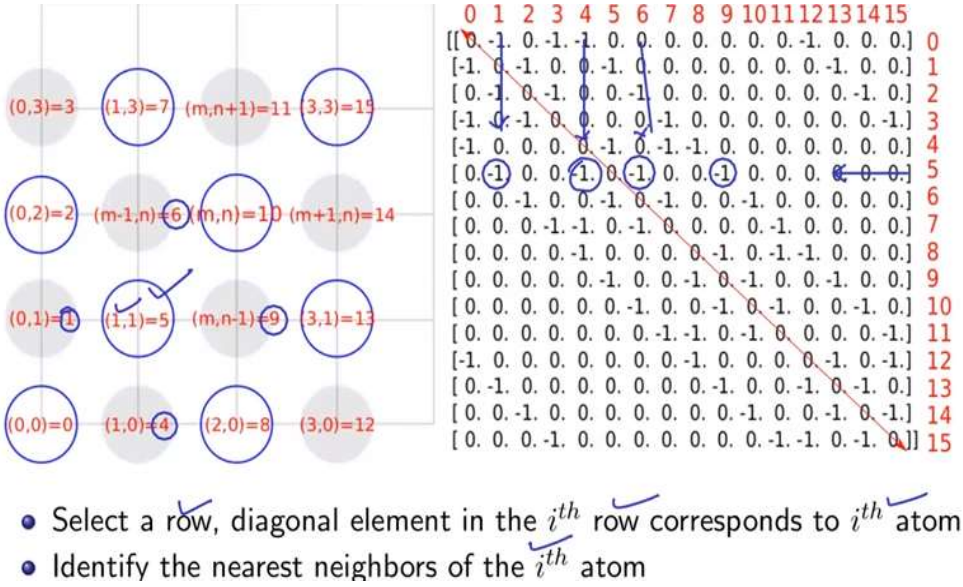
Electronics Properties of the Materials: Computational Approach
The Tight-Binding Method
Prof. Somnath Bhowmick
Department of Material Science and Engineering
Indian Institute of Technology, Kanpur

Module No # 07
Lecture No # 35
Tight Binding Method: Part 2

Hello friends we are going to continue our discussion on tight-binding model in this lecture. I already have discussed the technical details of type-binding model for the only lattice. In this lecture I am going to show how to extend the higher dimensions. I am also going to discuss how to build the type-binding model starting from the isolated atoms.

(Refer Slide Time: 00:41)



- Select a row, diagonal element in the i^{th} row corresponds to i^{th} atom
- Identify the nearest neighbors of the i^{th} atom

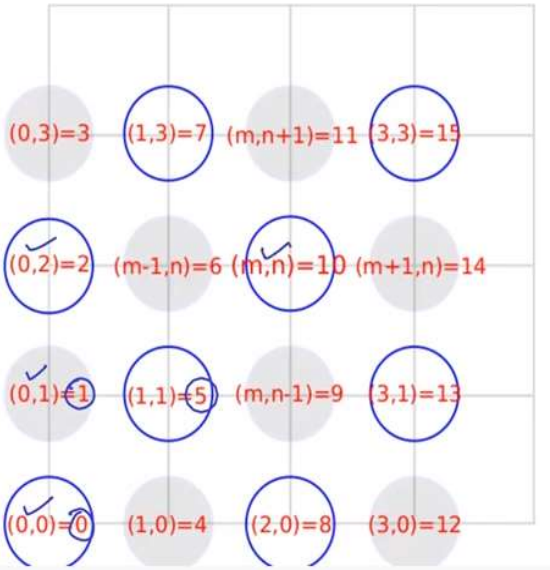
Now let us extend our tight-binding recipe to a square lattice. Step 1 select a row diagonal element in the ith row corresponds to the ith atom step 2 identify the nearest neighbours of the ith atom. In case of square lattice there are four nearest neighbours say a, b, c and d. Enter - t in ath bth cth and dth column of the ith row. Let us take some example say the fifth atom first we have to index each lattice side.

This is the fifth atom sitting at the 11 position in the lattice left hand side neighbour of the fifth atom has an index 1. So we go to the fifth row and the first column that means this side and we put -1 right hand side neighbour of the fifth atom as an index 9. So we have to go to column 9 in the

fifth row that means this 1 and we enter -1 top neighbour of the fifth atom as an index 6. Thus we have to go to the sixth column in the fifth row that is this column and we have to enter -1 bottom neighbour of the fifth atom as an index 4. Thus in the fifth row we have to go to the fourth column that is this 1 and enter -1 rest of the terms in fifth row are equal to 0.

(Refer Slide Time: 03:00)

```
#indexing the atoms
for i in range(n):
    for j in range(n):
        idx[i,j] = k
        k += 1
#building the neighbor list
k=0
for i in range(n):
    for j in range(n):
        if i < (n-1):
            nlist[k,0] = idx[i+1,j]
        elif i == (n-1):
            nlist[k,0] = idx[0,j]
        if i > 0:
            nlist[k,1] = idx[i-1,j]
        elif i == 0:
            nlist[k,1] = idx[n-1,j]
        if j < (n-1):
            nlist[k,2] = idx[i,j+1]
        elif j == (n-1):
            nlist[k,2] = idx[i,0]
        if j > 0:
            nlist[k,3] = idx[i,j-1]
        elif j == 0:
            nlist[k,3] = idx[i,n-1]
        k += 1
#constructing and diagonalizing the hamiltonian
l=0
for i in range(n*n):
    for j in range(n*n):
        for k in range(4):
            atm=nlist[l,k]
```



This is the python code for the square lattice. I show only the portion to build the neighbour list and construct the Hamilton rest of it is very similar to the 1 D code. First step is to index each side we have a square grid with each grid point having co-ordinates like 0 0, 0 1, 0 2 etc. Using a fall loop i index each grid point for example 0 0 point as an index 0. 0 1 Point has an index equal to 1 and 1 1 point as an index to equal to 5 etc.

Next I build the neighbour list let us take m n sight. In the code I have used i and j in place of m n. The right hand side neighbour is m+1, n as defined here in the code. The left hand side neighbour is m -1 and as defined here in the code. The top neighbour is m, n+1 as defined in the code and the bottom number is m n-1 as defined here in the code. Note that I also have implemented periodic boundary condition in this code.

For example if we take the fourteenth atom which is at the edge of the box it does not have a right hand side neighbour if we do not have periodic boundary condition. If we have a periodic box then atom number 2 is repeated periodically over here. As a result the right hand side neighbour of atom 40 is atom 2 if we have a periodic box .This is what is implemented in this line of the code and

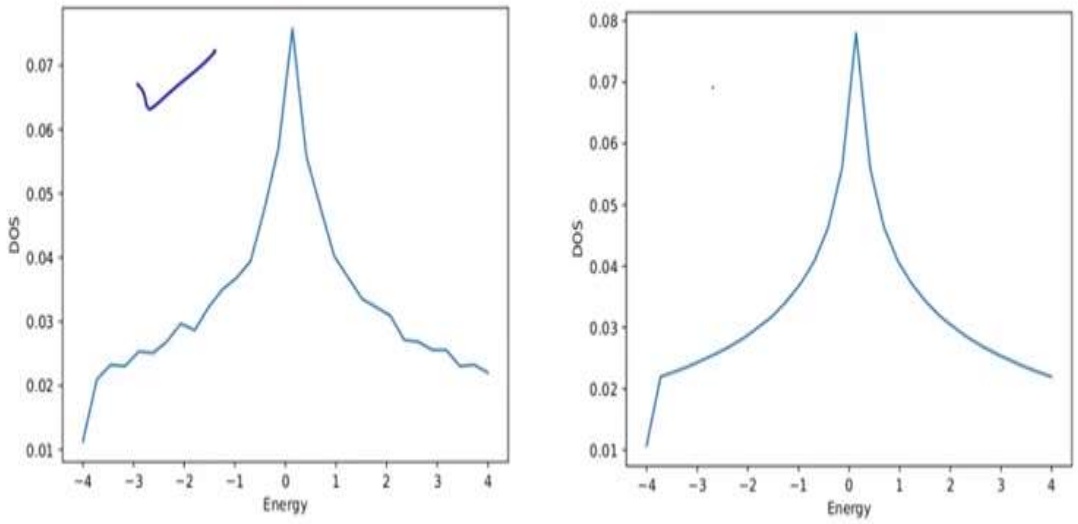
similarly for other neighbour's. Then I construct the Hamiltonian Matrix I find the nearest neighbours for each atom. There are 4 nearest neighbours and I enter –t in the corresponding terms of the matrix.

(Refer Slide Time: 05:55)

```
n=int(input("Enter the size of the square lattice n x n: "))
t=1
a=np.zeros([n*n,n*n],float)
idx=np.zeros([n,n],int)
nlist=np.zeros([n*n,4],int)
k=0
#indexing the atoms
for i in range(n):
    for j in range(n):
        idx[i,j] = k
        k += 1
#building the neighbor list
k=0
for i in range(n):
    for j in range(n):
        if i < (n-1):
            nlist[k,0] = idx[i+1,j]
        elif i == (n-1):
            nlist[k,0] = idx[0,j]
        if i > 0:
            nlist[k,1] = idx[i-1,j]
        elif i == 0:
            nlist[k,1] = idx[n-1,j]
        if j < (n-1):
            nlist[k,2] = idx[i,j+1]
        elif j == (n-1):
            nlist[k,2] = idx[i,0]
        if j > 0:
            nlist[k,3] = idx[i,j-1]
```

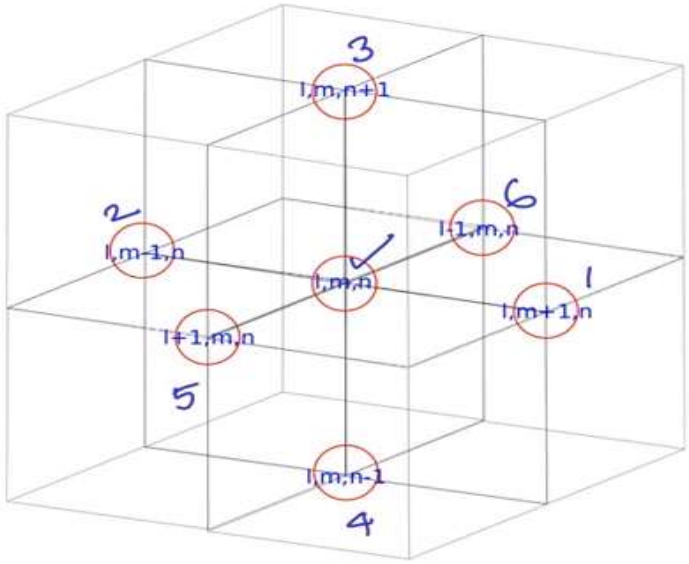
This is the code for the 2D Square lattice right I indexed atoms using for loop here then I build the neighbour list here I construct the Hamiltonian. Here the Hamiltonian is diagonalzed to get the energy Eigen values and then I calculate dos here and finally the density of states is plotted in this step.

(Refer Slide Time: 06:40)



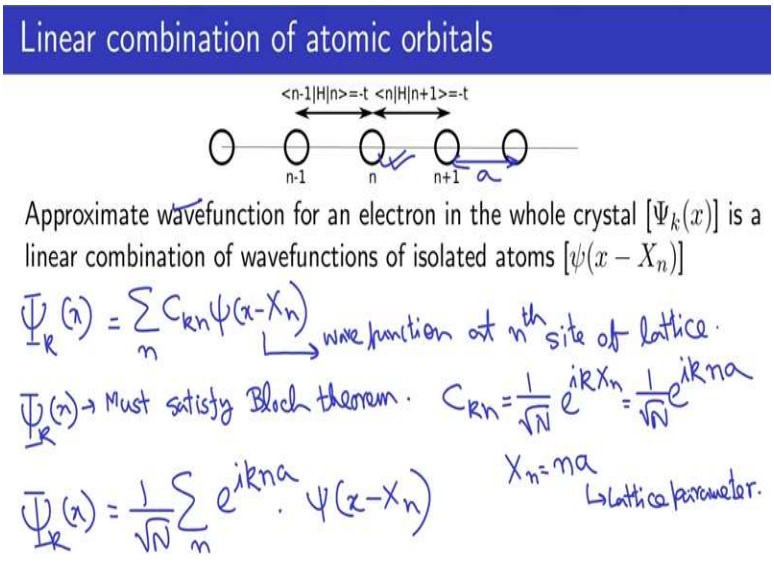
This is how the density of States looks like for 2D Square lattice. This is the analytical result we get a good match. Densities of states for a square lattice have a logarithmic singularity at e = 0.

(Refer Slide Time: 07:03)



Finally let us discuss a 3D lattice the simple cubic lattice. I just show how to get the neighbour list and leave it as an exercise for you to write the rest of the code. A lattice point has 6 nearest neighbours in case of a simple cubic lattice. 1, 2, 3, 4, 5, 6 I have shown the coordinate of each lattice point it should not be too difficult for you to write the code using our type-binding testing.

(Refer Slide Time: 07:45)



Now we know technically what could do to get energy Eigen value of an electron in a periodic potential using type-binding model. Let us now see how we get type binding model starting from

wave functions of free atoms. This is also known as linear combination of atomic orbital approximation. Approximate wave function for an electron in the whole crystal is a linear combination of wave functions of isolated atoms.

So big psi is the wave function of an electron in the whole crystal and this is a linear combination of atomic wave functions. C is a constant and psi is wave function at nth site of lattice. For example this is the nth psi so psi is the wave function localized at this site.

$$\Psi_k(x) = \sum_n C_{kn}\, \psi(x - X_n)$$

Since we are talking about a lattice the wave function must satisfy block theorem. So psi k of x must satisfy block theorem for that to happen this constants C kn should be equal to 1 by square root of N where N is the total number of lattice sites e power ik X n.

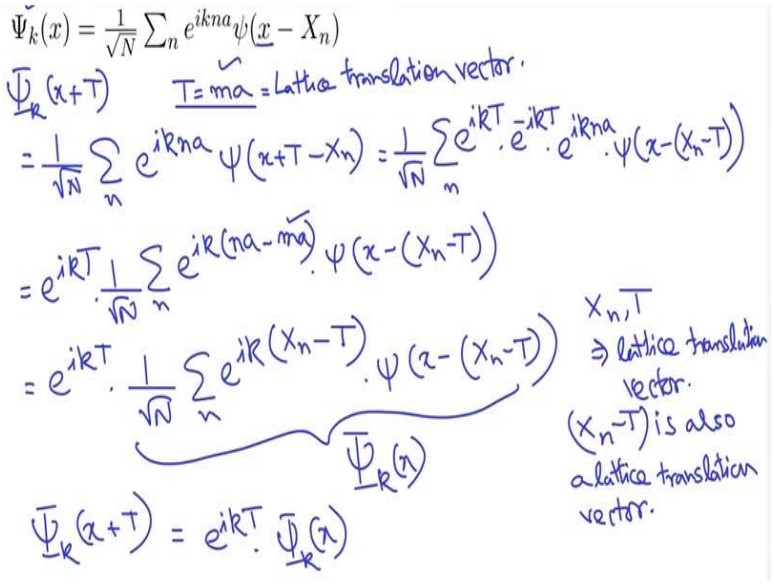
So X n is a lattice translation vector side to that X n = n times a where a, is the lattice parameter that is the distance between 2 neighbouring sides. That C kn = 1 by square root of N e power ikn times a

$$C_{kn} = \frac{1}{\sqrt{N}} e^{ikX_n} = \frac{1}{\sqrt{N}} e^{ikna}; X_n = na$$

so we can write the psi function s psi k of x = 1 by square root of n sum over n e power ikna . Where n is an integer times psi so psi is the atomic wave function localized at the nth site.

$$\Psi_k(n) = \frac{1}{\sqrt{N}} \sum_n e^{ikna}\, \psi(x - X_n)$$

(Refer Slide Time: 11:33)



Let me prove that Psi satisfies Bloch theorem.

$$\Psi_k(x+T) = \frac{1}{\sqrt{N}} \sum_n e^{ikna}\, \psi(x+T-X_n) = \frac{1}{\sqrt{N}} \sum_n e^{ikT} \,.\, e^{-ikT} \,.\, e^{ikna} \,.\, \psi\big(x-(X_n-T)\big)$$

$$= e^{ikT} \frac{1}{\sqrt{N}} \sum_n e^{ik(na-ma)} \,.\, \psi\big(x-(X_n-T)\big)$$

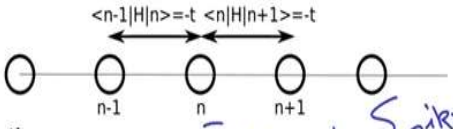
$$= e^{ikT} \frac{1}{\sqrt{N}} \sum_n e^{ik(X_n-T)} \,.\, \psi\big(x-(X_n-T)\big)$$

Now note that both X_n and T are lattice translation vector that means X n - T is also a lattice translation vector. This implies that this term is equal to Psi k of x. Thus, we have Psi k of x + T = e power ik T times Psi k of x.

$$\Psi_k(x+T) = e^{ikT} \Psi_k(x)$$

In conclusion the wave function indeed satisfies the Bloch theorem.

(Refer Slide Time: 15:59)



$$\Psi_k(x) = \frac{1}{\sqrt{N}} \sum_n e^{ikna} \psi(x - na)$$

$$\bar{\Psi}_k(x) = \frac{1}{\sqrt{N}} \sum_n e^{ikna} \Psi_n(x).$$

$$\langle \bar{n} | H | n \rangle = \int \psi_n^* H \psi_n \, dx = \varepsilon_0 = \text{Constant.}$$ $H_{n,n}$ → Diagonal term in Hamiltonian matrix.

$$\langle \bar{n+1} | H | n \rangle = \int \psi_{n+1}^* H \psi_n \, dx = -t = H_{n+1,n}$$

$$\langle \bar{n-1} | H | n \rangle = \int \psi_{n-1}^* H \psi_n \, dx = -t = H_{n-1,n}$$

$$\langle m | H | n \rangle = \int \psi_m^* H \psi_n \, dx = 0 = H_{m,n}.$$

So, we have obtained the wave function of electron in a periodic potential by using the linear the combination of atomic orbitals. Psi is the atomic wave function centered at atom n and I just rewrite this equation by slightly changing the notation. For that atomic wave function I just write the subscript n. Now let us calculate the following terms

$$\langle n|H|n\rangle = \int \psi_n^* H \psi_n dx = \varepsilon_0 = \text{constant}$$

This is the $H_{n,n}$ that is the diagonal term in Hamiltonian matrix.

Next we calculate

$$\langle n+1|H|n\rangle = \int \psi_{n+1}^* H \psi_n dx = -t = H_{n+1,n}$$

–t is one of the off diagonal terms of the Hamiltonian matrix.

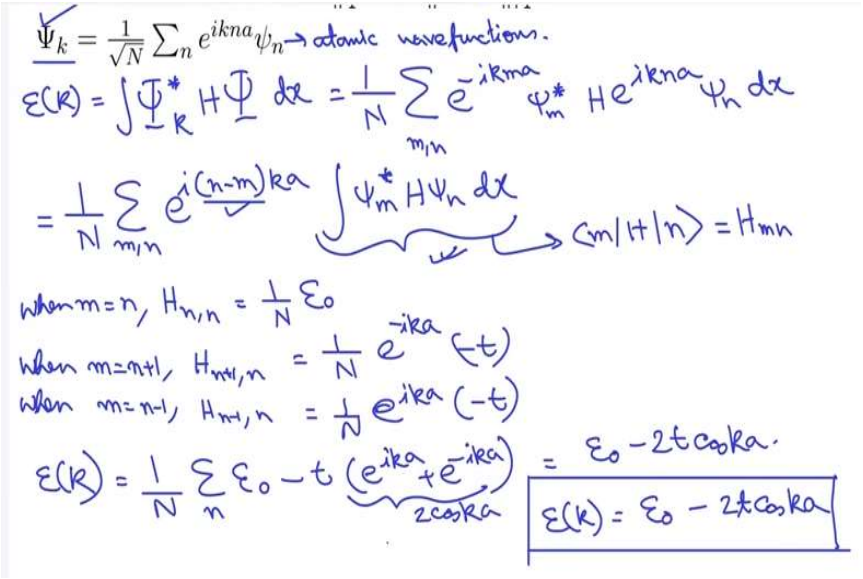
Next, we calculate

$$\langle n-1|H|n\rangle = \int \psi_{n-1}^* H \psi_n dx = -t = H_{n-1,n}$$

this is another off diagonal term of the Hamiltonian Matrix H n-1 n for any value other than n n+1 and n-1 the individual is 0.

$$\langle m|H|n\rangle = \int \psi_m^* H \psi_n dx = 0$$

(Refer slide Time: 19:19)



So this is the wave function of an electron in a periodic potential obtained by linear combination of atomic wave functions. So now let us try to calculate the energy Eigen value for this particular wave function. This is given by integral Psi star H Psi dx which is equal to 1 by N sum over m, n so first I write psi star which is equal to e power – ikma times Psi m star H and then I write Psi which is equal to e power ikna Psi of n dx and this is equal to 1 by n sum over m n distort we can bring outside the integral and then we have integral Psi m star H Psi n dx.

Note that we already have evaluated these integrals. And this is also related to the m n element of the matrix.

$$\varepsilon(k) = \int \Psi_k^* H \Psi dx = \frac{1}{N} \sum_{m,n} e^{-ikm} \ \psi_m^* H e^{ikna} \psi_n dx$$

$$= \frac{1}{N} \sum_{m,n} e^{i(n-m)ka} \int \psi_m^* H \, \psi_n dx$$

Now when $m = n, H_{n,n} = \frac{1}{N} \varepsilon_0$

When $m = n + 1, H_{n+1,n} = \frac{1}{N} e^{-ika}(-t)$

When $m = n - 1, H_{n-1,n} = \frac{1}{N} e^{ika}(-t)$

Thus we have E of k = 1 by N sum over n E naught –t e power ika + e power - ika and this term is equals to 2 Cos ka such that we get E naught –2t Cos ka.

$$\varepsilon(k) = \frac{1}{N}\sum_{N} \varepsilon - t\left(e^{ika} + e^{-ika}\right) = \varepsilon_0 - 2t\cos k\,a$$

Thus the energy dispersion relation is E of k = e not -2t Cos ka. Note that E naught is a constant that without any loss of generality we can set it to 0. That is what we have done when we have written the code.