HW-1

Xichen Li, EE521 - Group 5

Problem 11

Consider a potential (take $L = 8nm$)

$$U(x) = \begin{cases} U_0 = 400meV & x < 0\&x > L \\ 0 & 0 < x < L \end{cases}$$

a)By solving Schrödinger's equation, find the lowest eigenvalue (energy level) and the corresponding eigenfunction.

b)Plot the eigenfunction using a plotting package. Ensure that you present a zoomed-in plot of the eigenfunction for $x < 0$ and $x > L$.

c) Show that the eigenvalues form a discrete spectrum for energies less than U_0 and a continuous spectrum for higher energies. That is, for this Hamiltonian, there are both discrete quantum numbers with eigenenergies smaller than U_0 and a continuum of quantum numbers with eigenenergies larger than U_0.

Solution

A) Xichen Li: I am the only person in my group. I wrote the problem by myself. I started the problem by solving the eigenvalue and eigenfunction of the time-independent Schrödinger equation and then process the solved results.

a) Solve timd-indepednent Schrödinger equation:

For $0 < x < L$

$$[-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2}]\psi(x) = E\psi(x)$$
$$-\frac{\partial^2}{\partial x^2}\psi(x) = k_1^2\psi(x)$$

Where $k_1 = \frac{2mE}{\hbar^2}$, and then the general solution for $\psi(x)$ can be written as :

$$\psi(x) = A_1 sin(k_1 x) + B_1 cos(k_1 x)$$

For $x < 0$ and $x > L$

$$[-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + U_0]\psi(x) = E\psi(x)$$
$$-\frac{\partial^2}{\partial x^2}\psi(x) = k_2^2\psi(x)$$

Where $k_1 = \frac{2m(E-U_0)}{\hbar^2}$, and then the general solution for $\psi(x)$ can be written as :

$$\psi(x) = A_2 sin(k_2 x) + B_2 cos(k_2 x)$$

Boundary Condition 1: $\psi(0^+) = \psi(0^-)$

$$A_1 sin(0) + B_1 cos(0) = A_2 sin(0) + B_2 cos(0)$$

Then $B_1 = B_2$. Boundary Condition 2: $\psi(L^+) = \psi(L^-)$

$$A_1 sin(k_1 L) + B_1 cos(k_1 L) = A_2 sin(k_2 L) + B_2 cos(k_2 L)$$

It seems hard to determine all the parameters from the two boundary conditions, so the above time-independent Schrödinger equation is solved numerically which is shown below:

In [34]:
```
import numpy as np
%matplotlib notebook
import matplotlib.pyplot as plt
```

In [35]:
```
N = 200  # Grid size is 100
L = 8e-9

def V(x): # Return the potential energy given the position of each grid element
    if x<0 or x >L :
      result = 400e-3 * 1.6e-19
    else:
      result = 0
    return result

eta = 6.63e-34 #Reduced Plank constant in eV.s
m = 9.11e-31   #Assuming the particle mass is equal to a free electron
q = 1.6e-19
```

In [36]:
```
X = np.linspace(-2*L,2*L,num=N)
a = X[1] - X[0] # grid spacing

#print(X)

U=[]
U=np.array(U, dtype=float)

for x in X:
    U=np.append(U,V(x))

#print(U)

t = -eta**2 / (2 * m * a**2)
eps = -2*t + U

#print(t)
```

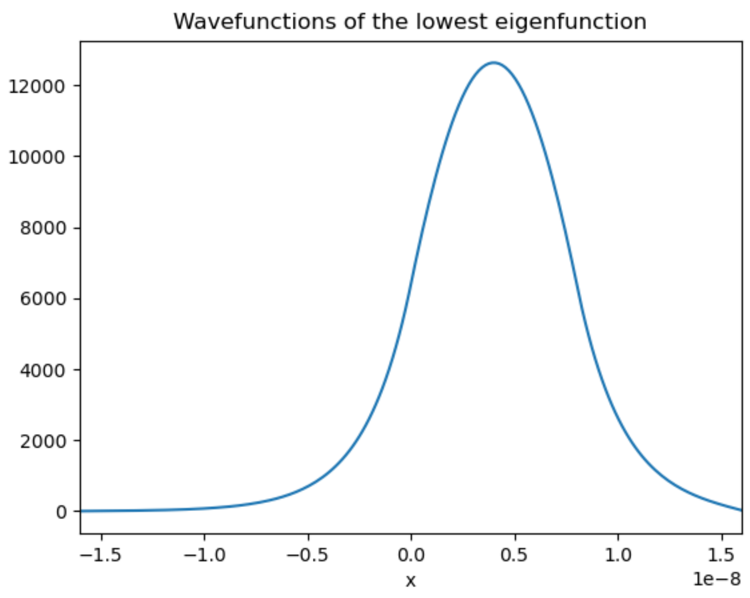
In [37]:
```
H = t*np.eye(N, k=-1) + eps*np.eye(N) + t*np.eye(N, k=1) # discretized hamiltonian
#plt.matshow(H)
```

In [38]:
```
vals, vecs = np.linalg.eig(H) # Solve for Eigen function and eigen values
#print(vals)
#print(vals.shape)
#print(vecs)
#print(vecs.shape)
order = np.argsort(vals)
vals, vecs = vals[order], vecs[:, order] # sort the eigen value and eigen functions
#print(vals)
#print(vecs)
vecs = vecs.T # Transpose
vecs /= np.sqrt(a) #Normalize
vals_eV=vals/1.6e-19
#print (vals_eV)
#print(vecs[0])
p = np.abs(vecs)**2
probs = a*p
#print(sum(probs[0]))
```

Now we have the eigenvalues (which correspond to the energies of each state) and the eigenvectors (which are the states, aka wavefunctions).

In [39]:
```
print ("The lowest energy is {:.4f}eV".format(vals_eV[0]))
print ("The corresponding eigenfunction is {}".format(vecs[0]))
print ("Verfiy the summed probablity of the frist eigenfunction: {} is equal to 1".format(sum(probs[0])))
#plot the first eigenfunction magnitude
plt.figure()
plt.plot(X,vecs[0],'-')
plt.title('Wavefunctions of the lowest eigenfunction')
plt.xlabel('x')
plt.xlim([-2*L,2*L])
#plt.ylim([-7000,-4000])
```

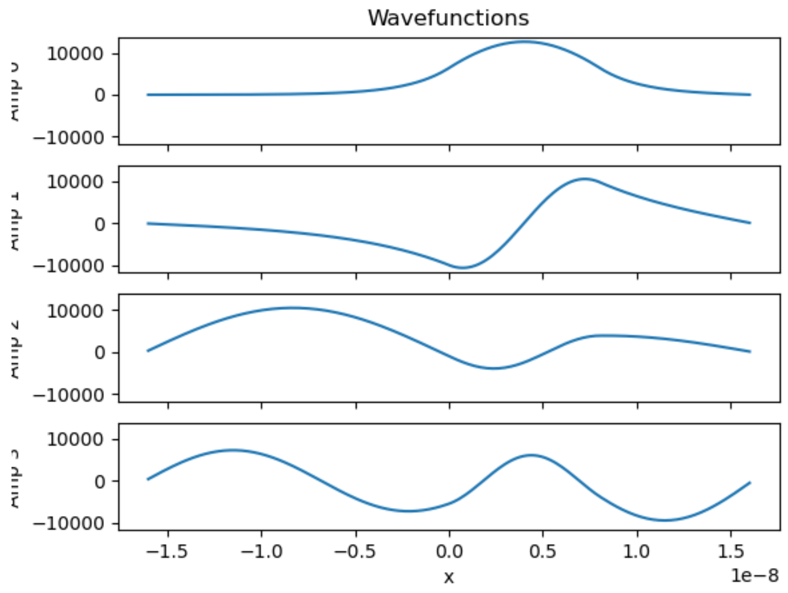
```
The lowest energy is 0.1015eV
The corresponding eigenfunction is [6.87458868e-01 1.37843719e+00 2.07647242e+00 2.78513818e+00
 3.50806247e+00 4.24894631e+00 5.01158267e+00 5.79987586e+00
 6.61786156e+00 7.46972746e+00 8.35983468e+00 9.29274014e+00
 1.02732198e+01 1.13062934e+01 1.23972496e+01 1.35516736e+01
 1.47754755e+01 1.60749206e+01 1.74566613e+01 1.89277716e+01
 2.04957827e+01 2.21687221e+01 2.39551545e+01 2.58642255e+01
 2.79057086e+01 3.00900551e+01 3.24284480e+01 3.49328585e+01
 3.76161080e+01 4.04919335e+01 4.35750578e+01 4.68812649e+01
 5.04274810e+01 5.42318610e+01 5.83138814e+01 6.26944402e+01
 6.73959636e+01 7.24425212e+01 7.78599488e+01 8.36759810e+01
 8.99203930e+01 9.66251532e+01 1.03824587e+02 1.11555551e+02
 1.19857624e+02 1.28773310e+02 1.38348252e+02 1.48631468e+02
 1.59675605e+02 1.71537202e+02 1.84276985e+02 1.97960175e+02
 2.12656824e+02 2.28442171e+02 2.45397030e+02 2.63608201e+02
 2.83168915e+02 3.04179316e+02 3.26746964e+02 3.50987396e+02
 3.77024711e+02 4.04992207e+02 4.35033063e+02 4.67301075e+02
 5.01961438e+02 5.39191596e+02 5.79182150e+02 6.22137831e+02
 6.68278552e+02 7.17840530e+02 7.71077497e+02 8.28262002e+02
 8.89686801e+02 9.55666358e+02 1.02653846e+03 1.10266593e+03
 1.18443850e+03 1.27227482e+03 1.36662456e+03 1.46797074e+03
 1.57683221e+03 1.69376629e+03 1.81937161e+03 1.95429123e+03
 2.09921585e+03 2.25488742e+03 2.42210291e+03 2.60171837e+03
 2.79465334e+03 3.00189557e+03 3.22450602e+03 3.46362436e+03
 3.72047475e+03 3.99637213e+03 4.29272898e+03 4.61106248e+03
 4.95300235e+03 5.32029916e+03 5.71483328e+03 6.13862453e+03
 6.59384251e+03 7.03758699e+03 7.46908582e+03 7.88758820e+03
 8.29236590e+03 8.68271460e+03 9.05795508e+03 9.41743442e+03
 9.76052710e+03 1.00866361e+04 1.03951941e+04 1.06856640e+04
 1.09575406e+04 1.12103506e+04 1.14436542e+04 1.16570455e+04
 1.18501532e+04 1.20226412e+04 1.21742093e+04 1.23045940e+04
 1.24135682e+04 1.25009424e+04 1.25665645e+04 1.26103204e+04
 1.26321339e+04 1.26319670e+04 1.26098202e+04 1.25657317e+04
 1.24997785e+04 1.24120753e+04 1.23027746e+04 1.21720667e+04
 1.20201789e+04 1.18473757e+04 1.16539576e+04 1.14402612e+04
 1.12066584e+04 1.09535556e+04 1.06813932e+04 1.03906449e+04
 1.00818165e+04 9.75544537e+03 9.41209945e+03 9.05237616e+03
 8.67690143e+03 8.28632859e+03 7.88133726e+03 7.46263214e+03
 7.03094178e+03 6.58701735e+03 6.13163128e+03 5.70763615e+03
 5.31286130e+03 4.94528569e+03 4.60302750e+03 4.28433455e+03
 3.98757528e+03 3.71123043e+03 3.45388526e+03 3.21422228e+03
 2.99101453e+03 2.78311931e+03 2.58947228e+03 2.40908208e+03
 2.24102519e+03 2.08444125e+03 1.93852862e+03 1.80254031e+03
 1.67578011e+03 1.55759908e+03 1.44739219e+03 1.34459524e+03
 1.24868195e+03 1.15916130e+03 1.07557499e+03 9.97495085e+02
 9.24521868e+02 8.56281747e+02 7.92425367e+02 7.32625815e+02
 6.76576947e+02 6.23991819e+02 5.74601223e+02 5.28152301e+02
 4.84407259e+02 4.43142144e+02 4.04145699e+02 3.67218280e+02
 3.32170839e+02 2.98823948e+02 2.67006889e+02 2.36556774e+02
 2.07317713e+02 1.79140016e+02 1.51879428e+02 1.25396389e+02
 9.95553164e+01 7.42239185e+01 4.92725105e+01 2.45733535e+01]
Verfiy the summed probablity of the frist eigenfunction: 0.9999999999999999 is equal to 1
```



```
Out[39]: (-1.6e-08, 1.6e-08)
```

b) Plot the eigenfunction of the lowest four eigenfunctions.

```
In [40]: fig, axes = plt.subplots(4, sharex=True, sharey=True)
         plt.sca(axes[0])
         plt.title('Wavefunctions')
         for i, (ax, Amp, E) in enumerate(zip(axes, vecs, vals_eV)):
             plt.sca(ax)
             plt.ylabel('Amp {}'.format(i))
             plt.plot(X, Amp)
             #plt.fill_between(X, psi, alpha=0.3)
             print('E_{} = {:.4f}'.format(i, E))
         plt.xlabel('x')
```

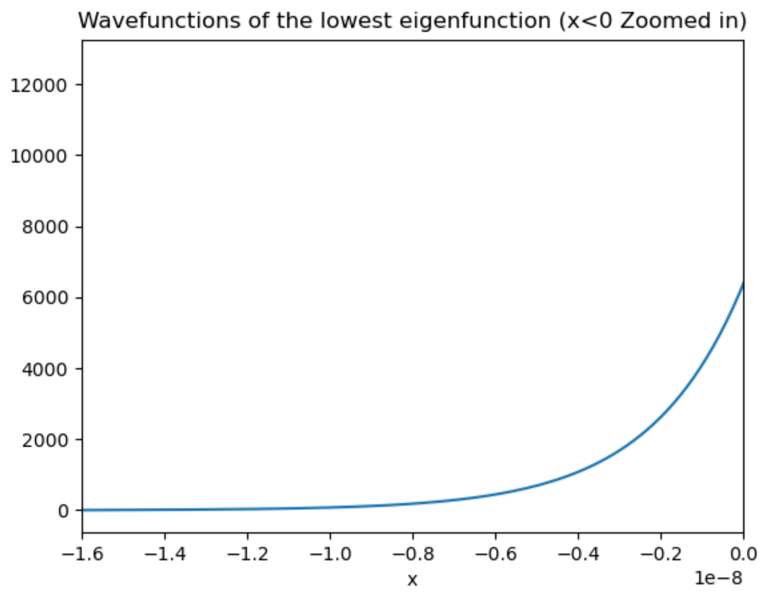


```
E_0 = 0.1015
E_1 = 0.3522
E_2 = 0.4604
E_3 = 0.5701
```

```
Out[40]: Text(0.5, 0, 'x')
```

A zoomed-in plot of the lowest eigenfunction for $x < 0$ is shown below:

In [41]:
```python
plt.figure()
plt.plot(X,vecs[0],'-')
plt.title('Wavefunctions of the lowest eigenfunction (x<0 Zoomed in)')
plt.xlabel('x')
plt.xlim([-2*L,0])
#plt.ylim([-7000,-4000])
```

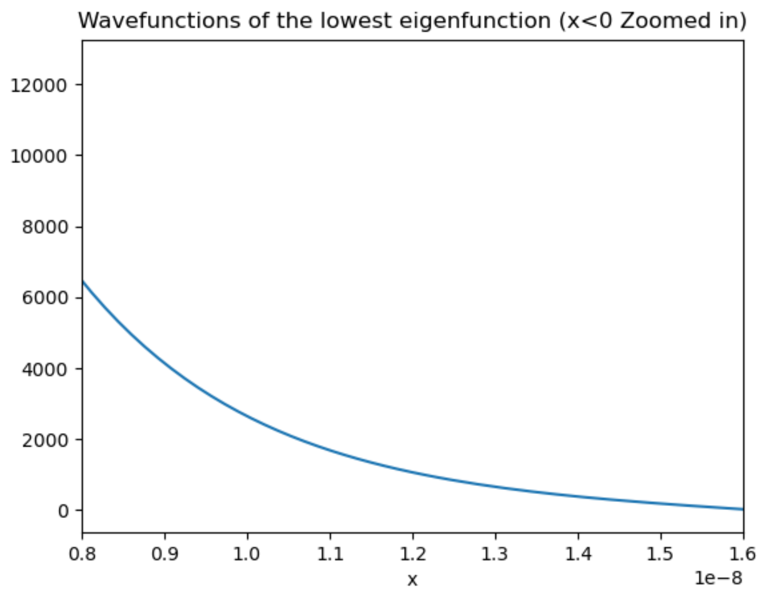


Out[41]:
```
(-1.6e-08, 0.0)
```

As you can see from the plotted figure above, the wavefunction is approaching zero at the boundary ($x = -L$).

A zoomed-in plot of the lowest eigenfunction for $x > L$ is shown below:

In [42]:
```python
plt.figure()
plt.plot(X,vecs[0],'-')
plt.title('Wavefunctions of the lowest eigenfunction (x<0 Zoomed in)')
plt.xlabel('x')
plt.xlim([L,2*L])
#plt.ylim([-7000,-4000])
```



Out[42]:
```
(8e-09, 1.6e-08)
```

As you can see from the plotted figure above, the wavefunction is approaching zero at the boundary ($x = 2L$).

c)Show the eigenvalues form a discrete spectrum for energies less than U_0 and a continuous spectrum for higher energies

Compare the Eigen values for different N

In [43]:
```
N1 = 20  # Grid size is 100
X1 = np.linspace(-2*L,2*L,num=N1)
a1 = X1[1] - X1[0] # grid spacing

U1=[]
U1=np.array(U1, dtype=float)

for x in X1:
    U1=np.append(U1,V(x))

t1 = -eta**2 / (2 * m * a1**2)
eps1 = -2*t1 + U1

H1 = t1*np.eye(N1, k=-1) + eps1*np.eye(N1) + t1*np.eye(N1, k=1) # discretized hamiltonian
vals1, vecs1 = np.linalg.eig(H1) # Solve for Eigen function and eigen values
order1 = np.argsort(vals1)
vals1, vecs1 = vals1[order1], vecs1[:, order1] # sort the eigen value and eigen functions
vecs1 = vecs1.T # Transpose
vecs1 /= np.sqrt(a1) #Normalize
vals_eV1=vals1/1.6e-19
print ("The eigenvalues are {}eV for N={}".format(vals_eV1,N1))
```

```
The eigenvalues are [0.09025355 0.32151214 0.44664935 0.52945043 0.59338314 0.71748984
 0.84142187 0.99621271 1.11655526 1.29616024 1.44482688 1.61001831
 1.73005353 1.89418252 2.01275554 2.12797139 2.21160146 2.37923125
 2.41469082 2.48864827]eV for N=20
```

In [44]:
```
N2 = 200  # Grid size is 200
X2 = np.linspace(-2*L,2*L,num=N2)
a2 = X2[1] - X2[0] # grid spacing

U2=[]
U2=np.array(U2, dtype=float)

for x in X2:
    U2=np.append(U2,V(x))

t2 = -eta**2 / (2 * m * a2**2)
eps2 = -2*t2 + U2

H2 = t2*np.eye(N2, k=-1) + eps2*np.eye(N2) + t2*np.eye(N2, k=1) # discretized hamiltonian
vals2, vecs2 = np.linalg.eig(H2) # Solve for Eigen function and eigen values
order2 = np.argsort(vals2)
vals2, vecs2 = vals2[order2], vecs2[:, order2] # sort the eigen value and eigen functions
vecs2 = vecs2.T # Transpose
vecs2 /= np.sqrt(a2) #Normalize
vals_eV2=vals2/1.6e-19
print ("The eigenvalues are {}eV for N={}".format(vals_eV2,N2))
```

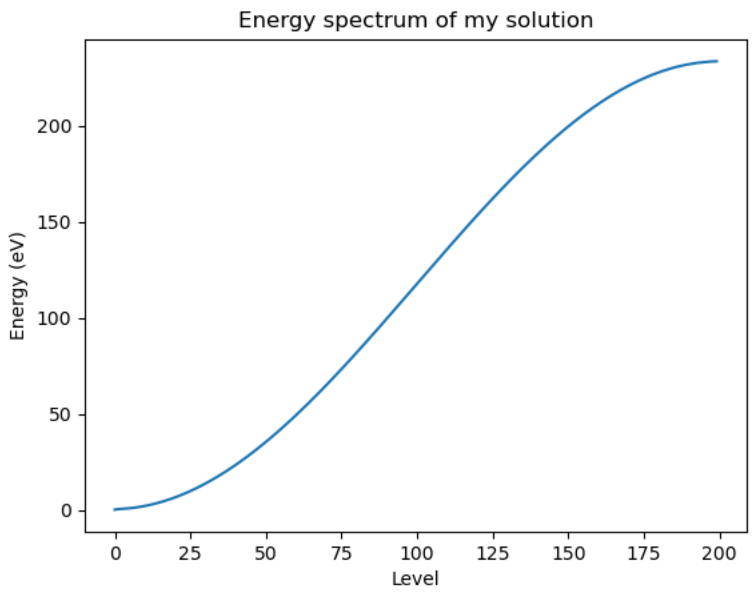
```
The eigenvalues are [1.01466487e-01 3.52182635e-01 4.60350405e-01 5.70121561e-01
 6.59051226e-01 8.28397020e-01 1.00496285e+00 1.22896778e+00
 1.44651647e+00 1.72990656e+00 2.02457791e+00 2.35491237e+00
 2.69450166e+00 3.08599699e+00 3.49461502e+00 3.93358660e+00
 4.38904754e+00 4.88865389e+00 5.40830249e+00 5.95568488e+00
 6.52303981e+00 7.12980160e+00 7.75794676e+00 8.41224002e+00
 9.08819986e+00 9.80030494e+00 1.05342821e+01 1.12932308e+01
 1.20745749e+01 1.28895706e+01 1.37264173e+01 1.45872090e+01
 1.54705429e+01 1.63854526e+01 1.73218544e+01 1.82812076e+01
 1.92628641e+01 2.02742521e+01 2.13065291e+01 2.23607405e+01
 2.34367397e+01 2.45407522e+01 2.56648614e+01 2.68098367e+01
 2.79758738e+01 2.91682691e+01 3.03798143e+01 3.16110931e+01
 3.28625390e+01 3.41387131e+01 3.54329596e+01 3.67457374e+01
 3.80776475e+01 3.94326575e+01 4.08045488e+01 4.21936984e+01
 4.36008266e+01 4.50294133e+01 4.64735901e+01 4.79336833e+01
 4.94104985e+01 5.09071094e+01 5.24179303e+01 5.39432597e+01
 5.54839651e+01 5.70427776e+01 5.86143406e+01 6.01989431e+01
 6.17974971e+01 6.34124423e+01 6.50386079e+01 6.66762878e+01
 6.83264267e+01 6.99912140e+01 7.16656289e+01 7.33499827e+01
 7.50452440e+01 7.67533862e+01 7.84695084e+01 8.01939498e+01
 8.19276971e+01 8.36725360e+01 8.54236601e+01 8.71814461e+01
 8.89468942e+01 9.07216275e+01 9.25009108e+01 9.42851677e+01
 9.60754094e+01 9.78731169e+01 9.96736063e+01 1.01477357e+02
 1.03285389e+02 1.05099061e+02 1.06913720e+02 1.08729911e+02
 1.10548662e+02 1.12371224e+02 1.14192960e+02 1.16014489e+02
 1.17836847e+02 1.19661192e+02 1.21482885e+02 1.23302629e+02
 1.25121467e+02 1.26940480e+02 1.28755009e+02 1.30565852e+02
 1.32374058e+02 1.34180646e+02 1.35980916e+02 1.37775777e+02
 1.39566282e+02 1.41353399e+02 1.43132373e+02 1.44904231e+02
 1.46670035e+02 1.48430714e+02 1.50181434e+02 1.51923361e+02
 1.53657560e+02 1.55384937e+02 1.57100556e+02 1.58805739e+02
 1.60501554e+02 1.62188896e+02 1.63862703e+02 1.65524472e+02
 1.67175273e+02 1.68816006e+02 1.70441451e+02 1.72053308e+02
 1.73652640e+02 1.75240374e+02 1.76811094e+02 1.78366733e+02
 1.79908343e+02 1.81436898e+02 1.82946742e+02 1.84440079e+02
 1.85917938e+02 1.87381367e+02 1.88824420e+02 1.90249612e+02
 1.91657939e+02 1.93050558e+02 1.94421159e+02 1.95772631e+02
 1.97105915e+02 1.98422320e+02 1.99715088e+02 2.00987552e+02
 2.02240572e+02 2.03475669e+02 2.04685518e+02 2.05873997e+02
 2.07041839e+02 2.08190863e+02 2.09313022e+02 2.10412868e+02
 2.11490943e+02 2.12549485e+02 2.13579509e+02 2.14586425e+02
 2.15570481e+02 2.16534516e+02 2.17468296e+02 2.18378352e+02
 2.19264478e+02 2.20130399e+02 2.20964165e+02 2.21773821e+02
 2.22558444e+02 2.23323107e+02 2.24053416e+02 2.24759548e+02
 2.25439399e+02 2.26100203e+02 2.26723900e+02 2.27323837e+02
 2.27895861e+02 2.28450900e+02 2.28965013e+02 2.29456610e+02
 2.29917694e+02 2.30366118e+02 2.30767588e+02 2.31149366e+02
 2.31495460e+02 2.31838458e+02 2.32123353e+02 2.32394489e+02
 2.32617367e+02 2.32859883e+02 2.33021383e+02 2.33166058e+02
 2.33271540e+02 2.33473510e+02 2.33508970e+02 2.33606534e+02]eV for N=200
```

In [45]:
```
N3 = 2000  # Grid size is 200
X3 = np.linspace(-2*L,2*L,num=N3)
a3 = X3[1] - X3[0] # grid spacing

U3=[]
U3=np.array(U3, dtype=float)

for x in X3:
    U3=np.append(U3,V(x))

t3 = -eta**2 / (2 * m * a3**2)
eps3 = -2*t3 + U3

H3 = t3*np.eye(N3, k=-1) + eps3*np.eye(N3) + t3*np.eye(N3, k=1) # discretized hamiltonian
vals3, vecs3 = np.linalg.eig(H3) # Solve for Eigen function and eigen values
order3 = np.argsort(vals3)
vals3, vecs3 = vals3[order3], vecs3[:, order3] # sort the eigen value and eigen functions
vecs3 = vecs3.T # Transpose
vecs3 /= np.sqrt(a3) #Normalize
vals_eV3=vals3/1.6e-19
print ("The eigenvalues are {}eV for N={}".format(vals_eV3,N3))
```

```
The eigenvalues are [1.02115100e-01 3.53873765e-01 4.61724405e-01 ... 2.35368513e+04
 2.35368866e+04 2.35369865e+04]eV for N=2000
```

By comparing the eigenvalues for $N1 = 20$, $N2 = 200$, and $N3 = 2000$ cases, we can find for eigenvalue is smaller than $U_0 = 0.4eV$, the eigenvalues have two discrete values which are $E_0 = 0.102eV$ and $E_1 = 0.354eV$. When eigenvalue is larger than $U_0 = 0.4eV$, the eigenvalues have continuous energy spectrum. The energy spectrum for $N2 = 100$ can alsobe found in the figure below.

In [46]:
```
plt.figure()
plt.plot(vals_eV,'-')
plt.title('Energy spectrum of my solution')
plt.xlabel('Level')
plt.ylabel('Energy (eV)')
```



Out[46]:
```
Text(0, 0.5, 'Energy (eV)')
```

In [ ]: