Homework 1 - Due 2/13/2013

John Kitchin

2013-02-04 Mon

Contents

1 Compute the distance between the (111) planes of Ag.

Use the experimental lattice constant of Ag.

1.1 solution :solution:

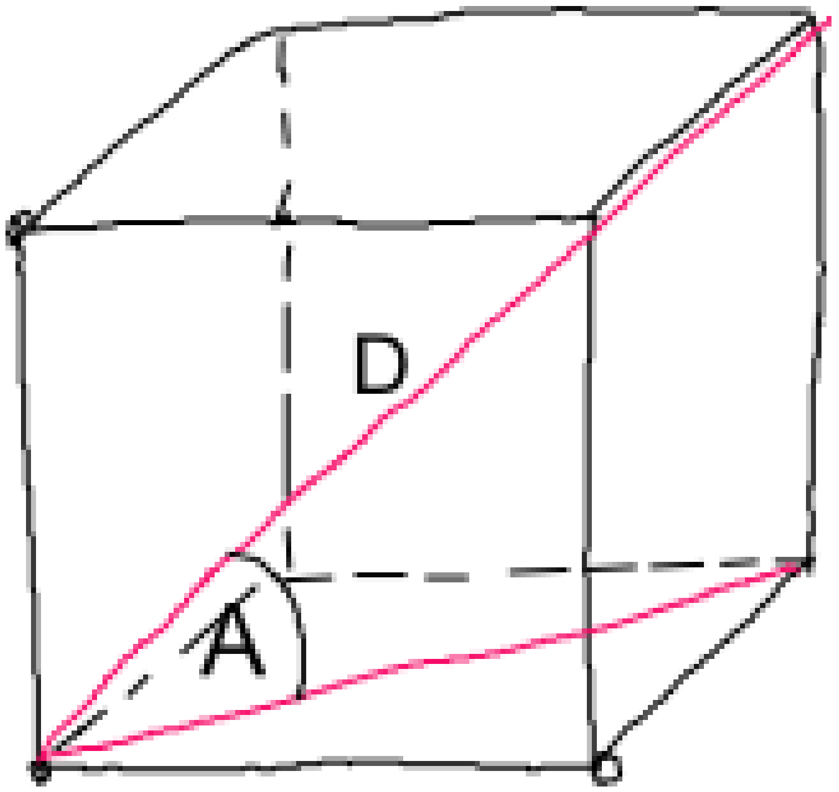
We use the reciprocal metric tensor to directly compute the spacings. It does not matter which unit cell you choose.

```python
import numpy as np

a = 4.09

uc = np.array([[a, 0, 0],
               [0, a, 0],
              [0, 0, a]])

ucstar = np.linalg.inv(uc).T

gstar = np.dot(ucstar, ucstar.T)

v = [1, 1, 1]
d_111 = np.sqrt(np.dot(v, np.dot(gstar, v)))

print 'The spacing between the (111) planes is {0} angstroms.'.format(1.0 / d_111)

# if you use the primitive cell
b = a/2.0
uc = np.array([[b, b, 0],
              [b, 0, b],
              [0, b, b]])

ucstar = np.linalg.inv(uc).T

gstar = np.dot(ucstar, ucstar.T)

v = [1, 1, 1]
d_111 = np.sqrt(np.dot(v, np.dot(gstar, v)))

print 'The spacing between the (111) planes is {0} angstroms.'.format(1.0 / d_111)
```

```
The spacing between the (111) planes is 2.36136260099 angstroms.
The spacing between the (111) planes is 2.36136260099 angstroms.
```

2 Angles and distances

Compute the length of the vector indicated by D, and the angle indicated by A. You can assume the unit cell is a cube with length of 3.6 nm on each side.



2.1 solution :solution:

The most general approach is to define the unit cell, and then define the vector in the unit cell coordinate system, and compute the distance.

```
import numpy as np

a= 3.6
uc = np.array([[a, 0, 0],
               [0, a, 0],
               [0, 0, a]])

g = np.dot(uc, uc.T)

P = [1, 1, 1]

d = np.sqrt(np.dot(np.dot(P, g), P))
print d
```

```
6.23538290725
```

For the angle,

```
import numpy as np

```

```python
a= 3.6
uc = np.array([[a, 0, 0],
               [0, a, 0],
               [0, 0, a]])

g = np.dot(uc, uc.T)

P = [1, 1, 0]
Q = [1, 1, 1]

lp = np.sqrt(np.dot(P, np.dot(g, P)))
lq = np.sqrt(np.dot(Q, np.dot(g, Q)))

print lp, lq

theta = np.arccos(np.dot(P, np.dot(g, Q))/(lp * lq))
print theta * 180.0 / np.pi
```

```
5.09116882454 6.23538290725
35.2643896828
54.7356103172
```

3 Coordinate system transformations

The unit cell of a material is given by:

```
4.2266540199664249    0.0000000000000000    0.0000000000000000
0.0000000000000000    4.2266540199664249    0.0000000000000000
0.0000000000000000    0.0000000000000000    2.6888359272289208
```

The coordinates of the atoms in the unit cell are given in fractional coordinates as:

```
0.0000000000000000  0.0000000000000000  0.0000000000000000
0.5000000000000000  0.5000000000000000  0.5000000000000000
0.3067891334429594  0.3067891334429594  0.0000000000000000
0.6932108665570406  0.6932108665570406  0.0000000000000000
0.1932108665570406  0.8067891334429594  0.5000000000000000
0.8067891334429594  0.1932108665570406  0.5000000000000000
```

Compute the Cartesian coordinates of each atom.

3.1 Solution :solution:

We simply have to compute for each position (s1, s2, s3) the vector sum of s1*A + s2*B + s3*C where A, B, C are the unit cell vectors. We have a matrix where row 1 is A, row 2 is B, and row 3 is C.

Let this be our matrix:

$$M = \begin{array}{ccc} a1 & a2 & a3 \\ b1 & b2 & b3 \\ c1 & c2 & c3 \end{array}$$

We end up with the following equations:

p_x = s1*a1 + s2*b1 + s3*c1 p_y = s1*a2 + s2*b2 + s3*c2 p_z = s1*a3 + s2*b3 + s3*c3

or in matrix form:

$$\begin{bmatrix} s1 & s2 & s3 \end{bmatrix} \cdot \begin{bmatrix} a1 & a2 & a3 \\ b1 & b2 & b3 \\ c1 & c2 & c3 \end{bmatrix} = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}$$

```
import numpy as np

uc = np.array([[4.2266540199664249,    0.0000000000000000,    0.0000000000000000],
               [0.0000000000000000,    4.2266540199664249,    0.0000000000000000],
               [0.0000000000000000,    0.0000000000000000,    2.6888359272289208]])


sp = np.array([[  0.0000000000000000,  0.0000000000000000,  0.0000000000000000],
               [  0.5000000000000000,  0.5000000000000000,  0.5000000000000000],
               [  0.3067891334429594,  0.3067891334429594,  0.0000000000000000],
               [  0.6932108665570406,  0.6932108665570406,  0.0000000000000000],
               [  0.1932108665570406,  0.8067891334429594,  0.5000000000000000],
               [  0.8067891334429594,  0.1932108665570406,  0.5000000000000000]])

cp = np.dot(sp, uc)

print cp



print 'Alternative approach'
# alternate approach
for row in sp:
    print uc[0]*row[0] + uc[1]*row[1] + uc[2]*row[2]
#print uc[0] * sp[:,0] + uc[1] * sp[:,1] + uc[2] * sp[:,2]
```

```
[[ 0.          0.          0.        ]
 [ 2.11332701  2.11332701  1.34441796]
 [ 1.29669152  1.29669152  0.        ]
 [ 2.9299625   2.9299625   0.        ]
 [ 0.81663549  3.41001853  1.34441796]
 [ 3.41001853  0.81663549  1.34441796]]
Alternative approach
[ 0.  0.  0.]
[ 2.11332701  2.11332701  1.34441796]
[ 1.29669152  1.29669152  0.        ]
[ 2.9299625  2.9299625  0.        ]
[ 0.81663549  3.41001853  1.34441796]
[ 3.41001853  0.81663549  1.34441796]
```

4 Computing unit cell parameters

For the following unit cell (in rows), compute the length of each vector, and the angle between the vectors, e.g. the “a b c α β γ” representation.

```
3.817   -0.011  -0.243
-0.011  3.817   -0.243
-1.519  -1.519  4.986
```

4.1 solution :solution:

```
import numpy as np
from Scientific.Geometry import Vector
A = Vector([3.817,      -0.011, -0.243])
B = Vector([-0.011,     3.817,  -0.243])
C = Vector([-1.519,     -1.519, 4.986])

a = A.length()
b = B.length()
c = C.length()
alpha = B.angle(C) * 180.0 / np.pi
beta = A.angle(C) * 180.0 / np.pi
gamma = A.angle(B) * 180.0 / np.pi

print 'a={a:1.2f} b={b:1.2f} c={c:1.2f} alpha={alpha:1.2f} deg beta={beta:1.2f} deg gamma={gamma:1.2f} deg'.format(**locals())
```

```
a=3.82 b=3.82 c=5.43 alpha=109.68 deg beta=109.68 deg gamma=90.10 deg
```

5 Computing fractional coordinates

Given this unit cell (in rows)

```
3.817   -0.011  -0.243
-0.011  3.817   -0.243
-1.519  -1.519  4.986
```

And atoms at these cartesian coordinates:

```
[[ 1.23141,   0.239958,  3.102345]
 [ 1.05559,   2.047042,  1.397655]
 [-0.626458,  2.21009,   3.890655]
 [ 2.913458,  0.07691,   0.609345]
 [ 1.004314,  0.139186,  1.125   ]
 [ 1.282686,  2.147814,  3.375   ]]
```

Compute the fractional coordinates of each atom in the unit cell.

5.1 solution :solution:

```
import numpy as np

uc = np.array([[3.817,  -0.011, -0.243],
               [-0.011, 3.817,  -0.243],
               [-1.519, -1.519, 4.986]])

cp = np.array([[ 1.23141,   0.239958,  3.102345],
                 [ 1.05559,   2.047042,  1.397655],
                 [-0.626458,  2.21009,   3.890655],
                 [ 2.913458,  0.07691,   0.609345],
                 [ 1.004314,  0.139186,  1.125   ],
                 [ 1.282686,  2.147814,  3.375   ]])

print np.dot(np.linalg.inv(uc.T), cp.T).T

print 'alternative, and equivalent linear algebra'
print np.dot(cp, np.linalg.inv(uc))
```

```
[[ 0.589  0.33   0.667]
 [ 0.411  0.67   0.333]
 [ 0.17   0.911  0.833]
 [ 0.83   0.089  0.167]
 [ 0.363  0.137  0.25 ]
 [ 0.637  0.863  0.75 ]]
alternative, and equivalent linear algebra
[[ 0.589  0.33   0.667]
 [ 0.411  0.67   0.333]
 [ 0.17   0.911  0.833]
 [ 0.83   0.089  0.167]
 [ 0.363  0.137  0.25 ]
 [ 0.637  0.863  0.75 ]]
```