I) Discuss about the first Brillouin zone

The first Brillouin zone is primitive cell in reciprocal space. It is like Wigner–Seitz cells in real space. The bigger primitive cell is in real space, The smaller Brillouin zone is in reciprocal space.

II) Linear chain of atoms:

a) band structure of linear chain of atoms

$$H(K) = te^{(ikd)} + te^{(-ikd)} = 2tcos(kd)$$

b) when the unit cell is twice

$$H(K) = \begin{bmatrix} 0 & 2tcos(kd) \\ 2tcos(kd) & 0 \end{bmatrix}$$

c) when the site potentials for blue and orange have respectively VA = -1 eV and VB = 1 eV

$$H(K) = \begin{bmatrix} -1 & 2tcos(kd) \\ 2tcos(kd) & +1 \end{bmatrix}$$

d) Discuss your observations from the band structures in each of the above cases and also by changing parameters t , d and site potentials.

As t and site potential are bigger, band structure has high value. In case d, it has opposite effect.

III) Two coupled linear chain of atoms

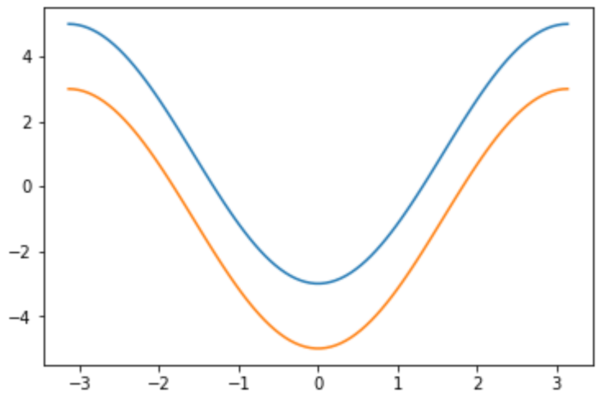
In [3]:
```
import pybinding as pb
import numpy as np
import matplotlib.pyplot as plt
from numpy import linalg as LA
```

a) band structure of two coupled linear chain of atoms

```
In [22]: #The structure I made:
         ###
         #a#
         #b#
         ###
         a = 1 # distance
         Va = 0  # site energy
         Vb = 0  # site energy
         t = -2 # horizontal hopping
         tv = 1 # vertical hopping

         R1 = [0,2]
         R2 = [1,0]
         R3 = [-1,0]
         energyband = []

         k = np.linspace(-np.pi/a, np.pi/a, 100) #wave vector k

         def f(R, x):
             return np.exp(1j*np.dot([x,0], [R[0],R[1]]))
         for i in range(len(k)):
             Hab = tv*(f(R1, k[i]))
             Hba = np.conjugate(Hab)
             Haa = Va+t*(f(R2, k[i])+f(R3, k[i]))
             Hbb = Vb+t*(f(R2, k[i])+f(R3, k[i]))
             H = np.array([[Haa,Hab],
                           [Hba,Hbb]])
             w,v = LA.eig(H)
             energyband.append(w)

         plt.figure()
         plt.plot(k,energyband, label='energy band')
```

```
/home/2013/2013550006/anaconda3/lib/python3.6/site-packages/numpy/core/nume
ric.py:492: ComplexWarning: Casting complex values to real discards the ima
ginary part
  return array(a, dtype, copy=False, order=order)
```

```
Out[22]: [<matplotlib.lines.Line2D at 0x7f05dd6dd048>,
          <matplotlib.lines.Line2D at 0x7f05dd6dd208>]
```

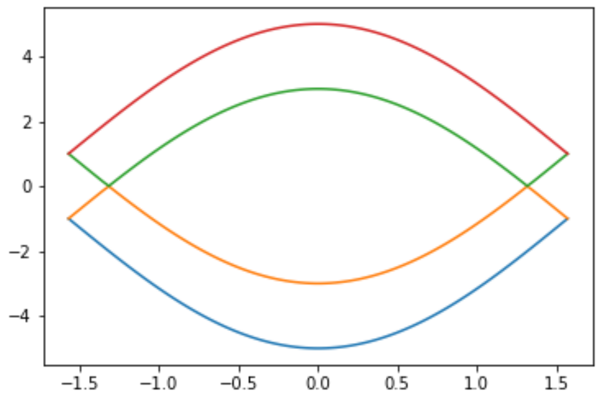


b) when the site potentials for blue and orange

In [21]:
```
VA = 0  # site energy of A
VB = 0 # site energy of B
t0 = -2 # horizontal hopping
t1 = 1 # vertical hopping
k = np.linspace(-np.pi/(2*a), np.pi/(2*a), 100) #wave vector k
R0 = [1,0]
R1 = [0,2]
R2 = [-1,0]

energyband = []
def f(R, x):
    return np.exp(1j*np.dot([x,0], [R[0],R[1]]))
for i in range(len(k)):
    HAA = VA
    Haa = VA
    HBB = VB
    Hbb = VB
    HAa = t1*(f(R1, k[i]))
    HaA = np.conjugate(HAa)
    HAB = t0*(f(R0,k[i])+f(R2,k[i]))
    HBA = np.conjugate(HAB)
    Hab = t0*(f(R0,k[i])+f(R2,k[i]))
    Hba = np.conjugate(Hab)
    HBb = t1*(f(R1, k[i]))
    HbB = np.conjugate(HBb)
    HAb = 0
    HbA = 0
    HaB = 0
    HBa = 0
    H = np.array([[HAA,HAa,HAB,HAb],
                  [HaA,Haa,HaB,Hab],
                  [HBA,HBa,HBB,HBb],
                  [HbA,Hba,HbB,Hbb]])
    w, v = LA.eig(H)
    w.sort()
    energyband.append(w)

plt.figure()
plt.plot(k,energyband, label='energy band')
```

```
/home/2013/2013550006/anaconda3/lib/python3.6/site-packages/numpy/core/nume
ric.py:492: ComplexWarning: Casting complex values to real discards the ima
ginary part
  return array(a, dtype, copy=False, order=order)
```

```
Out[21]: [<matplotlib.lines.Line2D at 0x7f05dd771e48>,
          <matplotlib.lines.Line2D at 0x7f05dd77a048>,
          <matplotlib.lines.Line2D at 0x7f05dd77a198>,
          <matplotlib.lines.Line2D at 0x7f05dd77a2e8>]
```

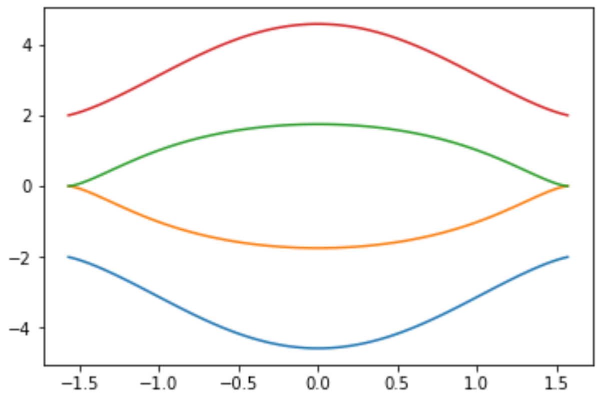


c) when the site potentials for blue and orange

In [47]:
```
#The structure I made:
 #####
 #A B#
 #a b#
 #####
VA = 1  # site energy of A
VB = -1 # site energy of B
t0 = 2 # horizontal hopping
t1 = 1 # vertical hopping
d=1
k = np.linspace(-np.pi/(2*a), np.pi/(2*a), 100) #wave vector k
R0 = [d,0]
R1 = [0,2*d]
R2 = [-d,0]

energyband = []
def f(R, x):
    return np.exp(1j*np.dot([x,0], [R[0],R[1]]))
for i in range(len(k)):
    HAA = VA
    Haa = VA
    HBB = VB
    Hbb = VB
    HAa = t1*(f(R1, k[i]))
    HaA = np.conjugate(HAa)
    HAB = t0*(f(R0,k[i])+f(R2,k[i]))
    HBA = np.conjugate(HAB)
    Hab = t0*(f(R0,k[i])+f(R2,k[i]))
    Hba = np.conjugate(Hba)
    HBb = t1*(f(R1, k[i]))
    HbB = np.conjugate(HBb)
    HAb = 0
    HbA = 0
    HaB = 0
    HBa = 0
    H = np.array([[HAA,HAa,HAB,HAb],
                  [HaA,Haa,HaB,Hab],
                  [HBA,HBa,HBB,HBb],
                  [HbA,Hba,HbB,Hbb]])
    w, v = LA.eig(H)
    w.sort()
    energyband.append(w)
plt.figure()
plt.plot(k,energyband, label='energy band')
```

```
/home/2013/2013550006/anaconda3/lib/python3.6/site-packages/numpy/core/nume
ric.py:492: ComplexWarning: Casting complex values to real discards the ima
ginary part
  return array(a, dtype, copy=False, order=order)
```

```
Out[47]: [<matplotlib.lines.Line2D at 0x7f05dcf9f518>,
          <matplotlib.lines.Line2D at 0x7f05dcf9f6d8>,
          <matplotlib.lines.Line2D at 0x7f05dcf9f828>,
          <matplotlib.lines.Line2D at 0x7f05dcf9f978>]
```

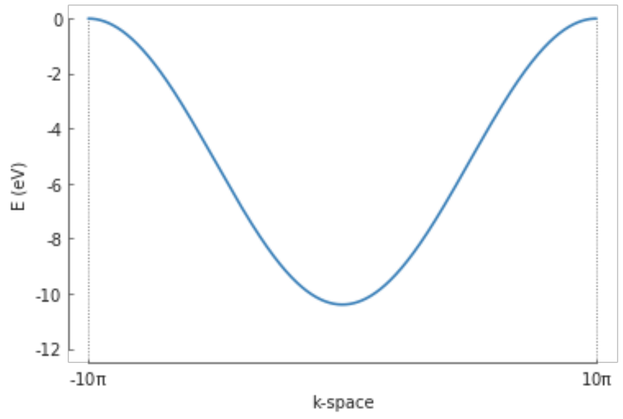


d) Discuss your observations

In this case, d makes brillouin zone much smaller portional to 1/d. t make band sturcture more round shape, tv makes flat shape. dv doesn't seems to make big difference.

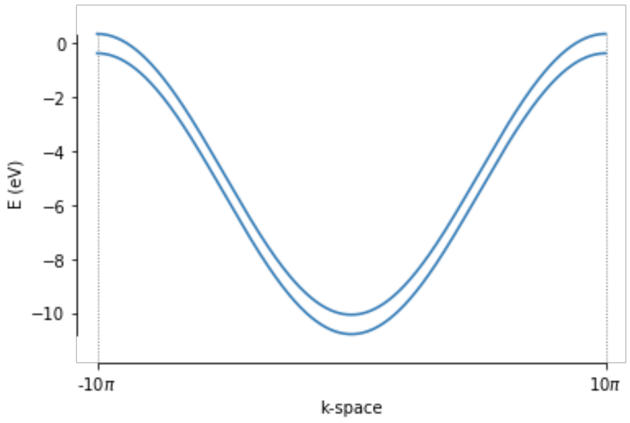
IV) Square lattice

In [68]:
```
d = 0.1  # [nm] unit cell length
t = -2.6    # [eV] hopping energy

def square_lattice(d, t):
    lat = pb.Lattice(a1=[d, 0], a2=[0, d])
    lat.add_sublattices(('A', [0, 0]))
    lat.add_hoppings(([0, 1], 'A', 'A', t),
                     ([1, 0], 'A', 'A', t))
    return lat

model = pb.Model(square_lattice(d,t), pb.translational_symmetry())

plt.figure()
solver = pb.solver.lapack(model)
bands = solver.calc_bands(-10*np.pi, 10*np.pi, step=0.1)
bands.plot()
```



V) Bilayer square lattice

In [30]:
```
d = 0.1  # [nm] unit cell length
t = -2.6    # [eV] hopping energy
tv = 0.36  # [eV] vertical hopping energy

def square_lattice(d, t):
    lat = pb.Lattice(a1=[d, 0], a2=[0, d])
    lat.add_sublattices(('A', [0,0,0]),
                        ('B', [0,0,1]))
    lat.add_hoppings(([0, 1], 'A', 'A', t),
                     ([1, 0], 'A', 'A', t),
                     ([0, 1], 'B', 'B', t),
                     ([1, 0], 'B', 'B', t),
                     ([0, 0], 'A', 'B', tv)
                    )
    return lat

model = pb.Model(square_lattice(d,t), pb.translational_symmetry())

plt.figure()
solver = pb.solver.lapack(model)
bands = solver.calc_bands(-10*np.pi, 10*np.pi, step=0.1)
bands.plot()
```


E (eV)
-10π
10π
k-space