Python code and analysis

```python
import numpy as np
import matplotlib.pyplot as plt

a = 3.0              # the half width of the well in angstroms
m = 1.0              # mass of electron in 1 me units
V0 = 10.0            # height/depth of well in eV
hbar = 1.0           # hbar
delta = np.sqrt(2.0*m*V0*(a**2.0)/(hbar**2.0))    # Defined in the hand written solution
eta = np.arange(0.001,2*np.pi,0.01)  # Defined in the hand written solutiom

def f(f,eta):
    """
    This is the lhs in the transcendental equations.
    """
    return np.sqrt(((delta/eta)**2.0)-1.0)

def f1(eta):
    """
    The equation whose zeros are to be found to obtain the even solution.
    """
    return f(eta) - np.tan(eta)

def f2(eta):
    """
    The equation whose zeros are to be found to obtain the odd solution.
    """
    return f(eta) + (1.0/np.tan(eta))

def firstDerivative_O4(function,x0,stepsize):
    """
    Returns first derivative of the function "function" at the point x0 by considering points, one and two steps on either side of x0.
    The Accuracy is of order (stepsize)^4.
    """
    return (function(x0 - 2.0*stepsize) - 8.0*function(x0 - stepsize) + 8.0*function(x0 + stepsize) - function(x0 + 2.0*stepsize))/(12.0*stepsize)

def Newton_Raphson(f,x0,stepsize):
    """
    Returns the zero of f. x0 is the guess. stepsize will be used for determining the first derivative of f.
    """
    fprime = firstDerivative_O4(f,x0,stepsize)
    x1 = x0 - (f(x0)/fprime)
    while((x0 - x1) >= stepsize**(8.0)):
        x0 = x1
        fprime = firstDerivative_O4(f,x0,stepsize)
        x1 = x0 - f(x0)/fprime
    return x1

def get_energy(eta):
    """
    Gets energy value for given eta value.
    """
    return (eta**2.0)*(hbar**2.0)/(2.0*m*(a**2.0))

##### Main portion begins #####

even_guess = np.pi/4.0   # Guess for even solution
odd_guess = 0.999*np.pi  # Guess for odd solution
even_sol = Newton_Raphson(f1,even_guess,10**(-3)) # Obtaining the solution using the Newton Raphson method
odd_sol = Newton_Raphson(f2,odd_guess,10**(-3))   # Obtaining the solution using the Newton Raphson method

### Printing the solutions
print("eta for even sol : {etev:0.6f}".format(etev=even_sol))
print("eta for odd sol : {etod:0.6f}".format(etod=odd_sol))
print("Energy for even state is : {e2:0.6f}".format(e2=get_energy(even_sol)))
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

C++ code and analysis

```cpp
#include <iostream>
#include <math.h>
using namespace std;

double lhs(double delta, double eta){
        return sqrt(pow((delta/eta),2.0)-1.0);
}
double f1(double delta, double eta){
        return lhs(delta, eta) - tan(eta);
}
double f2(double delta, double eta){
        return lhs(delta, eta) + 1.0/tan(eta);
}
double df1(double delta, double x0, double stepsize){
        return (f1(delta, x0 - 2*stepsize) - 8*f1(delta, x0 - stepsize) + 8*f1(delta, x0 + stepsize) - f1(delta, x0 + 2*stepsize))/(12*stepsize) ;
}
double df2(double delta, double x0, double stepsize){
                return (f2(delta, x0 - 2*stepsize) - 8*f2(delta, x0 - stepsize) + 8*f2(delta, x0 + stepsize) - f2(delta, x0 + 2*stepsize))/(12*stepsize) ;
}
double NewtonRaphsonf1(double delta, double x0, double stepsize){
        double der = df1(delta, x0, stepsize);
        double x1 = x0 - (f1(delta, x0)/der);
        while ((x0 - x1) >= pow(stepsize, 8.0)){
                x0 = x1;
                der = df1(delta, x0, stepsize);
                x1 = x0 - (f1(delta, x0)/der);
        }
        return x1;
}
double NewtonRaphsonf2(double delta, double x0, double stepsize){
        double der = df2(delta, x0, stepsize);
        double x1 = x0 - (f2(delta, x0)/der);
        while ((x0 - x1) >= pow(stepsize, 8.0)){
                x0 = x1;
                der = df2(delta, x0, stepsize);
                x1 = x0 - (f2(delta, x0)/der);
                }
        return x1;
}
int main(){
        const double PI = 3.14159265358979323846;

        double a = 3.0;              // the half width of the well in angstroms
        double m = 1.0;              // mass of electron in 1 me units
        double V0 = 10.0;            // height/depth of well in eV
        double hbar = 1.0;           // hbar
        double delta = sqrt(2.0*m*V0*pow(a,2.0)/pow(hbar,2.0));

        double even_guess = PI/4.0 ;   // Guess for even solution
        double odd_guess = 0.999*PI ;  // Guess for odd solution
        double even_sol = NewtonRaphsonf1(delta, even_guess,pow(10,-3.0)) ; // Obtaining the solution using the Newton Raphson method
        double odd_sol = NewtonRaphsonf2(delta, odd_guess,pow(10,-3.0)) ;  // Obtaining the solution using the Newton Raphson method

        cout << "eta for even sol : " << even_sol << endl;
        cout << "eta for odd sol : " << odd_sol << endl;

        return 0;

}
```

The outputs are shown below (values for energy remain the same):

Even solution (η)	Odd solution (η)
1.460	2.922

Plotting the solution and the wavefunctions