Run on Google Colab View on Github

Imports

In [2]:
```python
import numpy as np
import matplotlib.pyplot as plt
from sympy import *
```

Defining a Function to get the energy corresponding to a given K.

In [3]:
```python
def give_energy(K, h=1, m=1):
    """
    This function calculates the energy of a particle in a potential
    field.
    """
    return (h**2*K**2)/(2*m)

give_energy = np.vectorize(give_energy)
```

Solution

Defining the Matrix

The final matrix, that we get is:

$$M = \begin{bmatrix} \frac{\hbar^2K^2}{2m} - E & V_{\frac{-2\pi}{a}} & V_{\frac{2\pi}{a}} \\ V_{\frac{2\pi}{a}} & \frac{\hbar^2(K+\frac{2\pi}{a})^2}{2m} - E & V_{\frac{4\pi}{a}} \\ V_{\frac{-2\pi}{a}} & V_{\frac{-4\pi}{a}} & \frac{\hbar^2(K-\frac{2\pi}{a})^2}{2m} - E \end{bmatrix}$$

We need to diagonalize this. This can either be down by setting the determinant of E to zero:

$$det(M) = 0$$

Setting the determinant to zero will give us a third order eqaution in E, solving which gives three distinct roots which are the required energies.

Or alternatively, the same can also be done by finding the eigenvalues of the matrix M. But we'll use the first method.

In [4]:
```python
#Setting the variables
E = Symbol('E')
Vp = Symbol('V', complex=True)
Vm = Vp.conjugate()
Vpp = Symbol('Vp', complex=True)
Vmm = Vpp.conjugate()
E0 = Symbol('E0')
Ep = Symbol('Ep')
Em = Symbol('Em')
```

In [5]:
```python
#Creating the matrix
mat = Matrix([[E0-E, Vm, Vp], [Vm, Ep-E, Vpp], [Vp, Vmm, Em-E]])
mat
```

Out[5]:

$$\begin{bmatrix} -E + E_0 & \overline{V} & V \\ \overline{V} & -E + Ep & Vp \\ V & \overline{Vp} & -E + Em \end{bmatrix}$$

Here for simplicity, We have used:

$$E_0 = \frac{\hbar^2K^2}{2a^2}$$

$$E_p = \frac{\hbar^2(K + \frac{2\pi}{a})^2}{2a^2}$$

$$E_m = \frac{\hbar^2(K - \frac{2\pi}{a})^2}{2a^2}$$

$$V = V_{\frac{-2\pi}{a}}$$

$$\bar{V} = V_{\frac{2\pi}{a}}$$

$$V_p = V_{\frac{4\pi}{a}}$$

$$\bar{V_p} = V_{\frac{-4\pi}{a}}$$

Solving for E

If we solve the equation symbolically for E, we'll get a HUGE expression for E. To make the expression smaller, we'll set the values of V and V_P to be equal to 1.

In [6]:
```python
#Getting the determinant
eq = mat.det()
#Substituting Vp = Vpp =1 to simplify the equation
eq = eq.subs(Vp, 1)
eq = eq.subs(Vpp, 1)
eq
```

Out[6]:

$$-E^3 + E^2E_0 + E^2Em + E^2Ep - EE_0Em - EE_0Ep - EEmEp + 3E + E_0EmEp - E_0 - Em - Ep + 2$$

Now, we need to solve for E in terms of E_0, E_M and E_P. `sympy` has a function `solve` which solves the equation symbolically.

In [7]:
```python
Es = solve(eq, E)
len(Es)
```

Out[7]: 3

Since the equation was third order, we should get three different solutions of E. We can see from the output of the above cell, `len(Es)` is indeed 3. Let's see what these three solutions are:

First Solution

In [8]:
```python
E1 = Es[0]
E1
```

Out[8]:

$$\frac{E_0}{3} + \frac{Em}{3} + \frac{Ep}{3} - \frac{-3E_0Em - 3E_0Ep - 3EmEp + (-E_0 - Em - Ep)^2 + 9}{3\sqrt[3]{-\frac{27E_0EmEp}{2} + \frac{27E_0}{2} + \frac{27Em}{2} + \frac{27Ep}{2} + \frac{\sqrt{-4\left(-3E_0Em-3E_0Ep-3EmEp+(-E_0-Em-Ep)^2+9\right)^3 + \left(-27E_0EmEp+27E_0+27Em+27Ep-(-9E_0-9Em-9Ep)(E_0Em+E_0Ep+EmEp-3)+2(-E_0-Em-Ep)^3-54\right)^2}}{2} - \frac{(-9E_0-9Em-9Ep)(E_0Em+E_0Ep+EmEp-3)}{2} + (-E_0 - Em - Ep)^3 - 27}} - \frac{\sqrt[3]{-\frac{27E_0EmEp}{2} + \frac{27E_0}{2} + \frac{27Em}{2} + \frac{27Ep}{2} + \frac{\sqrt{-4\left(-3E_0Em-3E_0Ep-3EmEp+(-E_0-Em-Ep)^2+9\right)^3 + \left(-27E_0EmEp+27E_0+27Em+27Ep-(-9E_0-9Em-9Ep)(E_0Em+E_0Ep+EmEp-3)+2(-E_0-Em-Ep)^3-54\right)^2}}{2} - \frac{(-9E_0-9Em-9Ep)(E_0Em+E_0Ep+EmEp-3)}{2} + (-E_0 - Em - Ep)^3 - 27}}{3}$$

Second Solution

In [9]:
```python
E2 = Es[1]
E2
```

Out[9]:

$$\frac{E_0}{3} + \frac{Em}{3} + \frac{Ep}{3} - \frac{-3E_0Em - 3E_0Ep - 3EmEp + (-E_0 - Em - Ep)^2 + 9}{3\left(-\frac{1}{2} - \frac{\sqrt{3}i}{2}\right)\sqrt[3]{-\frac{27E_0EmEp}{2} + \frac{27E_0}{2} + \frac{27Em}{2} + \frac{27Ep}{2} + \frac{\sqrt{-4\left(-3E_0Em-3E_0Ep-3EmEp+(-E_0-Em-Ep)^2+9\right)^3 + \left(-27E_0EmEp+27E_0+27Em+27Ep-(-9E_0-9Em-9Ep)(E_0Em+E_0Ep+EmEp-3)+2(-E_0-Em-Ep)^3-54\right)^2}}{2} - \frac{(-9E_0-9Em-9Ep)(E_0Em+E_0Ep+EmEp-3)}{2} + (-E_0 - Em - Ep)^3 - 27}} - \frac{\left(-\frac{1}{2} - \frac{\sqrt{3}i}{2}\right)\sqrt[3]{-\frac{27E_0EmEp}{2} + \frac{27E_0}{2} + \frac{27Em}{2} + \frac{27Ep}{2} + \frac{\sqrt{-4\left(-3E_0Em-3E_0Ep-3EmEp+(-E_0-Em-Ep)^2+9\right)^3 + \left(-27E_0EmEp+27E_0+27Em+27Ep-(-9E_0-9Em-9Ep)(E_0Em+E_0Ep+EmEp-3)+2(-E_0-Em-Ep)^3-54\right)^2}}{2} - \frac{(-9E_0-9Em-9Ep)(E_0Em+E_0Ep+EmEp-3)}{2} + (-E_0 - Em - Ep)^3 - 27}}{3}$$

Third Solution

In [10]:
```python
E3  = Es[2]
E3
```

Out[10]:

$$\frac{E_0}{3} + \frac{Em}{3} + \frac{Ep}{3} - \frac{-3E_0Em - 3E_0Ep - 3EmEp + (-E_0 - Em - Ep)^2 + 9}{3\left(-\frac{1}{2} + \frac{\sqrt{3}i}{2}\right)\sqrt[3]{-\frac{27E_0EmEp}{2} + \frac{27E_0}{2} + \frac{27Em}{2} + \frac{27Ep}{2} + \frac{\sqrt{-4\left(-3E_0Em-3E_0Ep-3EmEp+(-E_0-Em-Ep)^2+9\right)^3 + \left(-27E_0EmEp+27E_0+27Em+27Ep-(-9E_0-9Em-9Ep)(E_0Em+E_0Ep+EmEp-3)+2(-E_0-Em-Ep)^3-54\right)^2}}{2} - \frac{(-9E_0-9Em-9Ep)(E_0Em+E_0Ep+EmEp-3)}{2} + (-E_0 - Em - Ep)^3 - 27}} - \frac{\left(-\frac{1}{2} + \frac{\sqrt{3}i}{2}\right)\sqrt[3]{-\frac{27E_0EmEp}{2} + \frac{27E_0}{2} + \frac{27Em}{2} + \frac{27Ep}{2} + \frac{\sqrt{-4\left(-3E_0Em-3E_0Ep-3EmEp+(-E_0-Em-Ep)^2+9\right)^3 + \left(-27E_0EmEp+27E_0+27Em+27Ep-(-9E_0-9Em-9Ep)(E_0Em+E_0Ep+EmEp-3)+2(-E_0-Em-Ep)^3-54\right)^2}}{2} - \frac{(-9E_0-9Em-9Ep)(E_0Em+E_0Ep+EmEp-3)}{2} + (-E_0 - Em - Ep)^3 - 27}}{3}$$

The solutions are still quite big and complex. The reason behind this is that `sympy` is not very good at simplifying expressions, so it is giving us a huge expression.

Plotting the Energy Curves

Final step is evaluating the energy of the system by substituting the values of E_P, E_M and E_0 into the equation. We already have a function `give_energy` that returns the energy E, given k, m and a. By default, $m = \hbar = 1$. We'll use this defualt values. However, we can easily pass m and h (which is the placeholder for $\hbar$) as arguments to the function in place of `**kwargs` to change the default values.

In [11]:
```python
def substitute(E, K, **kwargs):
    """
    Takes one value of E and returns numerical value of E after making all the substit
    for a specific K value
    """
    #Substituing E0
    E_num = E.subs(E0, give_energy(K=K, **kwargs))
    #Substituing Ep which is nothing but E0 with K= K + G and as G = 2*pi, so K = K +
    #We'll be Using a=1 for simplicity
    E_num = E_num.subs(Ep, give_energy(K=K+2*np.pi, **kwargs))
    #Substituing Em which is nothing but E0 with K= K - G and as G = 2*pi, so K = K -
    #Again, we'll be Using a=1 for simplicity
    E_num = E_num.subs(Em, give_energy(K=K-2*np.pi, **kwargs))
    E_num = E_num.evalf()
    #Sometimes, we are getting a complex number, with very small imaginary part(ususa
    #because of the precision of the computer. So, we are removing the imaginary part
    E_num = E_num.subs(I, 0)
    return E_num

#Finally, we'll vectorize the function to run it faster
substitute = np.vectorize(substitute)
```

Let's do a sanity check!

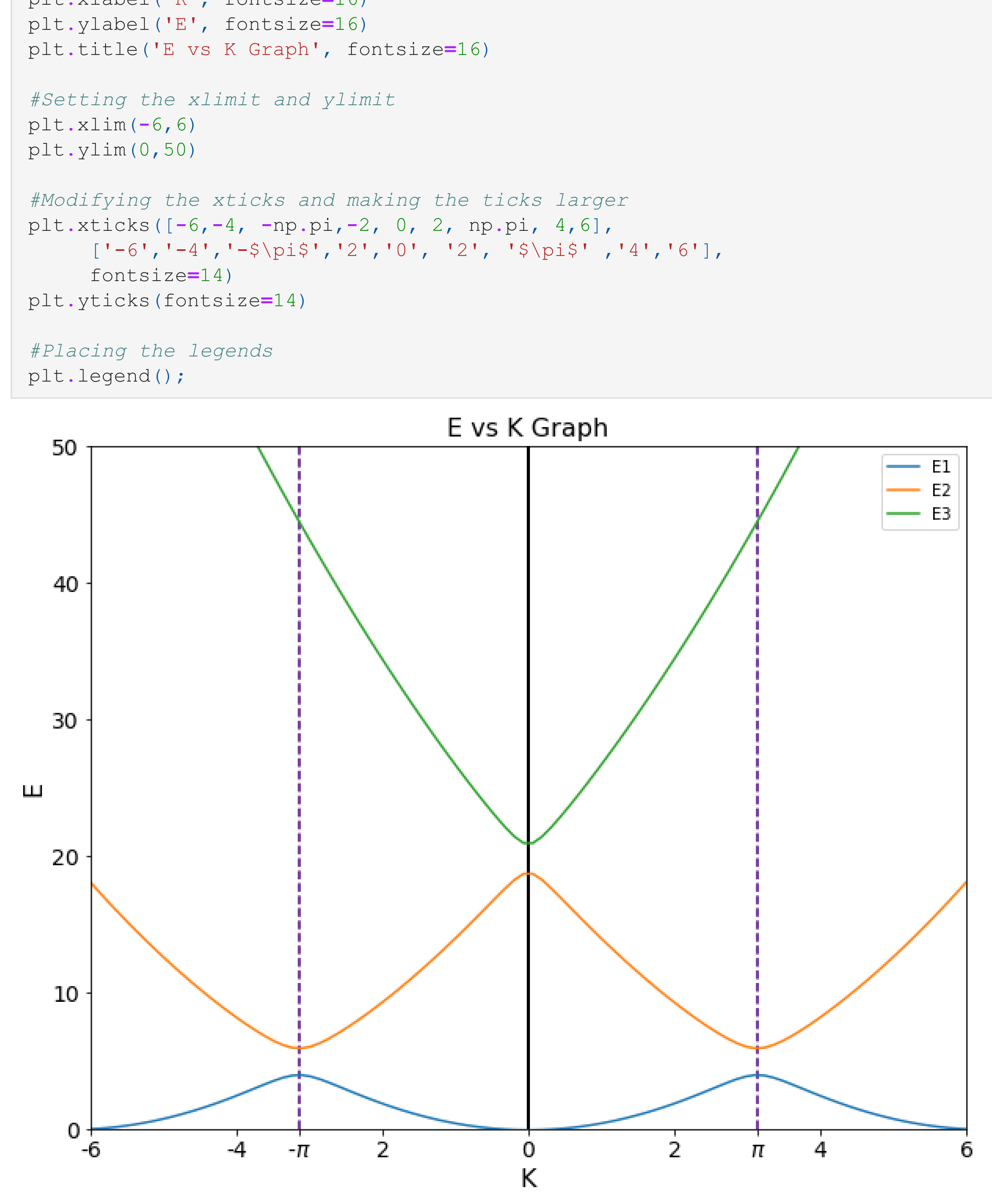
In [21]:
```python
display(float(substitute(E1, 0.5)))
display(float(substitute(E2, 0.5)))
display(float(substitute(E3, 0.5)))
display(float(substitute(E1, np.pi)))
display(float(substitute(E2, np.pi)))
display(float(substitute(E3, np.pi)))
```

```
0.026544447872931975
16.60906075547166
23.217812401012836
3.9348022005446843
5.88289503360872
44.46512697183807
```

So, it is working. Let's plot them.

In [13]:
```python
#Defining K values
K = np.linspace(-2*np.pi, 2*np.pi, 100)

#Getting the values of energies
Ev1 = substitute(E1, K)
Ev2 = substitute(E2, K)
Ev3 = substitute(E3, K)
```

In [19]:
```python
%matplotlib inline

#Setting Figure Size
plt.figure(figsize=(10,8))

#Plotting the three energies
plt.plot(K, Ev1, label='E1')
plt.plot(K, Ev2, label='E2')
plt.plot(K, Ev3, label='E3')

#Making three vertical lines, one for x=0, ie. y-axis
#and two for x=pi, and x= -pi

plt.vlines(0,0,85, color='black', linewidth=2)
plt.vlines(np.pi,0,85, color='indigo', linestyle='dashed')
plt.vlines(-np.pi,0,85, color='indigo', linestyle='dashed')

#Setting the x and y labels and title
plt.xlabel('K', fontsize=16)
plt.ylabel('E', fontsize=16)
plt.title('E vs K Graph', fontsize=16)

#Setting the xlimit and ylimit
plt.xlim(-6,6)
plt.ylim(0,50)

#Modifying the xticks and making the ticks larger
plt.xticks([-6,-4, -np.pi,-2, 0, 2, np.pi, 4,6],
    ['-6','-4','-$\pi$','2','0', '2', '$\pi$' ,'4','6'],
    fontsize=14)
plt.yticks(fontsize=14)

#Placing the legends
plt.legend();
```



This completes the assignment.