0 = 10.0        # height/depth of well in eV
7 hbar = 1.0       # hbar
8 delta = np.sqrt(2.0*m*V0*(a**2.0)/(hbar**2.0)) # Defined in the hand written solution
9 eta = np.arange(0.001,2*np.pi,0.01) # Defined in the hand written solution
10
11 def f(f,eta):
12     """
13     This is the lhs in the transcendental equations.
14     """
15     return np.sqrt(((delta/eta)**2.0)-1.0)
16
17 def f1(eta):
18     """
19     The equation whose zeros are to be found to obtain the even solution.
20     """
21     return f(eta) - np.tan(eta)
22
23 def f2(eta):
24     """
25     The equation whose zeros are to be found to obtain the odd solution.
26     """
27     return f(eta) + (1.0/np.tan(eta))
28
29 def firstDerivative_O4(function ,x0,stepsize):
30     """
31     Returns first derivative of the function "function" at the point x0 by considering points , one and two
32     steps on either side of x0.
33     The Accuracy is of order (stepsize)^4.
34     """
35     return (function(x0 - 2.0*stepsize) - 8.0*function(x0 - stepsize) + 8.0*function(x0 + stepsize) -
36             function(x0 + 2.0*stepsize))/(12.0*stepsize)
37
38 def Newton_Raphson(f,x0,stepsize):
39     """
40     Returns the zero of f. x0 is the guess. stepsize will be used for determining the first derivative of
41     f.
42     """
43     fprime = firstDerivative_O4(f,x0,stepsize)
44     x1 = x0 - (f(x0)/fprime)
45     while((x0 - x1) >= stepsize**(8.0)):
46         x0 = x1
47         fprime = firstDerivative_O4(f,x0,stepsize)
48         x1 = x0 - f(x0)/fprime
49     return x1
50
51 def get_energy(eta):
52     """
53     Gets energy value for given eta value.
54     """
55     return (eta**2.0)*(hbar**2.0)/(2.0*m*(a**2.0))
56
57 ##### Main portion begins #####
58
59 even_guess = np.pi/4.0 # Guess for even solution
60 odd_guess = 0.999*np.pi # Guess for odd solution
61 even_sol = Newton_Raphson(f1 ,even_guess,10**(-3)) # Obtaining the solution using the Newton Raphson method
62 odd_sol = Newton_Raphson(f2 ,odd_guess,10**(-3)) # Obtaining the solution using the Newton Raphson method
63
64 #### Printing the solutions
65 print("eta for even sol : {etev:0.6f}".format(etev=even_sol))
66 print("eta for odd sol : {etod:0.6f}".format(etod=odd_sol))
67 print("Energy for even state is : {e2:0.6f}".format(e2=get_energy(even_sol)))
68 print("Energy for odd state is : {e1:0.6f}".format(e1=get_energy(odd_sol)))
```

The outputs are shown below :

Even solution (η)	Odd solution (η)	Even energy (eV)	Odd energy (eV)
1.460	2.922	0.118	0.474

