Wigner Seitz Cell for simple cubic lattice

Define origin. Define a length scale a for graphical representation.

In[∘]:=
```
a = 2;
o = {0, 0, 0};
c = 2 π / a;
```

Define primitive vectors of the reciprocal lattice

In[∘]:=
```
b_1 = c {1, 0, 0};
b_2 = c {0, 1, 0};
b_3 = c {0, 0, 1};
```

Plot origin and first neighbours

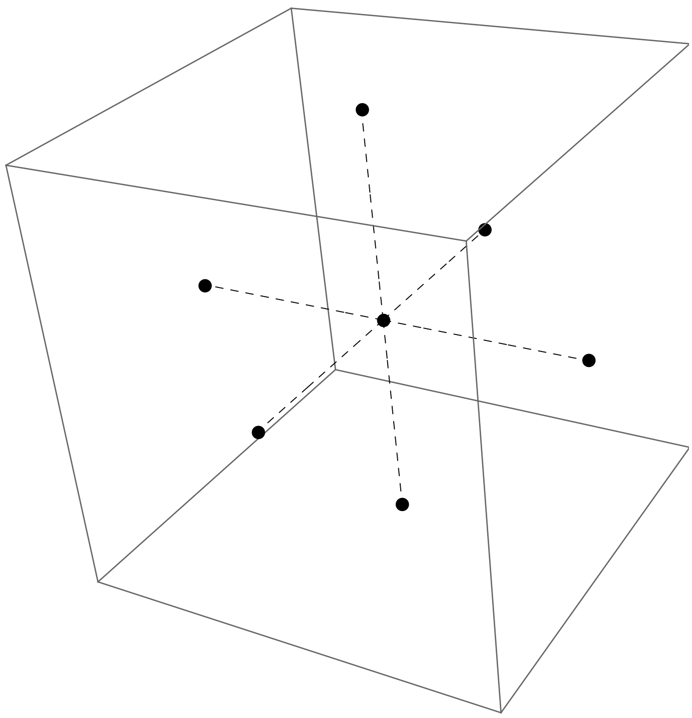
In[∘]:=
```
nodes = ListPointPlot3D[
   {o, o + b_1, o - b_1, o + b_2, o - b_2, o + b_3, o - b_3}, Filling → None, PlotStyle → Black];
```

Plotting connecting lines

In[∘]:=
```
g1 = Graphics3D[{Arrowheads[.0], Dashed, Arrow[{o, b_1}]}];
g2 = Graphics3D[{Arrowheads[.0], Dashed, Arrow[{o, -b_1}]}];
g3 = Graphics3D[{Arrowheads[.0], Dashed, Arrow[{o, b_2}]}];
g4 = Graphics3D[{Arrowheads[.0], Dashed, Arrow[{o, -b_2}]}];
g5 = Graphics3D[{Arrowheads[.0], Dashed, Arrow[{o, b_3}]}];
g6 = Graphics3D[{Arrowheads[.0], Dashed, Arrow[{o, -b_3}]}];
```

In[∘]:=
```
first = Show[g1, g2, g3, g4, g5, g6, nodes]
```

Out[∘]=



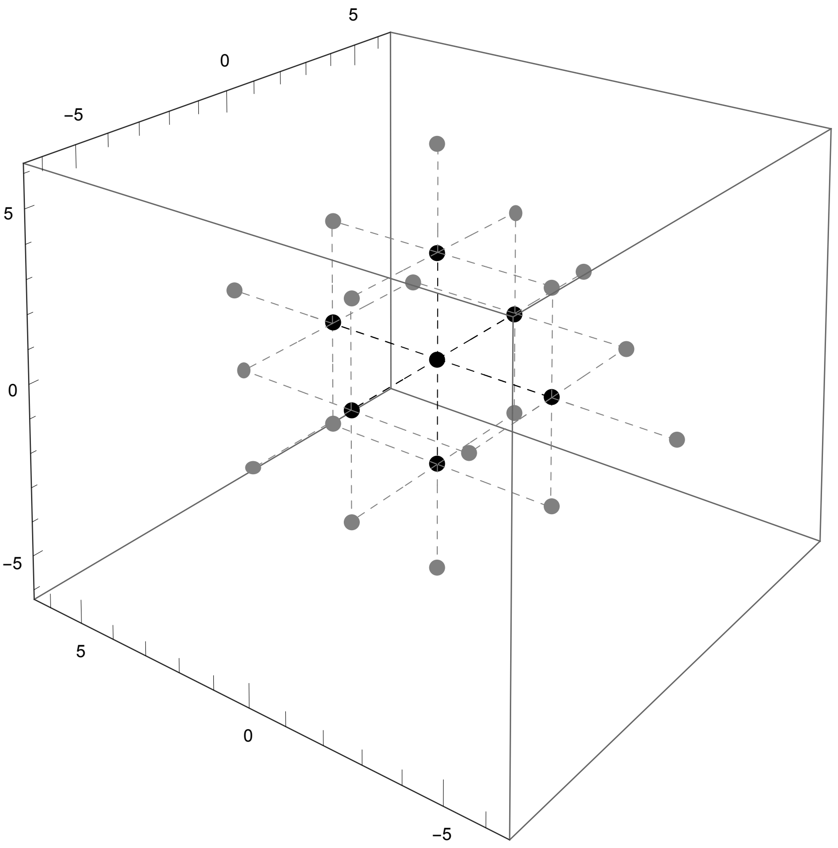
Adding second nearest neighbours for visualization

In[•]:=
```
nodes2 = ListPointPlot3D[
   {b1 + b1, b1 + b2, b1 - b2, b1 + b3, b1 - b3, b2 - b1, b2 + b3, b2 - b3, b2 + b2, b3 + b3,
    b3 - b1, b3 - b2, -b2 - b2, -b1 - b1, -b3 - b3, -b1 - b2, -b1 - b3, -b2 - b3},
   Filling → None, PlotStyle → Gray, PlotRange → Full];
```

In[•]:=
```
m1 = Graphics3D[{Arrowheads[.0], Dashed, Gray,
    Arrow[{b1, b1 + b1}], Arrow[{b1, b1 + b2}], Arrow[{b1, b1 - b2}],
    Arrow[{b1, b1 + b3}], Arrow[{b1, b1 - b3}], Arrow[{b2, b2 + b3}],
    Arrow[{b2, b2 + b1}], Arrow[{b2, b2 - b1}], Arrow[{b2, b2 - b3}],
    Arrow[{b2, b2 + b2}], Arrow[{b3, b3 + b3}], Arrow[{b3, b3 + b1}],
    Arrow[{b3, b3 - b1}], Arrow[{b3, b3 + b2}], Arrow[{b3, b3 - b2}]}];
m2 = Graphics3D[{Arrowheads[.0], Dashed, Gray, Arrow[{-b1, -b1 - b1}],
    Arrow[{-b1, -b1 + b2}], Arrow[{-b1, -b1 - b2}], Arrow[{-b1, -b1 + b3}],
    Arrow[{-b1, -b1 - b3}], Arrow[{-b2, -b2 + b1}], Arrow[{-b2, -b2 + b3}],
    Arrow[{-b2, -b2 - b1}], Arrow[{-b2, -b2 - b3}], Arrow[{-b2, -b2 - b2}],
    Arrow[{-b3, -b3 - b3}], Arrow[{-b3, -b3 + b1}], Arrow[{-b3, -b3 - b1}],
    Arrow[{-b3, -b3 + b2}], Arrow[{-b3, -b3 - b2}]}];
```

In[•]:=
```
seconds = Show[nodes2, nodes, m1, m2, g1, g2, g3, g4, g5, g6, AspectRatio → 1]
```

Out[•]=



Define a plane normal to a vector.

n•(**x**-**x0**)=0

n is a primitive vector of the reciprocal lattice: b1,b2,b3

x0 is a point laying on the plane.

x=(x,y,z) is a generic vector on the plane to be defined.
The equation becomes:
nx x + ny y + nz z = nx x0 + ny y0 + nz z0

3-dimensional plane: 2 degrees of freedom: z=z(f). The plane equation is

$z=\frac{\text{nx x0} + \text{ny y0} + \text{nz z0} - (\text{nx}\, x + \text{ny}\, y)}{\text{nz}}$ if nz ≠ 0

If nz=0 the plane equation is
nx x + ny y = nx x0 + ny y0

then no constraints on z, and
$y=\frac{\text{nx x0} + \text{ny y0} - (\text{nx}\, x)}{\text{ny}}$ if ny ≠ 0 (holds true if nx=0 too)
OR

$x=\frac{\text{nx x0} + \text{ny y0} - (\text{ny}\, y)}{\text{nx}}$ if nx ≠ 0 (holds true if ny=0 too)

Thus, we can define three functions based on these considerations:
- if normal vector has nz ≠ 0

```
In[∘]:= plane0[normal_, point_] :=
  Plot3D[(normal.point - normal[[1]] x - normal[[2]] y)/normal[[3]], {x, -c / 2, c / 2},
   {y, -c / 2, c / 2}, Mesh → None, PlotStyle → {Yellow, Opacity[0.25]}]
```

- if normal vector has y coordinate ≠ 0

```
In[∘]:= plane1[normal_, point_] := ParametricPlot3D[{x, (normal.point - normal[[1]] x)/normal[[2]], z},
   {x, -c / 2, c / 2}, {z, -c / 2, c / 2}, Mesh → None, PlotStyle → {Red, Opacity[0.25]}]
```

- if normal vector has x coordinate ≠ 0

```
In[∘]:= plane2[normal_, point_] := ParametricPlot3D[{(normal.point - normal[[2]] y)/normal[[1]], y, z},
   {y, -c / 2, c / 2}, {z, -c / 2, c / 2}, Mesh → None, PlotStyle → {Blue, Opacity[0.25]}]
```

We put everything together to define a general function with three cases

```
In[∘]:= plane[normal_, point_] := If[normal[[3]] ≠ 0, plane0[normal, point],
    If[normal[[2]] ≠ 0, plane1[normal, point], plane2[normal, point]]];
```

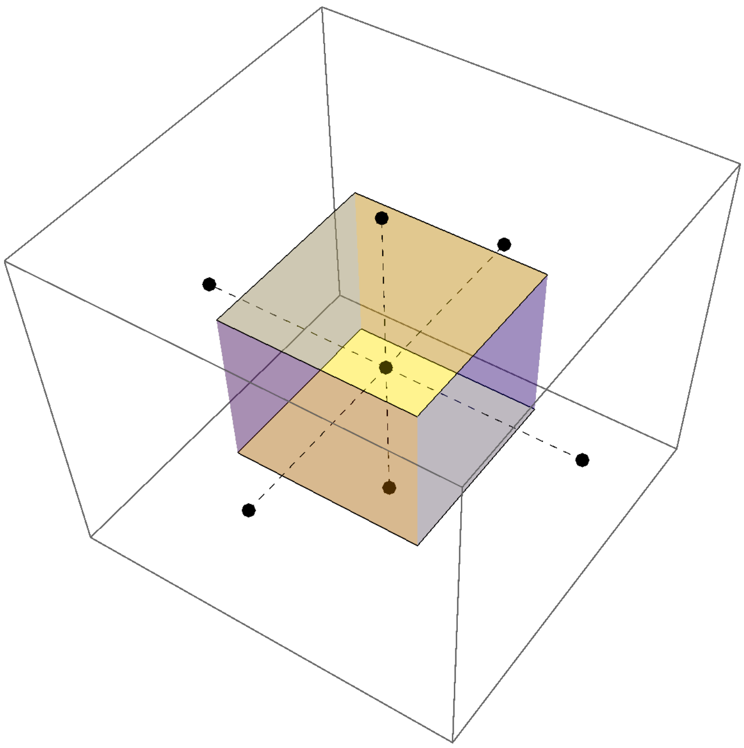
Plot planes: perpendicular to primitive vectors $\pm\boldsymbol{b}_1$,$\pm\boldsymbol{b}_2$,$\pm\boldsymbol{b}_3$. The Wigner-Seits is defined by the vectors having modulus smaller than the half distance to the center of the next cell

In[∘]:=
```
p01 = plane[b1, b1 / 2];
p10 = plane[-b1, -b1 / 2];
p02 = plane[b2, b2 / 2];
p20 = plane[-b2, -b2 / 2];
p03 = plane[b3, b3 / 2];
p30 = plane[-b3, -b3 / 2];
```

Plot planes on 1st neighbours

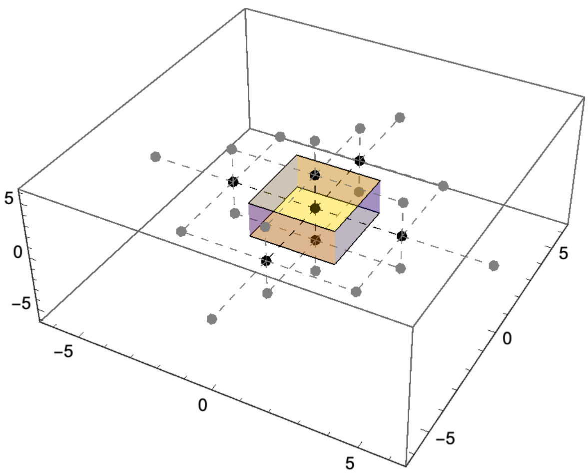
In[∘]:=
```
WScell = Show[first, p01, p02, p03, p10, p20, p30, AspectRatio → 1]
```

Out[∘]=



In[∘]:=
```
Show[seconds, p01, p02, p03, p10, p20, p30, AspectRatio → Automatic]
```

Out[∘]=



Points of interest:

Gamma: **0**

M: center of an edge

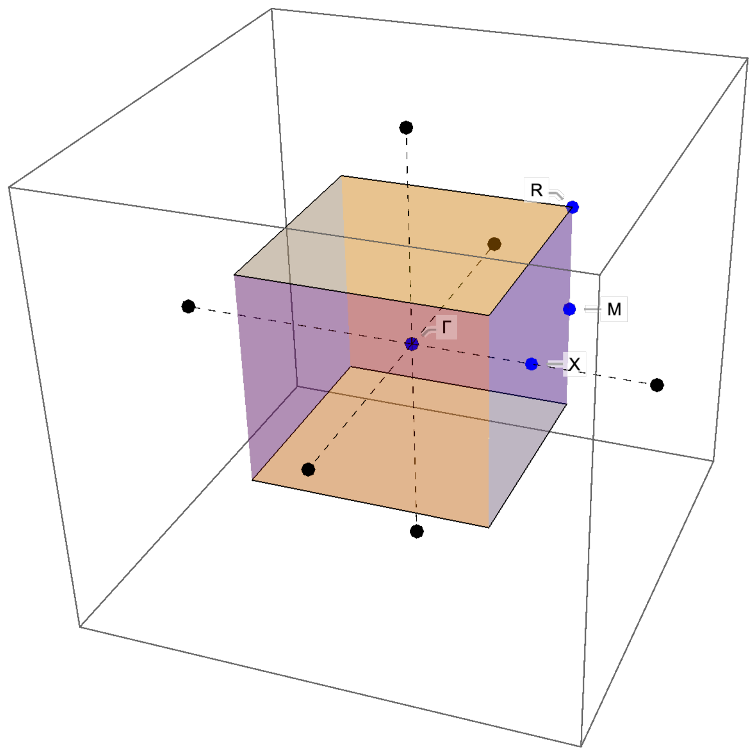
R: vertex

X: center of a face

In[∘]:=
```
points = ListPointPlot3D[{Callout[o, "Γ"], Callout[b1 / 2 + b2 / 2, "M", Right],
   Callout[b1 / 2 + b2 / 2 + b3 / 2, "R"], Callout[b1 / 2, "X", Right]},
  PlotStyle → {Blue, PointSize[Large]}];
```

In[∘]:=
```
Show[WScell, points]
```

Out[∘]=



Fcc lattice

Now, same story with a fcc lattice.
Primitive vectors in the Bravais lattice

In[∘]:=
```
a1 = a / 2 {1, 1, 0};
a2 = a / 2 {1, 0, 1};
a3 = a / 2 {0, 1, 1};
```

Calculate the primitive vectors of the reciprocal lattice

$\mathbf{b}_1 = 2\pi \frac{\boldsymbol{a}_2 \times \boldsymbol{a}_3}{\boldsymbol{a}_1 \cdot (\boldsymbol{a}_2 \times \boldsymbol{a}_3)}$

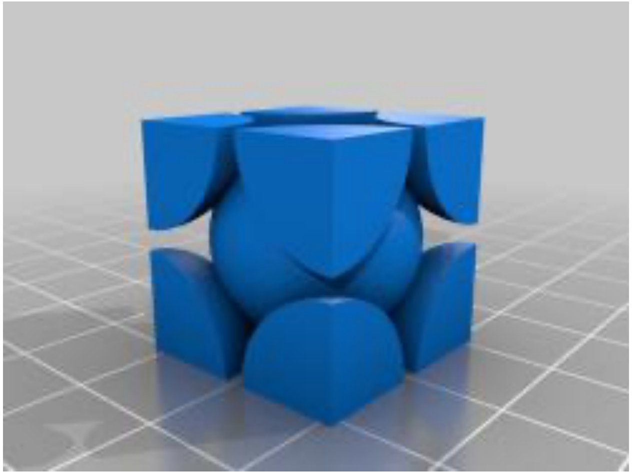
$\mathbf{b}_2 = 2\pi \frac{\boldsymbol{a}_3 \times \boldsymbol{a}_1}{\boldsymbol{a}_2 \cdot (\boldsymbol{a}_3 \times \boldsymbol{a}_1)}$

$\mathbf{b}_3 = 2\pi \frac{\boldsymbol{a}_1 \times \boldsymbol{a}_2}{\boldsymbol{a}_3 \cdot (\boldsymbol{a}_1 \times \boldsymbol{a}_2)}$

In[∘]:=
```
c = 2 π / a;
```

In[∘]:=
```
b1 = c {1, 1, -1};
b2 = c {1, -1, 1};
b3 = c {-1, 1, 1};
```

then the operations will be exactly the same as for the simple cubic lattice, but... the reciprocal lattice of a fcc cell is a bcc cell!



In rigid spheres approximation, there are **14** nearest neighbours:

- 8 atoms at the vertices of the cube
- 6 atoms at the center of the adjacent cubic cells.

Plot origin and first neighbours

```
In[]:= nodes = ListPointPlot3D[{o, o + b1, o - b1, o + b2, o - b2, o + b3, o - b3,
    o + b1 + b2 + b3, o - b1 - b2 - b3}, Filling → None, PlotStyle → Black];
halfnodes = ListPointPlot3D[{o + (b1 + b2)/1, -(b1 + b2)/1, (b1 + b3)/1, (b1 + b3)/1,
    -(b1 + b3)/1, (b3 + b2)/1, -(b3 + b2)/1}, Filling → None, PlotStyle → LightGray];
```

Plotting connecting lines between origin and first neighbours

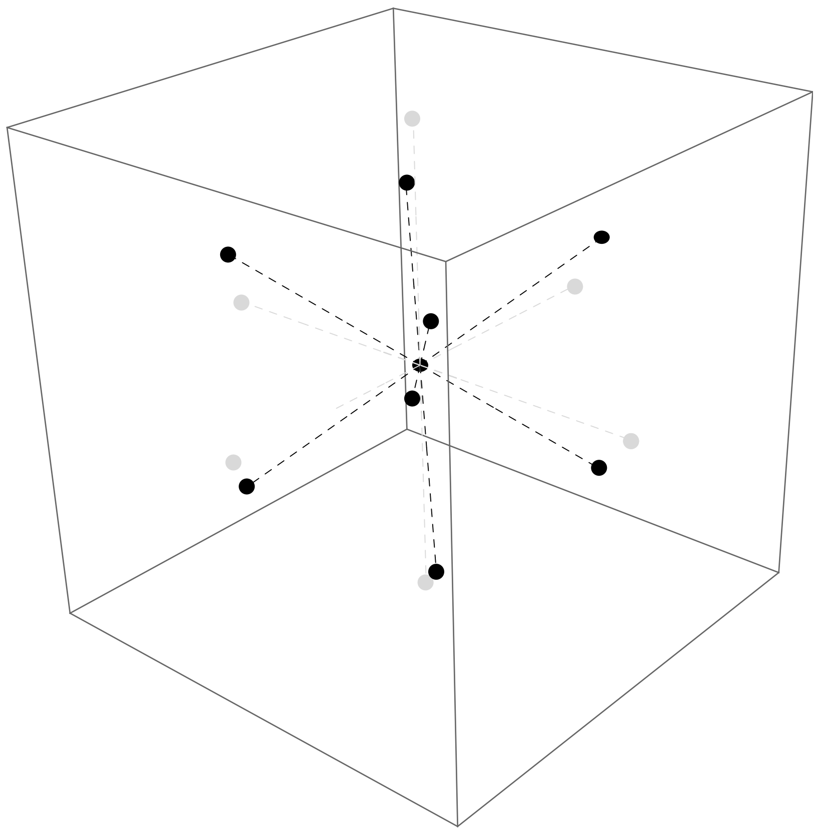
Vertices

```
In[]:= g1 = Graphics3D[{Arrowheads[.0], Dashed, Arrow[{o, b1}]}];
g2 = Graphics3D[{Arrowheads[.0], Dashed, Arrow[{o, -b1}]}];
g3 = Graphics3D[{Arrowheads[.0], Dashed, Arrow[{o, b2}]}];
g4 = Graphics3D[{Arrowheads[.0], Dashed, Arrow[{o, -b2}]}];
g5 = Graphics3D[{Arrowheads[.0], Dashed, Arrow[{o, b3}]}];
g6 = Graphics3D[{Arrowheads[.0], Dashed, Arrow[{o, -b3}]}];
g7 = Graphics3D[{Arrowheads[.0], Dashed, Arrow[{o, b3 + b2 + b1}]}];
g8 = Graphics3D[{Arrowheads[.0], Dashed, Arrow[{o, -b3 - b2 - b1}]}];
```

Center

In[∘]:=
```
h1 = Graphics3D[{Arrowheads[.0], LightGray, Dashed, Arrow[{o, (b1 + b2)/1}]}];
h2 = Graphics3D[{Arrowheads[.0], LightGray, Dashed, Arrow[{o, -(b1 + b2)/1}]}];
h3 = Graphics3D[{Arrowheads[.0], LightGray, Dashed, Arrow[{o, (b1 + b3)/1}]}];
h4 = Graphics3D[{Arrowheads[.0], LightGray, Dashed, Arrow[{o, -(b1 + b3)/1}]}];
h5 = Graphics3D[{Arrowheads[.0], LightGray, Dashed, Arrow[{o, (b3 + b2)/2}]}];
h6 = Graphics3D[{Arrowheads[.0], LightGray, Dashed, Arrow[{o, -(b3 + b2)/1}]}];
```

In[∘]:=
```
first = Show[g1, g2, g3, g4, g5, g6, g7, g8, h1, h2, h3, h4,
  h5, h6, nodes, halfnodes, PlotRange -> Full, BoxRatios → {1, 1, 1}]
```

Out[∘]=


Here, I modify the functions to plot planes. I need to select the range of each parameter to plot only the portion of plane in the region of interest. 2 DOF, each with min and max value for the plot. The mathematical equation is left unchanged

In[∘]:=
```
plane0[normal_, point_, range1_, range2_, range3_, range4_] :=
 Plot3D[(normal.point - normal[[1]] x - normal[[2]] y)/normal[[3]], {x, range1, range2},
  {y, range3, range4}, Mesh → None, PlotStyle → {Gray, Opacity[0.1]}]
```

if normal vector has y coordinate ≠ 0

```
In[•]:= plane1[normal_, point_, range1_, range2_, range3_, range4_] :=
  ParametricPlot3D[{x, (normal.point - normal[[1]] x)/normal[[2]], z}, {x, range1, range2},
   {z, range3, range4}, Mesh → None, PlotStyle → {Gray, Opacity[0.1]}]
```

if normal vector has x coordinate ≠ 0

```
In[•]:= plane2[normal_, point_, range1_, range2_, range3_, range4_] :=
  ParametricPlot3D[{(normal.point - normal[[2]] y)/normal[[1]], y, z}, {y, range1, range2},
   {z, range3, range4}, Mesh → None, PlotStyle → {Gray, Opacity[0.1]}]
```

Put together the cases to define the function

```
In[•]:= plane[normal_, point_, range1_, range2_, range3_, range4_] :=
  If[normal[[3]] ≠ 0, plane0[normal, point, range1, range2, range3, range4],
   If[normal[[2]] ≠ 0, plane1[normal, point, range1, range2, range3, range4],
    plane2[normal, point, range1, range2, range3, range4]]];
```

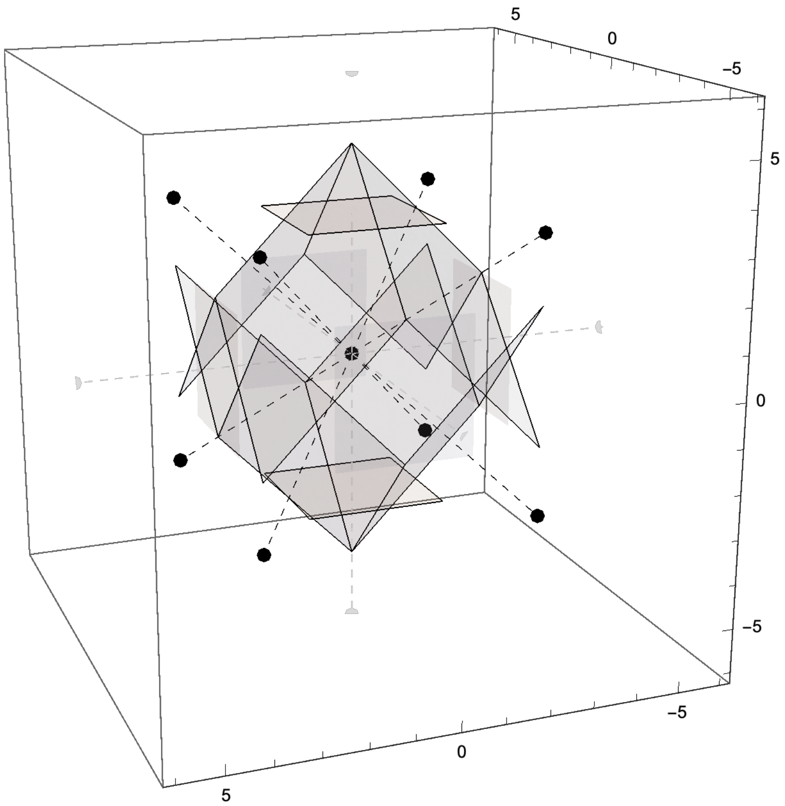
Plot planes: perpendicular to primitive vectors $\pm\boldsymbol{b_1}$,$\pm\boldsymbol{b_2}$,$\pm\boldsymbol{b_3}$. The Wigner-Seits is defined by the vectors having modulus smaller than the half distance to the center of the next cell

```
In[•]:= p01 = plane[b1, b1 / 2, 0, c, 0, c];
p10 = plane[-b1, -b1 / 2, 0, -c, 0, -c];
p02 = plane[b2, b2 / 2, 0, c, -c, 0];
p20 = plane[-b2, -b2 / 2, -c, 0, 0, c];
p03 = plane[b3, b3 / 2, -c, 0, 0, c];
p30 = plane[-b3, -b3 / 2, 0, c, -c, 0];
p04 = plane[b1 + b2 + b3, (b1 + b2 + b3) / 2, 0, c, 0, c];
p40 = plane[-(b1 + b2 + b3), -(b1 + b2 + b3) / 2, 0, -c, 0, -c];
```

Plot planes: perpendicular to the vectors connecting the center of the adjacent cells:

In[•]:=
```
p50 = plane[(b1 + b2), (b1 + b2) / 2, -c / 2, c / 2, -c / 2, c / 2];
p05 = plane[-(b1 + b2), -(b1 + b2) / 2, -c / 2, c / 2, -c / 2, c / 2];
p60 = plane[(b2 + b3), (b2 + b3) / 2, -c / 2, c / 2, -c / 2, c / 2];
p06 = plane[-(b2 + b3), -(b2 + b3) / 2, -c / 2, c / 2, -c / 2, c / 2];
p70 = plane[(b1 + b3), (b1 + b3) / 2, -c / 2, c / 2, -c / 2, c / 2];
p07 = plane[-(b1 + b3), -(b1 + b3) / 2, -c / 2, c / 2, -c / 2, c / 2];
WScell = Show[p01, p10, p02, p20, p03, p30, p04,
  p40, p50, p05, p60, p06, p70, p07, first, AspectRatio → 1,
  PlotRange → {{-3 a, 3 a}, {-3 a, 3 a}, {-3 a, 3 a}}, BoxRatios → {1, 1, 1}]
```

Out[•]=



Gamma: **0**

M: center of an edge

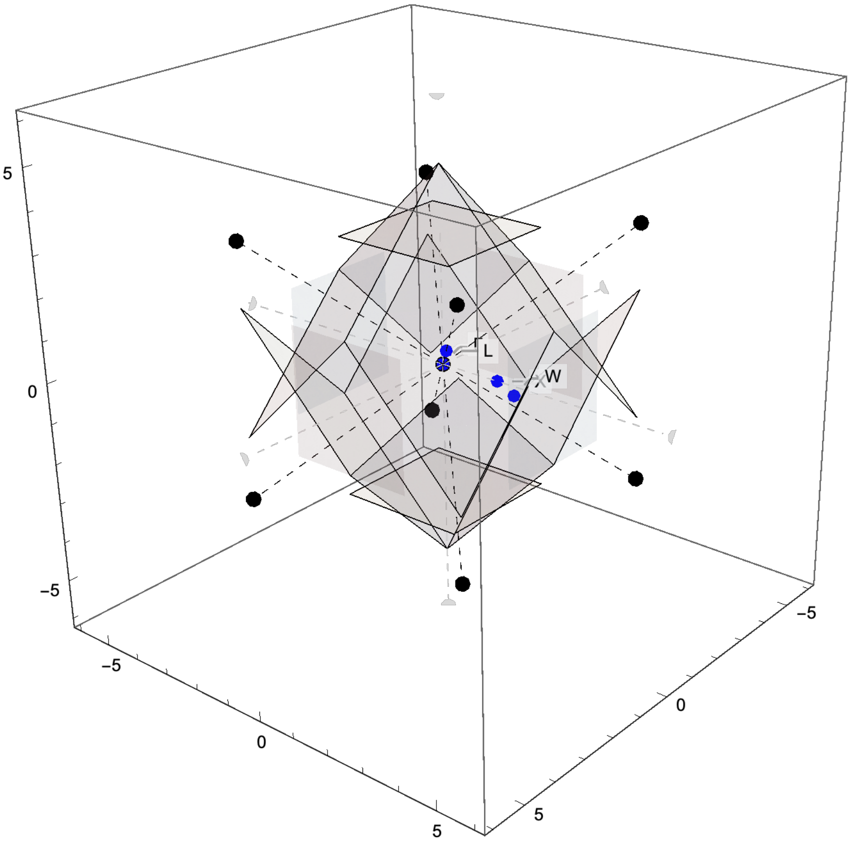
R: vertex

X: center of a face

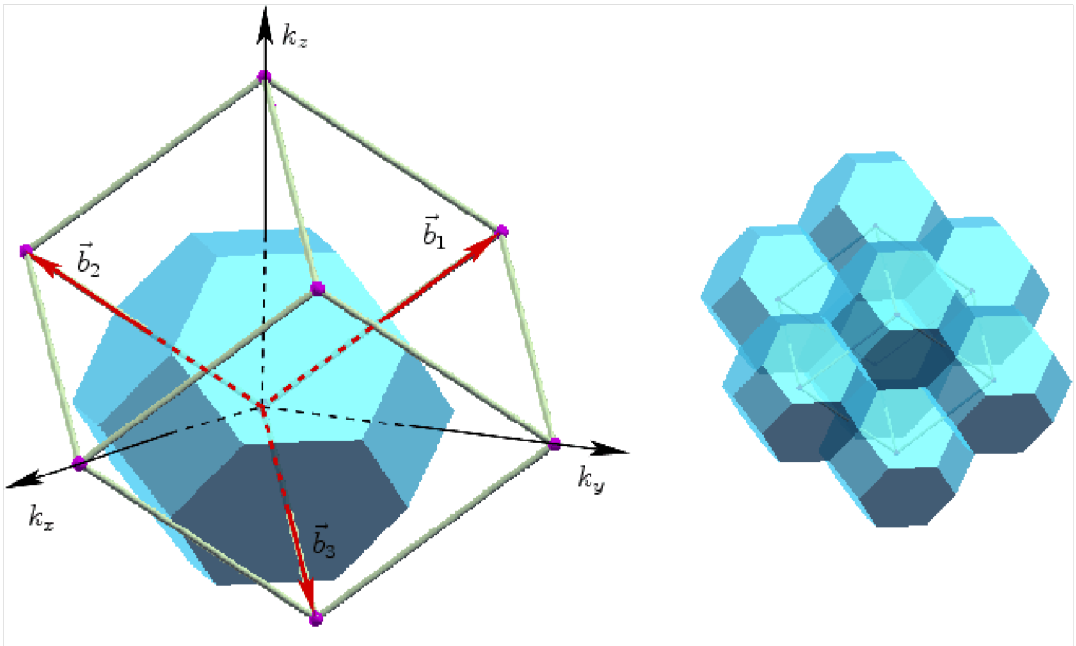
In[•]:=
```
points =
  ListPointPlot3D[{Callout[o, "Γ"], Callout[π / a {1 / 2, 1 / 2, 1 / 2}, "L", Right],
    Callout[π / a {1 / 4, 3 / 2, 0}, "W"], Callout[{0, π / a, 0}, "X", Right]},
   PlotStyle → {Blue, PointSize[Large]}, BoxRatios → {1, 1, 1}];
```

In[•]:=
```
Show[WScell, points]
```

Out[•]=



An artist impression is still better: truncated octahedron



(a) Reciprocal lattice cell with first Brillouin zone. (b) Brillouin zone cells on every node.

Points of interest:

Gamma: center of the cell: **Γ**=$\frac{\pi}{a}(0,0,0)$

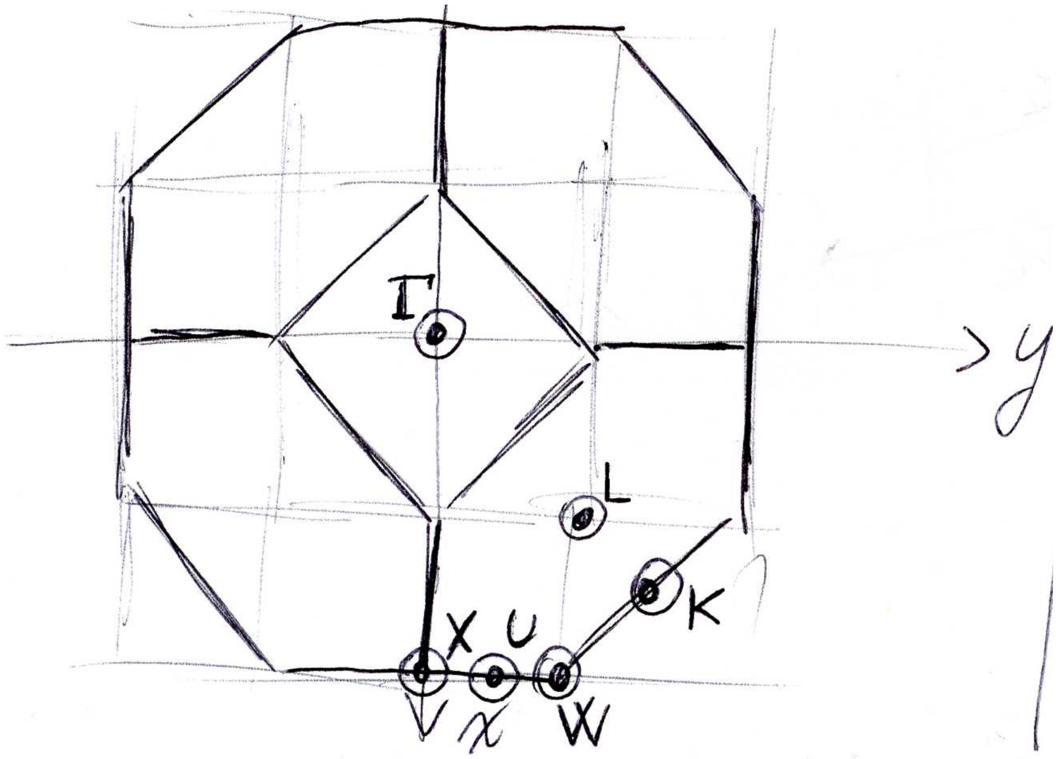
X: center of the square face **X**=$\frac{\pi}{a}(1,0,0)$

X: center of the hexagonal face **L**=$\frac{\pi}{a}(\frac{1}{2},\frac{1}{2},\frac{1}{2})$

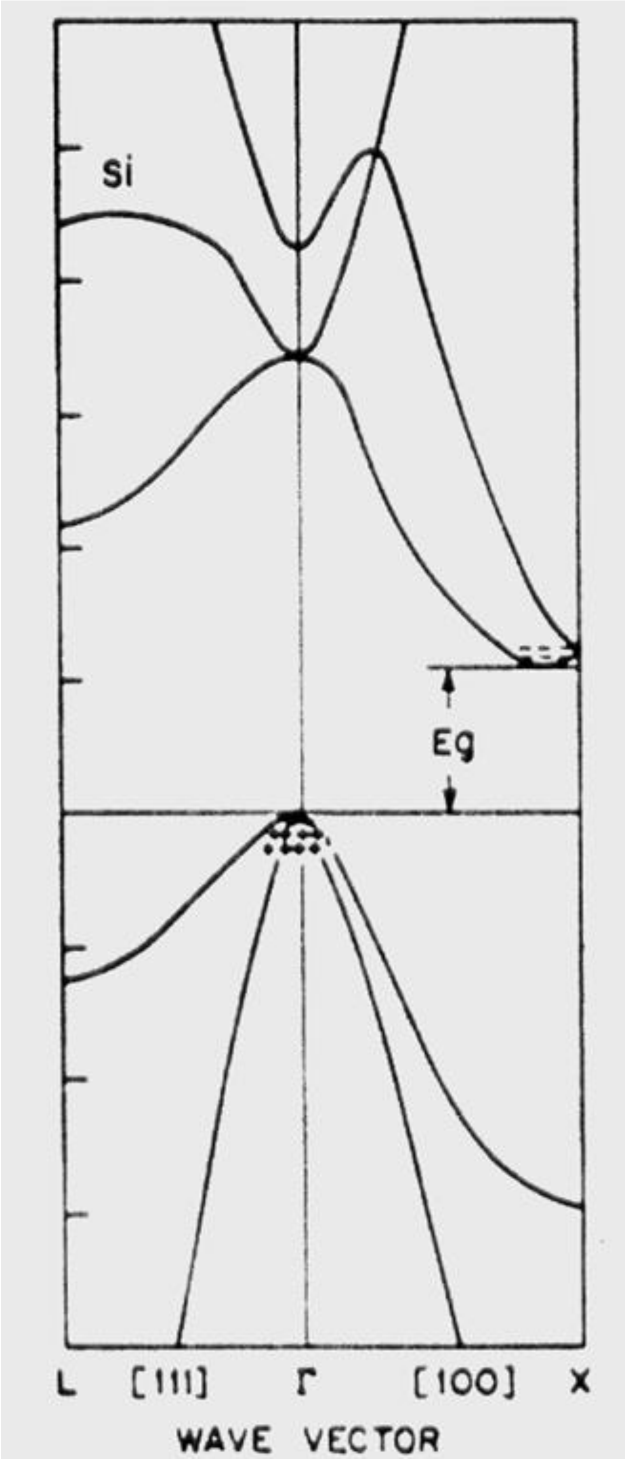
W: vertex $\mathbf{W}=\frac{\pi}{a}(1,\frac{1}{2},0)$

U: center of a square edge $\mathbf{U}=\frac{\pi}{a}(1,\frac{1}{4},\frac{1}{4})$

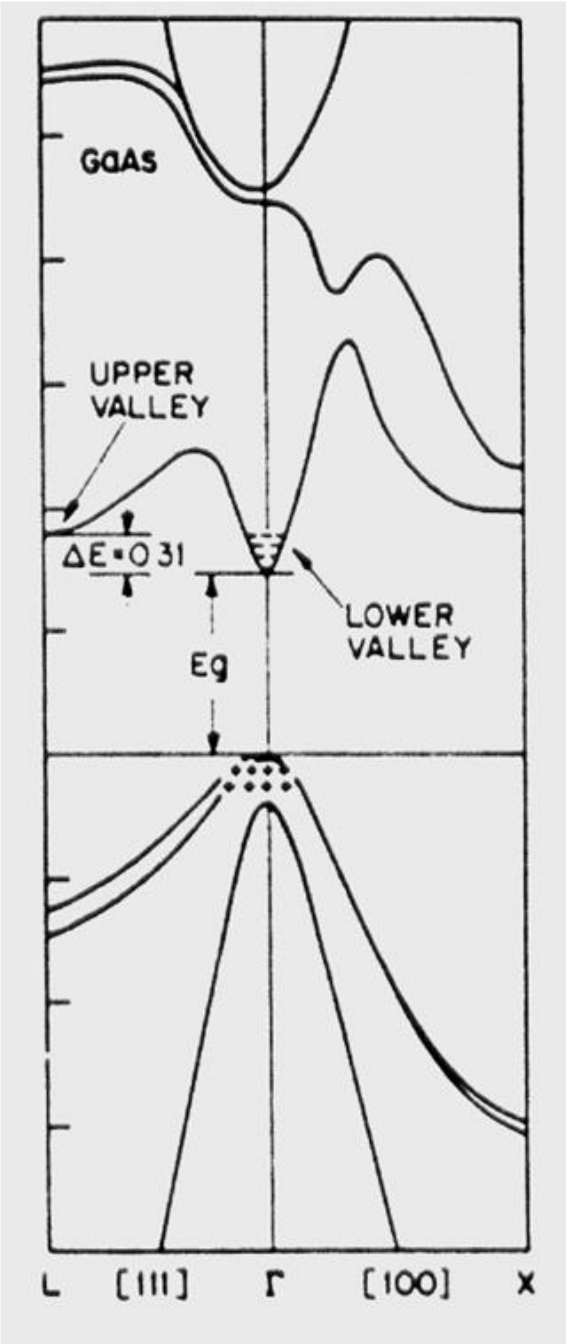
K: center of a hexagon edge $\mathbf{K}=\frac{\pi}{a}(\frac{3}{4},\frac{3}{4},0)$



Lecture notes here



Lecture notes here


GaAS
UPPER
VALLEY
ΔE=0.31
Eg
LOWER
VALLEY
L
[111]
Γ
[100]
X