Crystal Shapes

Christina Lee

July 8, 2017

Category: General
Prerequisites: none

In condensed matter, we find ourselves in the interesting middle ground of dealing with large numbers 10^{23} of extremely small particles, ex. atoms, electrons.

Luckily, the particles don't each do their own thing, but often come in nice, structured, repeated units. Lattices. So as our first step into the field, we will look at the most basic type, a Bravais Lattice.

In a Bravais Lattice, every site looks like every other site. Mathematically, we use three vectors, $\vec{a}, \vec{b}, \vec{c}$ to express how we move from one site to a neighbor.

$$\mathbf{R}_{lmn} = l\vec{a} + m\vec{b} + n\vec{c} \quad \text{for } l, m, n \in \mathbb{N} \tag{1}$$

To keep things working out right, we have to put a constraint on these vectors; that we can't get one from scaling and adding the other two. If we could, then we couldn't put sites in an entire 3 dimensional space.

Stay tuned for a later post where we explore more elaborate lattices.

```
In [9]: # importing our packages
        using Plots
        plotlyjs()

Out[9]: Plots.PlotlyJSBackend()
```

0.1 Define The Relevant Variables

Choose the lattice you want to look at, and put that string into the lattice variable.
Current options:

- Simple Cubic = "sc"
- Plane triangular lattice = "pt"
- Body-Centered Cubic = "bcc"
- Face-Centered Cubic = "fcc"

Note: Square is Simple Cubic for Nz=1
14 distinct lattice types are possible, but these common four give the important ideas.
Also, input the size of lattice you want to look at.

```
In [47]: lattice="pt";

         Nx=3;
         Ny=3;
         Nz=3;
```

0.2 The Lattice vectors

```
In [48]: # A cell to just evaluate
         # This one sets the unit vectors (a,b,c) for the different unit cells
         # Can you guess what a lattice will look like by looking at the vectors?
         if(lattice=="sc")
             d=3;
             a=[1,0,0];
             b=[0,1,0];
             c=[0,0,1];
         elseif(lattice=="pt")
             d=2;
             a=[1,0,0];
             b=[.5,sqrt(3)/2,0];
             c=[0,0,1];
         elseif(lattice=="bcc")
             d=3;
             a=[.5,.5,.5];
             b=[.5,.5,-.5];
             c=[.5,-.5,.5];
         elseif(lattice=="fcc")
             d=3;
             a=[.5,.5,0];
             b=[.5,0,.5];
             c=[0,.5,.5];
         end
         "Cell Finished"

Out[48]: "Cell Finished"
```

1 Creating the Position arrays

Now we have everything we need about the specific lattice we want to look at. From here on out, everything is completely general for any Bravais Lattice.

1.1 Some helpful variables

The variables created in the next cell just make computation a bit easier, get the above variables into nicer forms.

```
In [49]: # Another cell to just evaluate
         N=Nx*Ny*Nz;     #The total number of sites
         #these allow us to copy an entire row or layer at once
         aM=transpose(a);
         bM=transpose(repeat(b,outer=[1,Nx]));
         cM=transpose(repeat(c,outer=[1,Nx*Ny]));

         X=Array{Float64}(N,3);  #where we store the positions
         "Cell Finished"

Out[49]: "Cell Finished"
```

1.2 Entering the Position Values

Here, we first initialize a first row, then tile the first plane with the first row. Afterward, we can tile the entire lattice with that first plane.

```
In [50]: # Another cell to just evaluate
         # Here we are actually calculating the positions for every site
         for i in 1:Nx    #for the first row
             X[i,:]=(i-1)*a;
         end

         for j in 2:Ny    #copying the first row into the first layer
             X[Nx*(j-1)+(1:Nx),:]=X[1:Nx,:]+(j-1)*bM;
         end

         for j in 2:Nz    #copying the first layer into the entire cube
             X[Ny*Nx*(j-1)+(1:Nx*Ny),:]=X[1:Nx*Ny,:]+(j-1)*cM;
         end
```

Programming Tip: In Julia, ranges, like `1:Nx`, are a special variable type that can be manipulated. We can add numbers to them: `3+(1:3)=4:6`, or add a minus sign to force it to iterate in the opposite direction, though with different start/stop: `-(1:3)=-1:-1:-3`

Danger! Make sure to use the parentheses around the range if you are performing these operations.

1.3 Plotting

```
In [52]: scatter(X[:,1],X[:,2],X[:,3],markershape=:circle)

         ls=2
         v=collect(0:ls)
         zed=zeros(v)
         for i in 0:ls
             for j in 0:ls
                 plot!(zed+i,v,zed+j)
                 plot!(zed+i,zed+j,v)
```

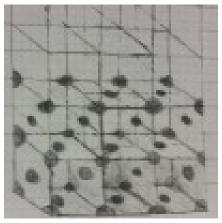
```
            plot!(v,zed+i,zed+j)
            plot!(zed+j,zed+i,v)

            plot!(v,zed+j,zed+i)
            plot!(zed+j,v,zed+i)
        end
    end
    plot!(xlabel="x",ylabel="y",zlabel="z",title="$lattice")
    savefig("$lattice.html")
```

1.4 Go Back and Fiddle!

As you might have noticed, this isn't just a blog where you just read through it. Interact with it. Change some lines, and see what happens. I choose body centered cubic to display first, but what do the other lattices look like?

Chose `pygui(true)` to pop open a window and manipulate the plot in 3D.

Look at different lattice sizes.

Can you hand draw them on paper? 

```
In [ ]:
```