Computational Microelectronics

Home Work 2

Professer : Sung-Min Hong

Student ID : 20172106

Student Name : Kim Hyo Seok (김 효 석)

1. Introduction

We solved the infinite poential square well problem here in the numerical method. As we increase the number of grids, we have found that the difference between the ground stage energies given by solving the numerical method and the analytic method is significantly reduced.

2. Simulation result and discussion

The number of grids was increased from 5 to 1000 with reasonable differences. The values of the ground state energy at N = 5, 50, and 500, respectively, are as follows.

N = 5, E = 0.0752 eV, N = 50, E = 0.0792 eV, N = 500, E = 0.0793 eV

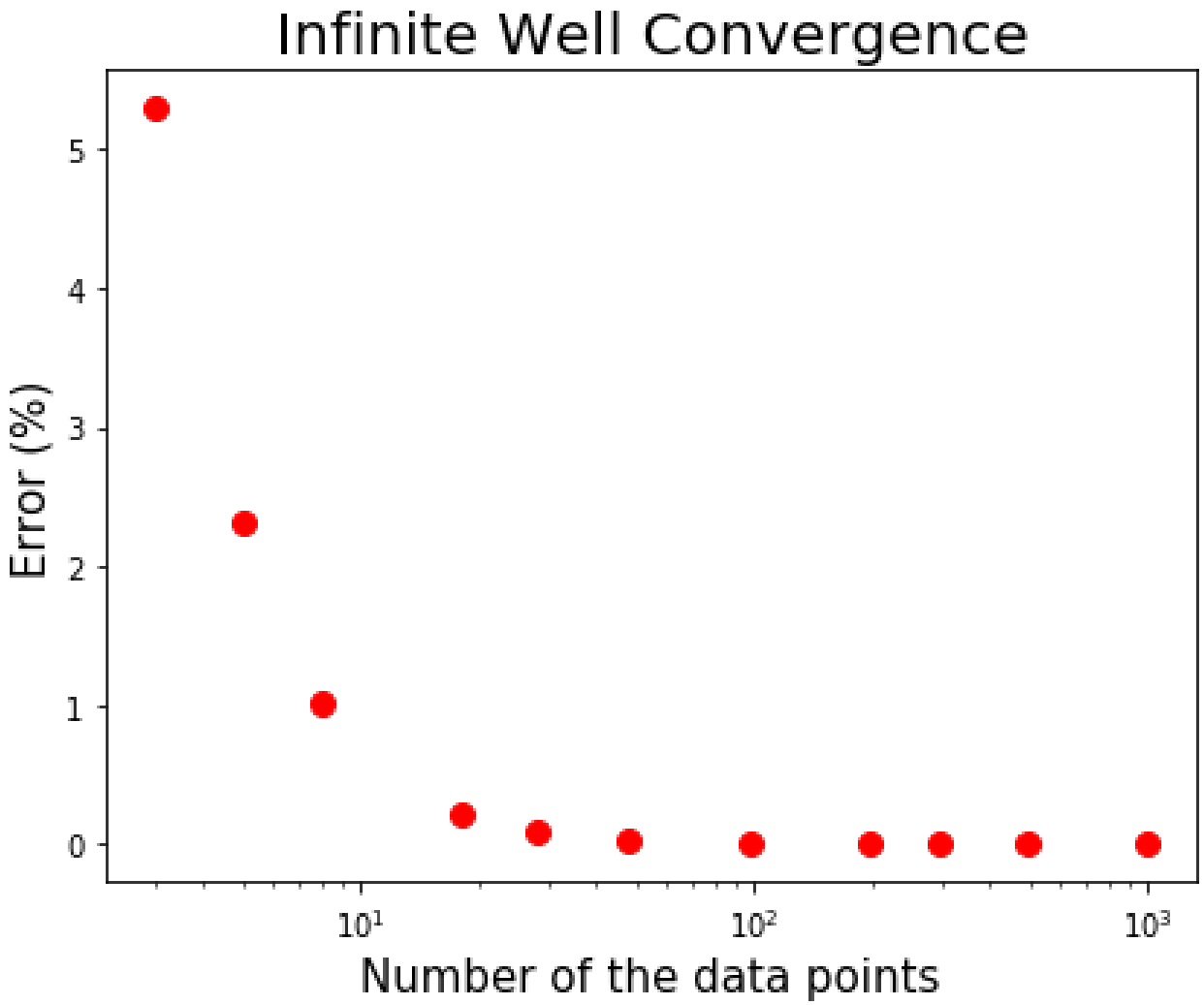

Figure 1. Error of the ground state energies with respect to the number of data points

The ground state energy when solving the analytic method is about 0.07917 eV. As you have seen Figure 1, the numerical solution approximates to the analytic soltion when the data points is increased.

3. Code (By python)

```
import numpy as np
import numpy.linalg as lin
import matplotlib.pyplot as plt
# Input variables #
a = 5.0 # [nm] , Total length
m = 0.19*9.10938356e-31 # electron's mass
dx = 0
E = 0.0
hbar = 1.0545819e-34 # hbar
N_number = [3, 5 ,8, 18, 28, 48, 98, 198, 298, 498, 998]
# Analytic solution for ground state #
E_anal = hbar*hbar*(np.pi*np.pi/a/a)/(2*m)
E_anal *= 1.0e+18*6.242e+18 # nm -> m & J -> eV
print("Analytic sol for E_ground = {} eV".format(E_anal))
E_history = []
E_Error_history = []
for N in N_number:
    fname = 'data_set%d.dat'%(N,) # name to save the data
    f = open(fname,'w')
    A = np.zeros(shape=(N,N)) # create zeros matrix [N by N]
    del_x = a/(N+1)
    # Boundary conditions
```

```
        A[0][0] = -2.0
        A[0][1] = 1.0
        A[N-1][N-1] = -2.0
        A[N-1][N-2] = 1.0
        # Laplacian components in A
        for i in range(N-2):
            A[i+1][i] = 1.0
            A[i+1][i+1] = -2.0
            A[i+1][i+2] = 1.0
        Eigen_value, Eigne_Vector = lin.eig(A) # D: eigenvalue, V: eigenvector
        Eigen_value_sort = np.sort(Eigen_value)
        # Energy calculation
        E_num = -hbar*hbar*Eigen_value_sort[N-1]/del_x/del_x/(2*m)
        E_num *= 1.0e+18 # nm -> m
        E_num *= 6.242e+18 # J -> eV
        # Energy & Error history
        E_history = np.append(E_history,E_num)
        E_Error_history = np.append(E_Error_history,(E_anal-E_num)/E_num*100)
        print("E_Error = {}%\n".format((E_anal-E_num)/E_num*100))
plt.figure(tight_layout=True, figsize=(6,5))
plt.semilogx(N_number,E_Error_history, 'ro', markersize=8)
plt.xlabel("Number of the grid", fontsize = 15)
plt.ylabel("Deviation for the ground energy (%)",fontsize = 15)
plt.title("Infinite Well Convergence",fontsize=20)
plt.savefig("./infinitewell_result.png")
plt.show()
```