

Surfaces and Adsorption

John Kitchin and Andrew Gellman

2013-01-13 Sun

Contents

1	About this book	3
1.1	Python	3
2	Introduction	3
2.1	The Importance of Solid Surfaces	3
2.2	Historical Development	4
2.3	Modern Surface Science	5
2.4	Surface Sensitivity	5
2.5	Surface Cleanliness	7
3	Structure of Solid Surfaces	8
3.1	Symmetry	10
3.1.1	Point Symmetry	11
3.1.2	Symmetry in 1D	12
3.1.3	Symmetry in 2D	14
3.1.4	Symmetry in 3D	20
3.2	Surfaces of the Simple Cubic Lattice	21
3.3	Miller Indices	23
3.3.1	Miller Index Formulation	25
3.4	Common Low Miller Index Surface Structures	27
3.4.1	The most common fcc surfaces	27
3.4.2	The most common bcc surfaces	29
3.4.3	hcp surfaces	30
3.5	High Miller Index Surfaces	31
3.6	Surface Relaxation	32
3.7	Reconstruction	33
4	Computational crystallography	34
4.1	Computing distances between two points	35
4.2	Computing the angle between two vectors	36
4.3	Computing unit cell volumes	38
4.4	The reciprocal lattice in crystallography	39
4.5	Coordinate system transformations	41

5	Low Energy Electron Diffraction (LEED)	42
5.1	Diffraction	43
5.2	The reciprocal lattice	46
5.3	The LEED Pattern	48
5.3.1	Ewald Sphere	48
5.3.2	LEED Patterns from Simple Clean Surfaces	50
5.4	Overlayers and Superlattices	54
5.4.1	Overlayer Notations	54
5.4.2	LEED Patterns from superlattices	60
5.5	General Reciprocal Space to Real Space Conversion	62
5.5.1	Example - LEED of $(\sqrt{2} \times \sqrt{2})R45^\circ$ on an fcc(100) lattice.	63
5.6	Domains	64
5.6.1	Example - (2×1) overlayer on a square fcc(100) lattice.	65
5.6.2	Example - $(\sqrt{5} \times \sqrt{5})R26^\circ$ on square lattice. Oxygen on Mo(100) at $\theta = 0.6$	65
6	Thermal desorption methods	67
6.1	TPD - Reversible Adsorption / Desorption	67
6.2	TPRS – Irreversible Surface Reactions	68
6.3	TPD Experiment	69
6.4	Analysis of First-Order Desorption Kinetics	72
6.4.1	Examples of first-order desorption curves.	74
6.5	Zero order Desorption Kinetics	75
6.6	Second-Order Desorption Kinetics	78
6.7	Assumptions in TPD Analysis	79
7	Worked examples	79
7.1	Analysis of XPS spectra	79
7.2	BET surface area	87
7.3	Summary notes	91
8	References	91
9	GNU Free Documentation License	91

Copyright ©2013–2013 John Kitchin, Andrew Gellman

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

1 About this book

This book intends to provide a freely available resource on concepts in surface science. You are free to copy this work, redistribute, even print and sell the work, provided you adhere to the terms in the [license](#).

The book is a work in progress and is being used to teach a course titled “Surfaces and adsorption”.

We have endeavored to avoid any violations of copyright law in the creation of this book, and we believe that all material used in this book falls under “fair use” as allowed by copyright law. If you are a copyright holder of material used in this book and you believe we have infringed upon your rights, please contact John Kitchin at jkitchin@andrew.cmu.edu to discuss the issue. It is not our intention to violate the law.

1.1 Python

The book uses Python wherever numerical analysis is required. Python is similar in nature to Matlab, but is freely available. We have attempted to introduce the language by examples throughout the book. If you are a new user, you should start at the beginning of the book. If you are experienced with Matlab, the syntax should be easy to read. We recommend the following Python distribution for use with this book.

- Enthought Python Distribution (<http://www.enthought.com/products/epd.php>)

This package is free for academic use, and available for Windows, Macs and Linux. The package includes all the typical python libraries needed for numerical, scientific and graphics computing.

- An alternative python environment that may be suitable is Python(x,y) (<http://code.google.com/p/pythonxy/>). This distribution is Windows focused, and there are not Mac or Linux installers available.
- For editing/writing python code, I like the Spyder editor (<http://code.google.com/p/spyderlib/>). It is also available for Windows, Macs and Linux. You may find the IDLE or SciTE editor that comes with Enthought suitable though.

2 Introduction

2.1 The Importance of Solid Surfaces

- We have never seen anything but the surface of an object.
- Catalysis. Over 90% of all commodity chemical are produced or processed through the use of heterogeneous catalysts. These catalysts are: dispersed metal particles, high surface area zeolites, finely divided oxide powders.

- Corrosion. Destructive oxidation of surfaces or etching for control of surface finishes.
- Brittle fracture. Fracture of solids is often due to segregation of foreign materials to grain boundaries.
- Thermionic emission. Electron emission from heated filaments in TVs (old), electronic tubes etc. Rate of emission depends on surface properties.
- Crystal growth. Growth from solution or from the vapor phase depends on reactions on surfaces and on diffusion on surfaces.
- Semiconductor properties and processing. As the size of devices decreases the surface-to-volume ratio increases and surfaces begin to have an important influence on physical properties.
- Nanophase materials. Solid materials with grains of nanometer dimensions have extremely high grain boundary densities and extraordinary properties.

2.2 Historical Development

- Solids were found to cause reactions. Priestley (1775) $\text{CH}_3\text{CH}_2\text{OH}$ decomposition on Cu. Davy (1817) CO and H_2 oxidation on Pt. Miner's lamp.
- Reactivity increased with porosity. One idea was that the surfaces compress gasses in pores and cause reaction. This was debunked by the fact that porous metal surfaces differ in reactivity.
- Van't Hoff and Sabatier show that surfaces affect the rate but not the equilibrium constant of a reaction. This is a major milestone in the development of chemical thermodynamics. Demonstrates that the equilibrium constant is path independent.
- Several catalytic processes are developed for commerce.
 - Messel (1875) $\text{SO}_2 + \frac{1}{2} \text{O}_2 \rightarrow \text{SO}_3$
 - Mond (1888) $\text{CH}_4 + \frac{1}{2} \text{O}_2 \rightarrow \text{CO} + 2 \text{H}_2$
 - Sabatier (1902) $\text{C}_2\text{H}_4 + \text{H}_2 \rightarrow \text{C}_2\text{H}_6$
 - Haber (1905) $\text{N}_2 + 3 \text{H}_2 \rightarrow 2 \text{NH}_3$
- Langmuir (1915) works on the development of long-life light bulbs for GE and studies the adsorption of gases on hot filaments.
- Davisson and Germer (1927) observe the diffraction of electron from the surface of a Ni crystal and demonstrate that this is due to the wave nature of the electron. Quantum mechanics is proved!

- Modern surface science is born in the 1960's as an outgrowth of space science and the development of instrumentation for achieving ultra-high vacuum (10^{-10} Torr) environments.

2.3 Modern Surface Science

- Atomistic level study of surface imposes extremely stringent demands on experimental methods.
- The total amount of material at the surface of a solid is extremely small. 10^{15} atoms per cm^2 or 10^{-9} moles.
- The surface must be analyzed in the presence of a bulk solid whose contribution to any measurement could swamp that of the surface.

2.4 Surface Sensitivity

- Surface sensitivity must be achieved in order to avoid studying the bulk of a solid rather than the surface of interest.
- Electrons and ions interact very strongly with matter and so they cannot penetrate or escape from the bulk of a solid. In scattering or emission experiments they only sample the surface.

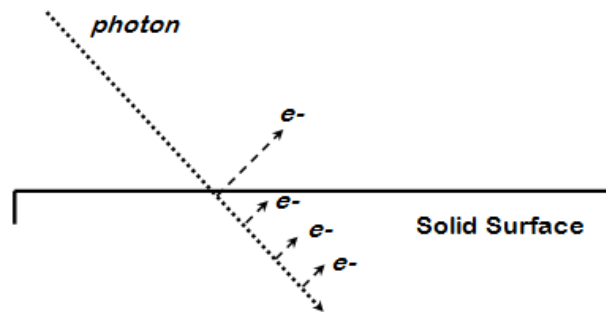


Figure 1: The XPS experiment with electrons coming from the surface only. X-rays penetrate the surface but electrons photomitted from the bulk cannot escape.

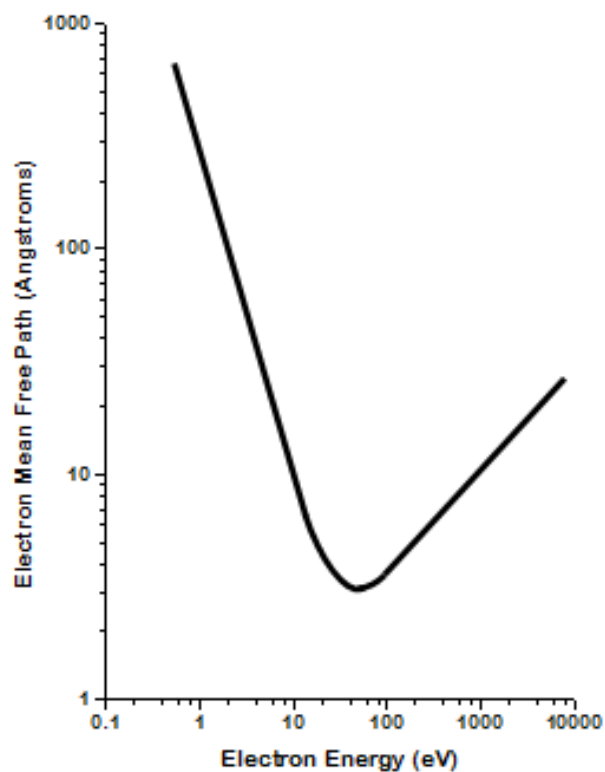


Figure 2: The universal curve of electron mean free paths in solids. The mean free path is the mean distance traveled before an electron is scattered by an atom. This curve has been obtained from measurements made with many materials.

- Ions are even more surface sensitive than electrons. Low energy ions (less than 100 eV) do not penetrate the bulk at all.
- Surface sensitive spectroscopies can almost always be classified into one of four types.
 - ion (or e^-) in \rightarrow ion (or e^-) out
 - ion (or e^-) in \rightarrow photon out
 - photon in \rightarrow ion (or e^-) out
 - photon in \rightarrow photon out

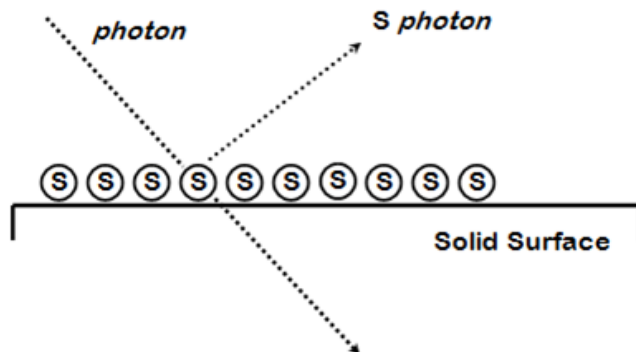


Figure 3: Photon in \rightarrow photon out only detecting sulfur atoms on a surface. If there were high concentrations of sulfur in the bulk then the bulk signal would swamp the signal from the surface atoms.

2.5 Surface Cleanliness

- During the course of an experiment (which may be many minutes to many hours) the state of the surface must remain stable (clean or otherwise). This means that if one studies a clean surface it must not become contaminated by collisions with gas phase molecules.
- Consider the flux of molecules colliding with a surface.

$$F = \frac{1}{4} N_g \langle \nu \rangle$$

Where F is the flux (molecules/m²/s), N_g is the gas molecular density, (molecules/m³), and $\langle \nu \rangle$ is the mean molecular speed of the gas (m/s).

The mean molecular speed is given by kinetic theory:

$$\langle \nu \rangle = \sqrt{\frac{8kT}{\pi m}}$$

Where k is the Boltzman constant (1.38×10^{-23} J/K), T is the absolute temperature, and m is the molar mass (kg).

The ideal gas law gives the density.

$$N_g = \frac{n}{V} = \frac{P}{kT}$$

where P is the pressure, n is moles of gas in the volume.

Putting this together we finally have the flux as

$$F = \frac{P}{\sqrt{2\pi m k T}}$$

Let us consider an example at $P = 1$ bar, $T = 300$ K, and $m = 30$ amu. We will show how to do this in Python.¹

¹We have to import the numpy library. Here we import it with the name np, and then access the functions using a dot notation. For example, the sqrt function is np.sqrt.

```

1  1: import numpy as np
2
3  3: P = 1.0          # bar
4  4: T = 300.0        # K
5  5: m = 30.0         # amu
6  6: kb = 1.3807e-23  # J/K
7
8  8: # conversion factors
9  9: amu2kg = 1.660538921e-27
10 10: bar2Pa = 100000.0
11 11: m2cm = 100.0
12 12:
13 13: F = (P * bar2Pa) / (np.sqrt(2 * np.pi * m * amu2kg * kb * T))
14 14: print 'The flux = {0:1.2e} mlc/cm^2/s'.format(F / m2cm**2)

```

The flux = 2.78e+23 mlc/cm²/s

In line 14 we use the syntax {0:1.2e} to format the answer in scientific notation with two decimal places.

Note the order of magnitude, about 0.5 moles of gas hit a square centimeter, every second. A typical density of surface atoms is $\rho_s = 2.7 \times 10^{14}$ atoms/cm². The collision frequency is then given by:

$$Z_c = \frac{F}{\rho_s} \approx 10^9$$

That is, each surface atom is bombarded about 1 billion times a second! Let us assume that we could start with a clean surface, and that every molecule that hits the surface sticks. We can estimate adsorption rates from the flux then.

Pressure (bar)	adsorption rate (molecules/sec)
1	10 ⁻⁹ /sec
10 ⁻⁹	1/sec
10 ⁻¹²	1/hr

You can see that we must have pressures less than 1×10^{-13} bar to keep surfaces clean for hours at a time to do experiments. Of course not every molecule that hits sticks, so this is only an approximate analysis.

3 Structure of Solid Surfaces

In this section we review the structure of solid surfaces. Recommended readings include chapter 2 in [Masel, 1996](#) and chapter 2 in [Somorjai and Li \[2010\]](#).

- The crystalline structure of solids has been observed since antiquity.
- The existence of crystals and the fact they could be subdivided into identical crystals lead to the original ideas that matter was formed of some elementary components with characteristic shape.
- Structure is defined by the positions of all atoms of all types in a solid.

$$\{(x_i^m, y_i^m, z_i^m)\} i \in I, m - type$$

- When a crystal is cleaved or sliced it exposes a surface with atoms in some ordered array.
- Electron diffraction first proved the existence of atomic order at crystal surfaces (Davisson and Germer, 1927).
- The first direct observation of atoms was on surfaces and was made using Field Ion Microscopy (Muller and Bahadur, Oct. 11, 1955).

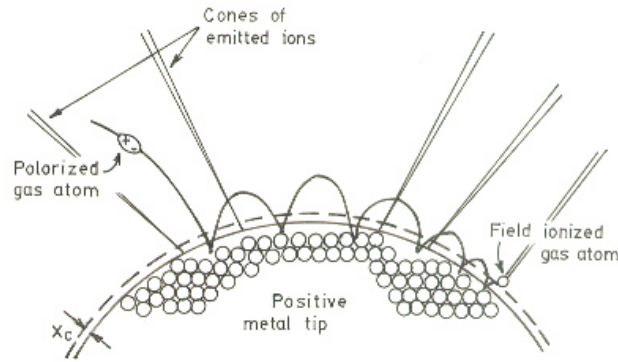


Figure 4: Field ion emission microscope. The metal tips has a high applied potential (≈ 25 kV). He atoms are ionized at the points of high field gradient (atoms at step edges and then accelerated away from the tip along the field lines. They are imaged on a phosphorus screen.

²Read this equation as the (x,y,z) coordinates of the i^{th} atom of type m for every atom i in the structure I .

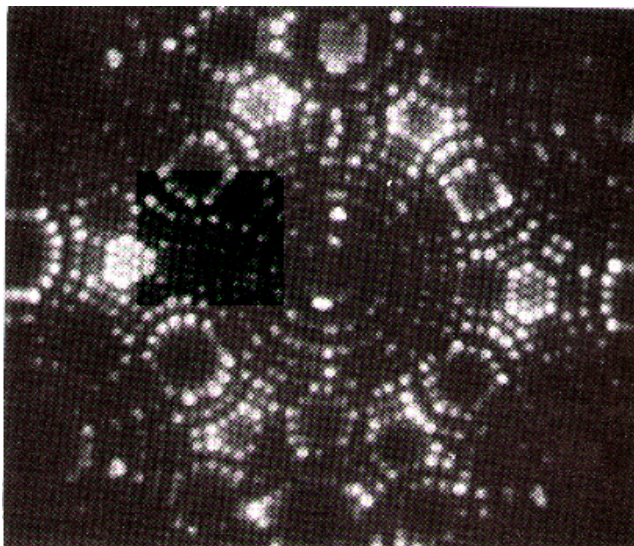
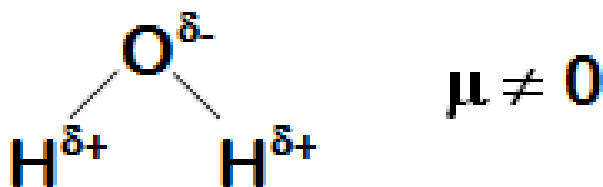


Figure 5: Field ion micrograph of a tungsten tip. The atomic structures of various planes are easily observable.

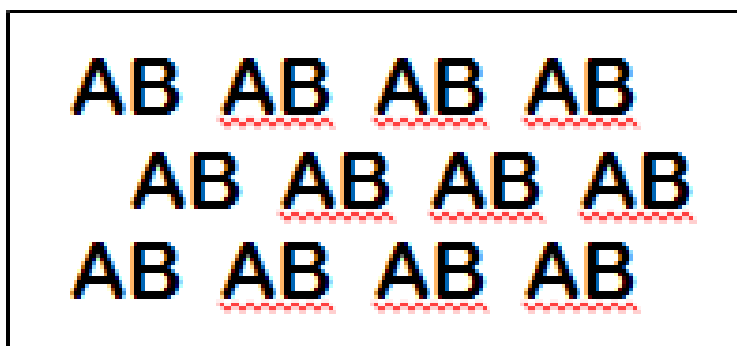
- In general we consider three types of solids and the surfaces that they expose:
 - Amorphous – no long range structure or relationship between the positions of atoms.
 - Periodic – the structure is based on a unit cell that is repeated in space through translation.
 - Quasi-periodic – the structure has long range order in the sense that there are rules that define the positions of all atoms based on the positions of a few, but there is no ‘unit cell’ and no translational repetition of the structure.
- All the surfaces that we will discuss are based on periodic structures and thus some discussion of symmetry is needed.

3.1 Symmetry

- Symmetry is one of the most powerful tools available for understanding and describing the structure of molecules and solids.
- Symmetry often dictates some of the important properties of molecules, solids, and surfaces.
- The existence of dipole moments of molecules can be determined immediately from symmetry arguments: CO_2 , H_2O . Here is an example.



- Second harmonic generation of light occurs only in crystals with no inversion symmetry.



We will see later that symmetry determines which vibrational modes are visible in a vibrational spectroscopy experiment.

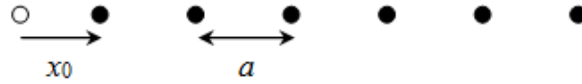
3.1.1 Point Symmetry

- Point group symmetry is used to describe isolated molecules.
- The symmetry elements that can exist in molecules are :
 - mirror planes,
 - rotational axes,
 - inversion symmetry, and
 - rotation-reflection axes.

3.1.2 Symmetry in 1D

- In extended dimensional spaces symmetry is defined by translational periodicity combined with symmetry elements.
- The constraint of translational periodicity limits the types of symmetry elements that can be considered.
- There are two elements that are needed to define the structure of a periodic solid: a lattice and a basis.
- The lattice is a set of abstract points related to one another by translational periodicity which divide space into unit cells.

1D Lattice



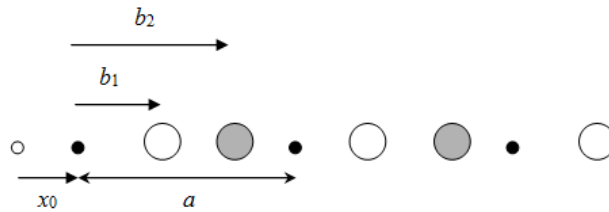
The positions of each of the points are the set $\{x_n\}$ related by

$$x_n = x_0 + n \cdot a$$

$$n \in I$$

- The basis is the set of atoms which are positioned at identical positions $\{b_i\}$ within each of the unit cells.
- The positions of atoms in the unit cell are usually given in terms of the lattice vector lengths, $b = 0 \rightarrow 1$.

Atomic basis



The position of any atom in the solid can then be given as

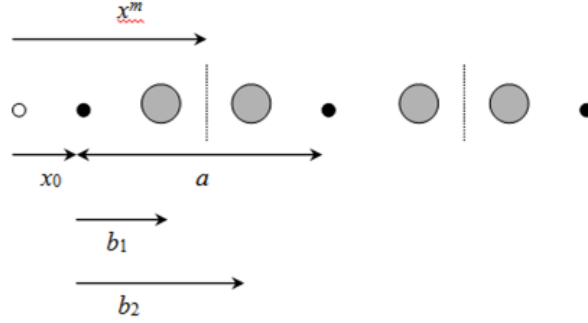
$$x_{i,n} = x_0 + (n + b_i) \cdot a$$

where i is the atom #, and n is the unit cell number.

- A symmetry element relates the position of two or more atoms (of the same type) within the unit cell to one another.

- Specifying the position of one atom and the existence of a symmetry element immediately implies the existence and dictates the position of one or more additional atoms.

Mirror plane



$$x_n^m = x_0 + a \cdot n + \frac{1}{2}(b_1 + b_2) \cdot a$$

- The mirror plane supplies a rule for relating the positions of identical points in the lattice or objects of the basis.
- A mirror plane at position x^m maps a point or atom at a position x into an identical point or atom at position $x' = x^m + (x^m - x) = 2x^m - x$.

Example : Consider the fact that we have atoms at positions $x_{1,n}$

$$x_{1,n} = x + 0 + b^m \cdot a$$

and a mirror plane at

$$x^m = x_0 + b^m \cdot a.$$

This implies the presence of a set of identical atoms at positions $x_{2,n}$.

$$x_{1,n} \implies 2x^m - x_{1,n} \quad (1)$$

$$x_{2,n} = (2x_0 + 2b^m \cdot a) - (x_0 + (n + b_1) \cdot a) \quad (2)$$

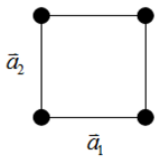
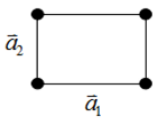
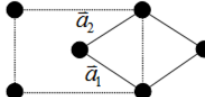
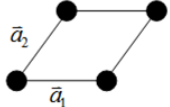
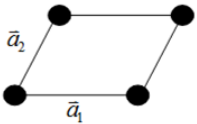
$$= x_0 + (2b^m - n - b_1) \cdot a \quad (3)$$

If the position of the mirror plane happened to be chosen such that $b^m = \frac{1}{2}$ then this would generate all the other atoms in the unit cell shown above.

In other words this generates the positions of the entire set of atoms at positions b_2 within each of the unit cells.

3.1.3 Symmetry in 2D

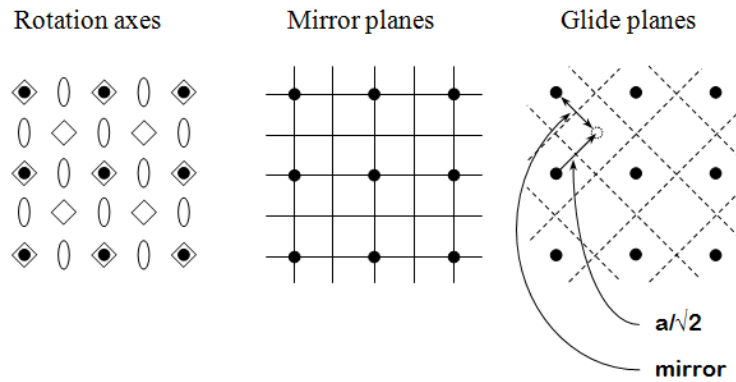
- When one slices a plane through a 3D lattice it generates a surface with the periodicity of a 2D lattice.
- Note that the surface of a real solid is not really 2D. It has some depth since it consists of atoms that are not necessarily all constrained to lie in the plane. The periodicity that we think of is actually due to the projection of the 3D semi-infinite solid onto the hypothetical plane that defines the surface.
- Also, realize that the act of cutting a 3D translationally periodic solid destroys the periodicity in the direction normal to the cut. It leaves only translational periodicity in the two directions parallel to the cutting surface.
- 2D Lattices
 - Symmetry in 2D is defined by a lattice of points combined with several symmetry elements.
 - The lattice in 2D is defined by two vectors which define the sides of the unit cells .
 - The need that the lattice span all space with translational periodicity constrains the lattice vectors to fall into one of five types known as Bravais lattices.

1.	$ \vec{a}_1 = \vec{a}_2 $ $\gamma = 90^\circ$	Square	
2.	$ \vec{a}_1 \neq \vec{a}_2 $ $\gamma = 90^\circ$	Rectangular	
3.	$ \vec{a}_1 = \vec{a}_2 $ $\gamma \neq 90^\circ, 120^\circ$	Centered Rectangular	
4.	$ \vec{a}_1 = \vec{a}_2 $ $\gamma = 120^\circ$	Hexagonal	
5.	$ \vec{a}_1 \neq \vec{a}_2 $ $\gamma \neq 90^\circ, 120^\circ$	Oblique	

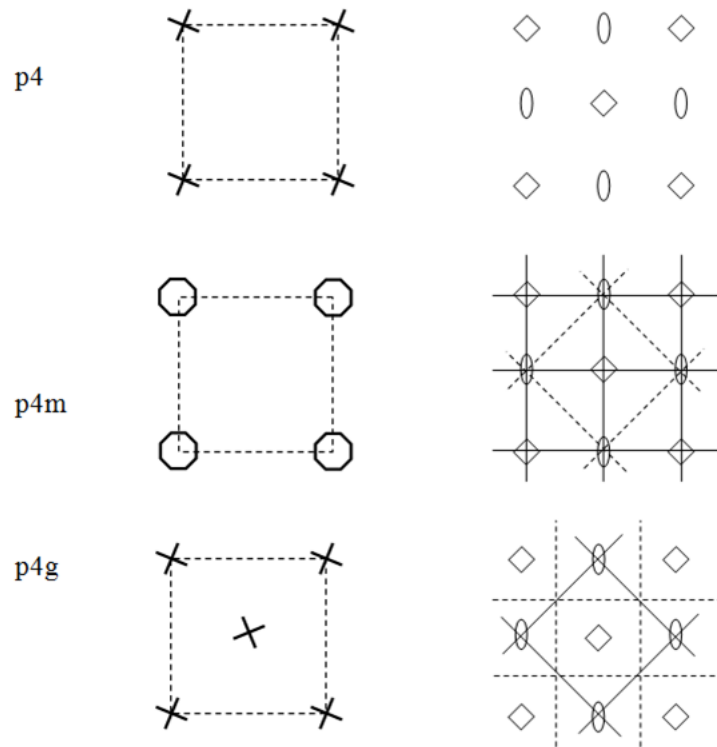
- One thing to note is that the unit cell for the lattice that is called centered rectangular doesn't look rectangular. However, the unit cell can be redrawn in such a way that it is rectangular but has a lattice point at the center and an angle of $\gamma = 90^\circ$.
- This unit cell is no longer “primitive” because it has more than one lattice point per unit cell. It is important to distinguish between primitive and non-primitive lattices. In a non-primitive lattice the points are truly equivalent and therefore there will be atoms that are equivalent and thus indistinguishable.

- Symmetry Elements

- In 2D there are three types of symmetry elements that are consistent with the constraint of translational periodicity.
 1. Rotation Axes : 2-, 3-, 4-, 6-fold
 2. Mirror planes
 3. Glide planes (screw axes according to Masel)

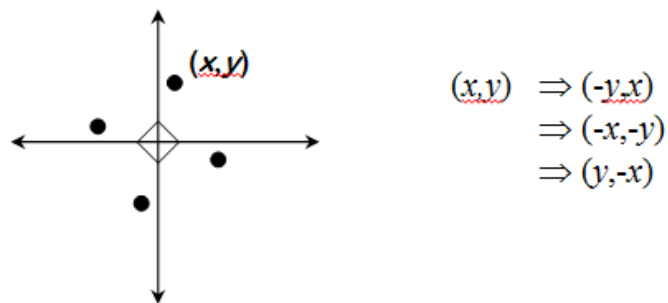


- Note that not all of these symmetry elements must be specified explicitly.
- Specifying two may imply the existence of others.
- The symmetry of the surface will be equal to or lower than that of the corresponding Bravais lattice.
- Combining the five Bravais lattices with the possible symmetry elements results in seventeen 2D plane groups. All 2D periodic structures fall into one of these groups.
- Consider just those that have the square Bravais lattice.



- Note that every one of the symmetry operations can be used to generate one point in the lattice from any other.

Example: Consider a four-fold axis at the origin (0,0) and a point at (x,y).



These transformations can be written down as matrices. The notation for the application of a rotation about fourfold axis by 90° is $\hat{C}_4^{(1)}$.

$$\hat{C}_4^{(1)} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \quad \hat{C}_4^{(1)} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -y \\ x \end{bmatrix}$$

$$\hat{C}_4^{(2)} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \quad \hat{C}_4^{(2)} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -x \\ -y \end{bmatrix}$$

$$\hat{C}_4^{(3)} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Note also that there is a rigorous mathematical meaning to these matrix operations. Two rotations by 90° about a fourfold axis would be represented by the following.

$$\hat{C}_4^{(1)} \cdot \hat{C}_4^{(1)} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} = \hat{C}_4^{(2)} = \hat{C}_2^{(1)}$$

This is the basis for a very powerful set of ideas that form the basis of “group theory”. [Carter \[1998\]](#), [Bishop \[1973\]](#)

Before continuing, we briefly show how to verify the results above. In python we have three options to describe a matrix, and it is important to know the differences.

```

1:  import numpy as np
2:
3:  C4l_list = [[0, -1],
4:             [1,  0]]   #(list)
5:
6:  C4l_array = np.array([[0, -1],
7:                       [1,  0]])   #(array)
8:
9:  C4l_matrix = np.matrix([[0, -1],
10:                         [1,  0]])   #(matrix)
11:
12:  C4l_matrix = np.matrix([[0, -1], [1,  0]])   #(matrix-alt)
```

In line 4 above, we created a list of lists; that is, there are two lists inside a list. This is not a matrix, and does not act like a matrix.

In line 7 we created a numpy.array. This is closer to a matrix, but most mathematical operations act element-wise.

In line 9 we create a numpy.matrix object. This acts like a matrix for mathematical operations.

In each of the three examples discussed above, the second row of the data was aligned with spaces so you could “see” the intended arrangement of numbers. Since the second row is “inside” a pair of brackets or parentheses, this indentation is meaningless. It is perfectly fine to define the matrix as in line 12, it is just a little harder to read.

Now we consider verifying that two four-fold rotations are equivalent to a two fold rotation.

```

1: import numpy as np
2:
3: C21 = np.matrix([[-1, 0], [0, -1]])
4:
5: C41 = np.matrix([[0, -1],      #(matrix)
6:                  [1,  0]])
7:
8: print(C41 * C41)                #(dot)
9:
10: print C41 * C41 == C21          #(equal)
11: print np.all(C41 * C41 == C21) #(all)

```

```

[[-1  0]
 [ 0 -1]]
[[ True  True]
 [ True  True]]
True

```

In line 7 we define a matrix. This is similar to a matrix in Matlab, except that you have to use a matrix function to create it. The rows are aligned for visual clarity. Because the rows are inside the function parentheses, the indentation of the second row is unimportant. In line 17 we multiply the matrix with itself. This works on a matrix, but not on a list or array.

You can visually see that the product of the two 4-fold matrices is equal to a 2-fold rotation matrix. In line 10 we compare the two quantities. Note the comparison is element-wise, and we get a new matrix of Boolean values. That is inconvenient to examine, we want a single value saying whether the matrices are equal or not. In line 11 we use the `numpy.all` command to tell us whether all of the elements in the matrix are True or not.

```

1: import numpy as np
2:
3: C21 = np.array([[-1, 0], [0, -1]])
4:
5: C41 = np.array([[0, -1],      #(matrix)
6:                  [1,  0]])
7:
8: print(np.dot(C41, C41))          #(dot)
9:
10: print np.dot(C41, C41) == C21    #(equal)
11: print np.all(np.dot(C41, C41) == C21) #(all)

```

```

[[-1  0]
 [ 0 -1]]
[[ True  True]
 [ True  True]]
True

```

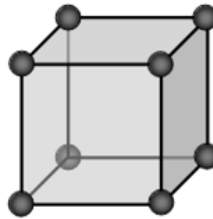
3.1.4 Symmetry in 3D

- Real surfaces are formed by slicing planes through 3D object.
- If the 3D solid has some periodicity or is crystalline then the surface that is produced by the cut will also have periodicity and symmetry.
- In 3D there are 14 possible Bravais lattices that will fill space with translational periodicity.
- The symmetry operations that can exist in 3D are :
 - mirror planes
 - rotations
 - inversion
 - rotation translation (screw axes)
- Combination of the Bravais lattices with these symmetry elements

gives 230 possible 3D space groups.

- The only metal with the simple cubic structure is Polonium.

Simple Cubic Lattice - “no-brainer”



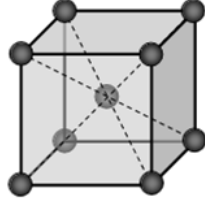
primitive lattice -
conventional lattice -

$|a_x| = |a_y| = |a_z|$, and $\alpha = \beta = \gamma = 90^\circ$
same

- Most metals have bulk structures that fall into the classes: face centered cubic, body centered cubic or hexagonal close packed.

- Both fcc and hcp are close packed structures of spheres.

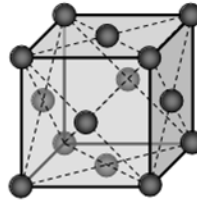
Body Centered Cubic - bcc



primitive lattice -
conventional lattice -

$|a| = |b| = |c|$, and $\alpha = 90^\circ, \beta = \gamma = 54.7^\circ$
 $|a'_x| = |a'_y| = |a'_z|$, and $\alpha' = \beta' = \gamma' = 90^\circ$
 two identical atoms - non-primitive

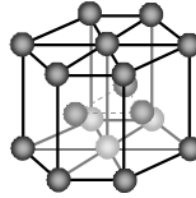
Face Centered Cubic - fcc



primitive lattice -
conventional lattice -

$|a| = |b| = |c|$, and $\alpha = 90^\circ, \beta = \gamma = 120^\circ$
 $|a'_x| = |a'_y| = |a'_z|$, and $\alpha' = \beta' = \gamma' = 90^\circ$
 four identical atoms, non-primitive

Hexagonal Close Packed - hcp



primitive lattice -
conventional lattice -

hexagonal
 $|a| = |b| \neq |c|$, and $\alpha = 120^\circ, \beta = \gamma = 90^\circ$
 same

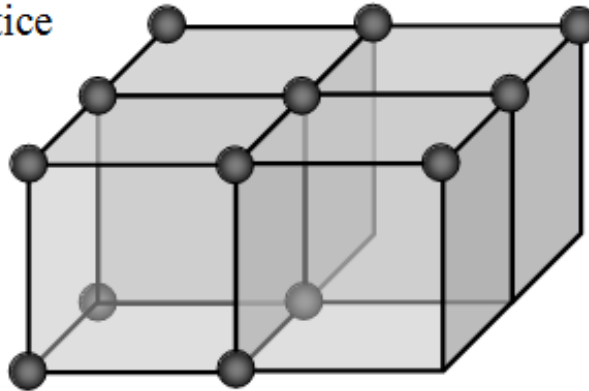
- Note that the two atoms in the hcp lattice are not identical. They are positioned above triangles of different orientation. As a result the hcp lattice has two atoms per unit cell but is still primitive.

3.2 Surfaces of the Simple Cubic Lattice

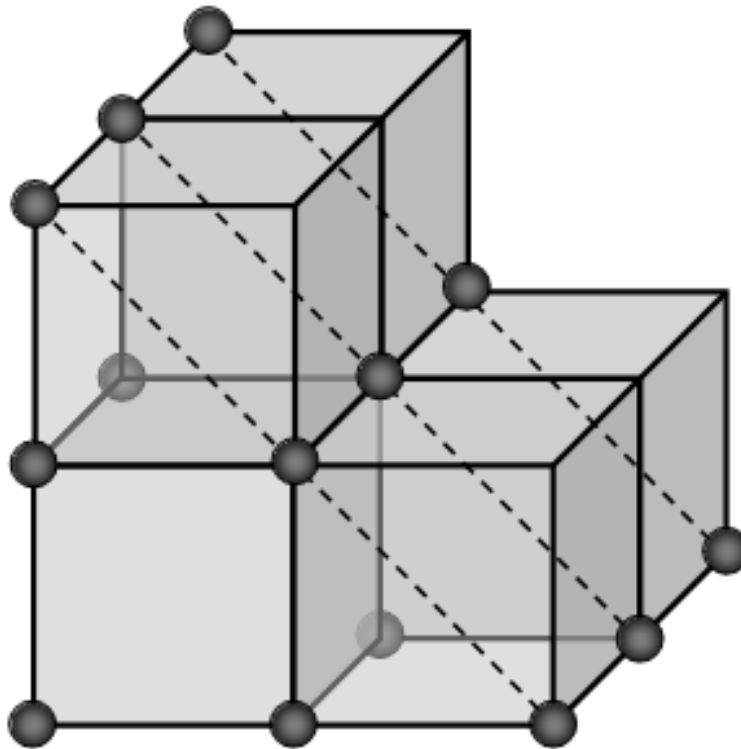
- Start by considering these just to get the idea of taking slices of three dimensional structures. Depending on how you cut the surface, you can

get square, rectangular, or even hexagonal arrangements of atoms at the surface.

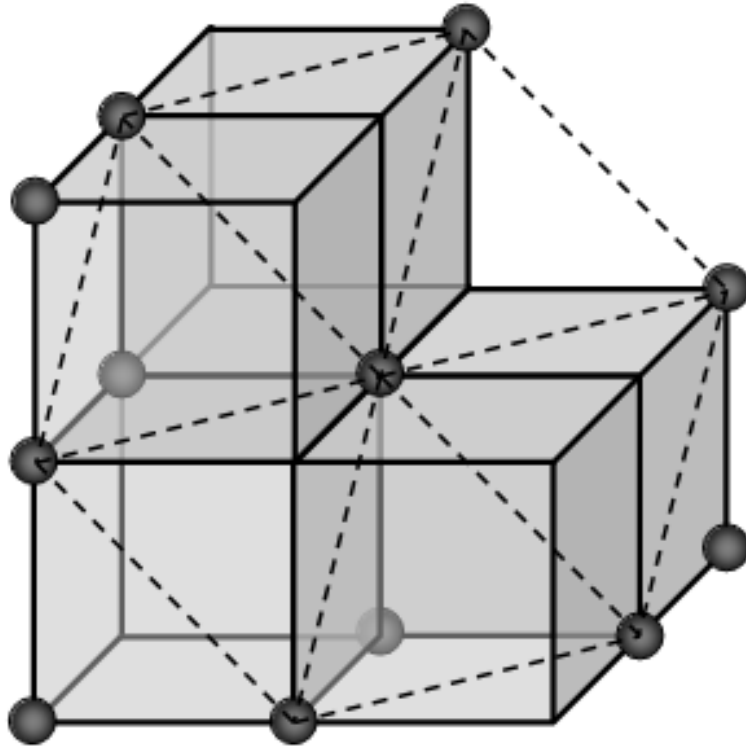
Square Lattice



Rectangular Lattice



Hexagonal Lattice



3.3 Miller Indices

- There are an infinite number of planes that can be cut through a 3D lattice. We need a way to name the planes.
- For every vector direction that one defines in the lattice there is a set of planes perpendicular to that vector.
- In a lattice that has some symmetry to it there will be equivalent directions that expose identical faces.

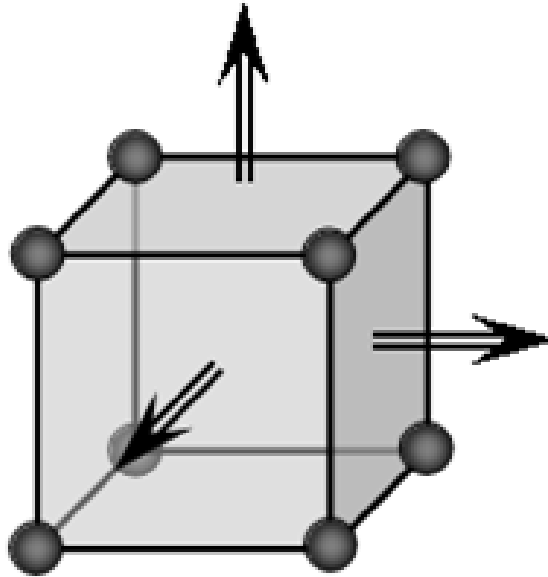
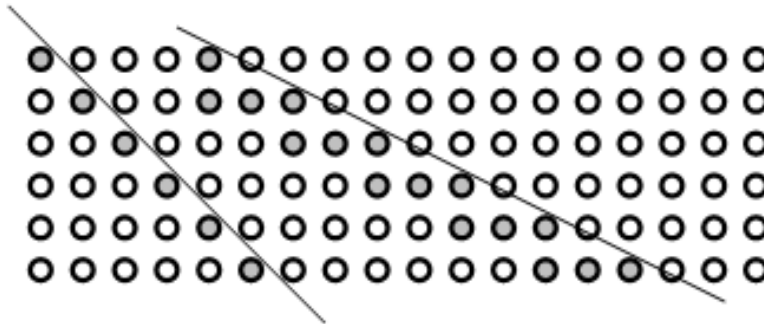
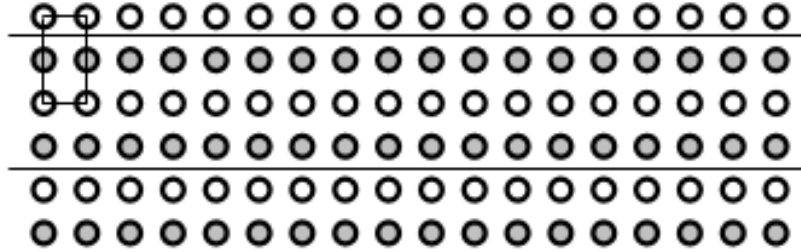


Figure 6: The arrows indicate symmetrically equivalent surfaces.

- For atomic lattices or even molecular lattices one does not usually think of cutting “through” atoms or molecules. Cuts pass between atoms and expose those whose centers lie to one side of the cutting plane.



- If one considers just the lattice then it doesn't matter at what position along a given direction that the cut is made. However, if one considered the contents of the unit cell then it does matter.



3.3.1 Miller Index Formulation

- For a plane cut through a lattice:
 1. Find the intersections (b_x , b_y , b_z) with $b_i > 1$ along each of the three lattice vectors (in units of the vector length).
 2. Find the lowest common denominator of the three lengths.

$$n = \text{lcd} \left\{ \frac{1}{b_x}, \frac{1}{b_y}, \frac{1}{b_z} \right\}$$

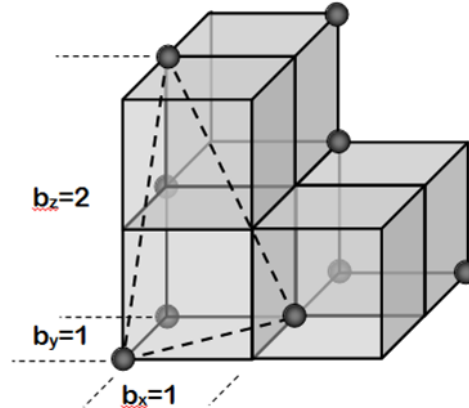
1. Define the Miller indices of the plane as:

$$(h, k, l) = \left(\frac{n}{b_x}, \frac{n}{b_y}, \frac{n}{b_z} \right).$$

1. If $b_x = \infty$ then $h = 0$

Examples :

$$\begin{aligned} (b_x, b_y, b_z) &= (1, 1, 2) \\ \text{lcd} \left(\frac{1}{1}, \frac{1}{1}, \frac{1}{2} \right) &= 2 \\ (h, k, l) &= \left(\frac{2}{1}, \frac{2}{1}, \frac{2}{2} \right) \\ &= (2, 2, 1) \end{aligned}$$



That example is pretty easy to work out in your head. Here is how to compute the lowest common denominator in Python. Note that the lowest common denominator is the same as the lowest common multiple of the numbers in the divisor, and is related to the greatest common divisor.³ Also, the lowest common denominator of three numbers is the same as the lowest common

³http://en.wikipedia.org/wiki/Least_common_multiple#Computing_the_least_common_multiple

denominator of one number with the lowest common denominator of the other two numbers.

```

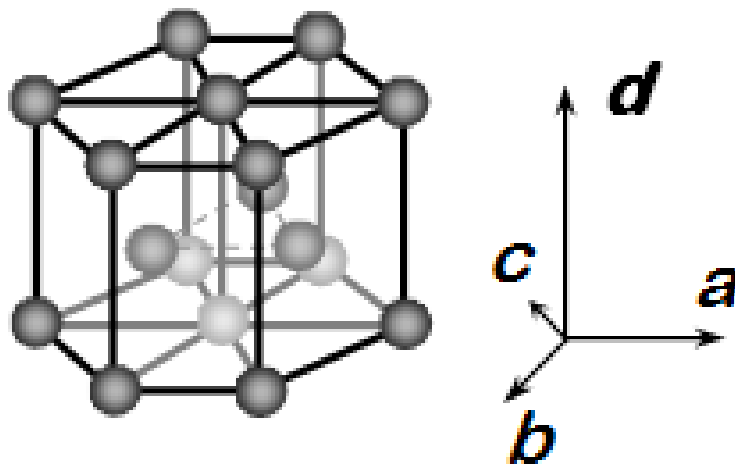
1 1: from fractions import gcd
2 2:
3 3: def LCM(a, b):
4 4:     'returns the least common multiple of a and b'
5 5:     return a * b / gcd(a, b)
6 6:
7 7: print(LCM(1, LCM(1, 2)))

```

2

In line 1 we import the `gcd` function from the `fractions` module. Line 3 illustrates a new python concept: the definition of a function. The function is called `LCM` and it takes two arguments. Line 4 is simply a documentation string that tells us what the function does. Note that the body of the function *must* be indented! This is a critical difference between python and Matlab. The standard indentation is 4 spaces. Finally, in line 7, we use the functional form of the `print` command to output the result.

- Note that for the hexagonal lattice it is quite common to use four indices (h, k, l, m) as though there were four vectors needed to define the unit cell. This is illustrated below.



- In this case the four indices are not unique and the relationship between them is that $h + k = -l$.

3.4 Common Low Miller Index Surface Structures

- The most common and important crystal structures for metals are the fcc, bcc, and hcp.
- The most common and important structures for the semiconductors are the diamond and zincblende structures.
- Of these, the most important surfaces are usually the low Miller index surfaces. These tend to be closely packed arrays that are thermodynamically stable.
- Metal crystallites in catalysts usually expose low Miller index surfaces.

3.4.1 The most common fcc surfaces

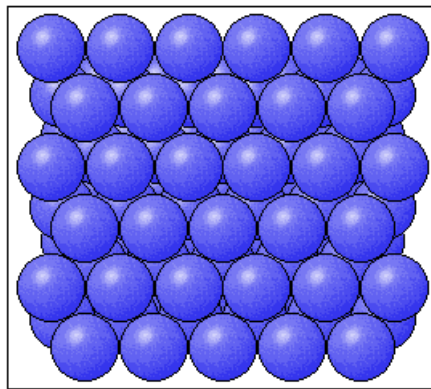


Figure 7: fcc(111)

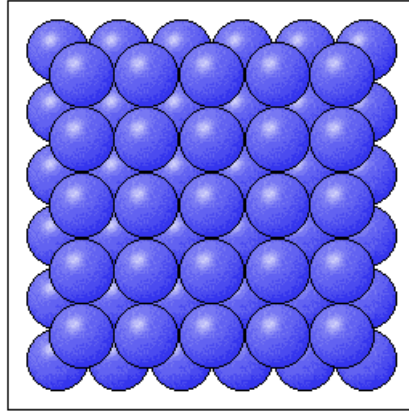


Figure 8: fcc(100)

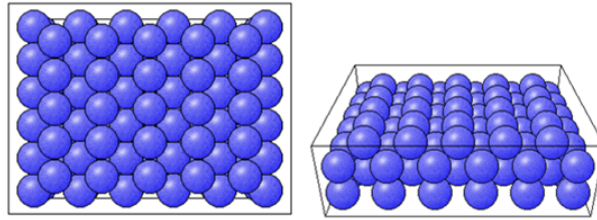


Figure 9: fcc(110)

3.4.2 The most common bcc surfaces

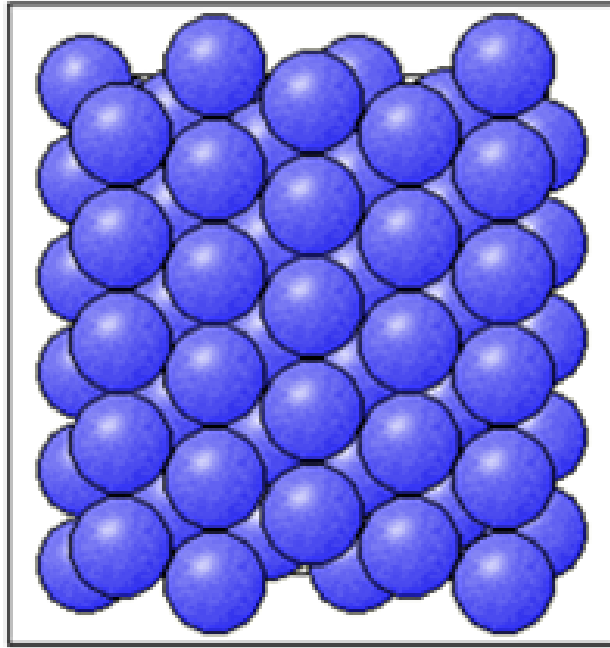


Figure 10: bcc(110)

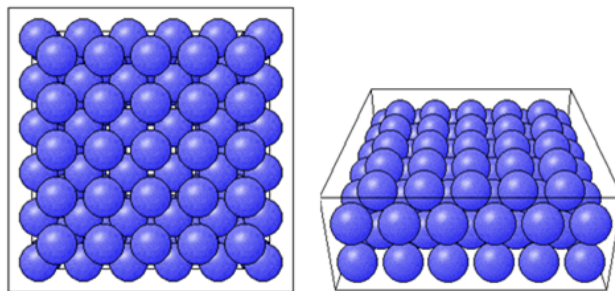


Figure 11: bcc(100)

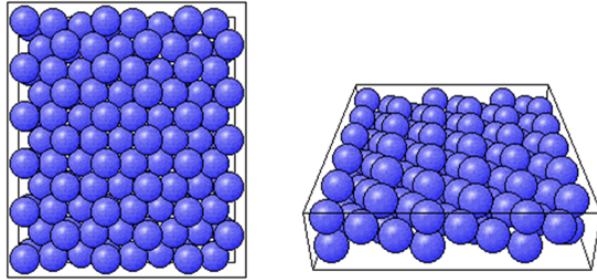


Figure 12: bcc(111)

3.4.3 hcp surfaces

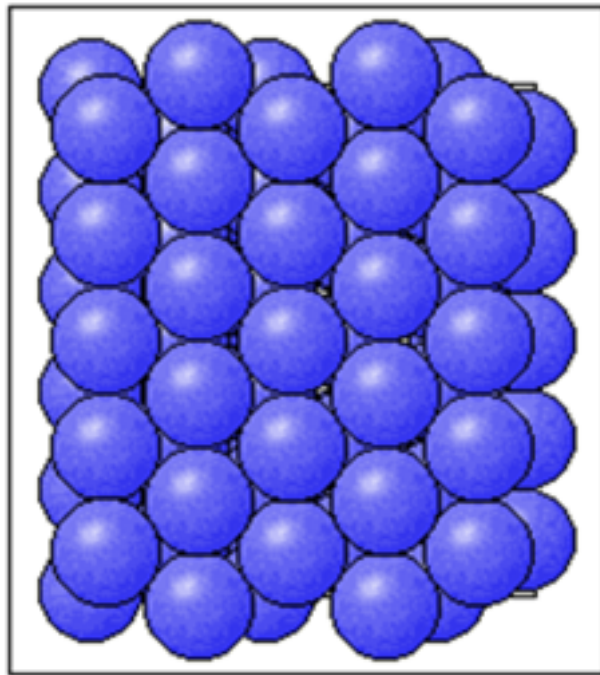


Figure 13: hcp(001)

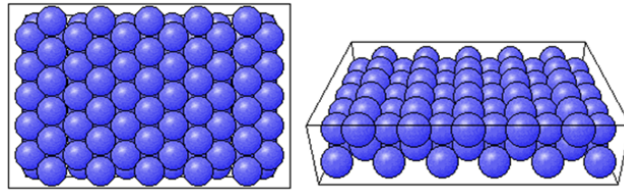
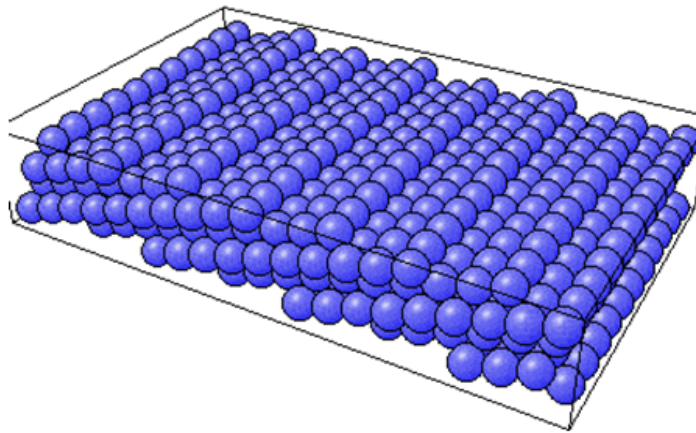


Figure 14: hcp(100)

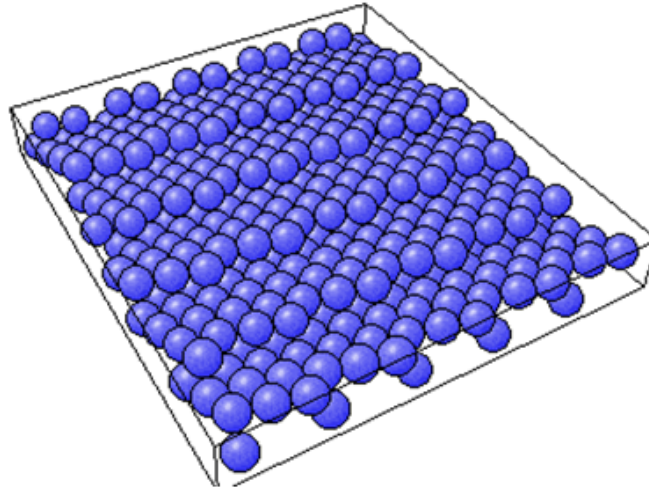
3.5 High Miller Index Surfaces

- Surfaces cut along planes that are not low Miller index planes can have very complex structures and are certainly not close packed.
- The structures that are exposed by high Miller index cuts are :
 - monoatomic steps, and
 - kinked steps.

fcc(511)



fcc(643)



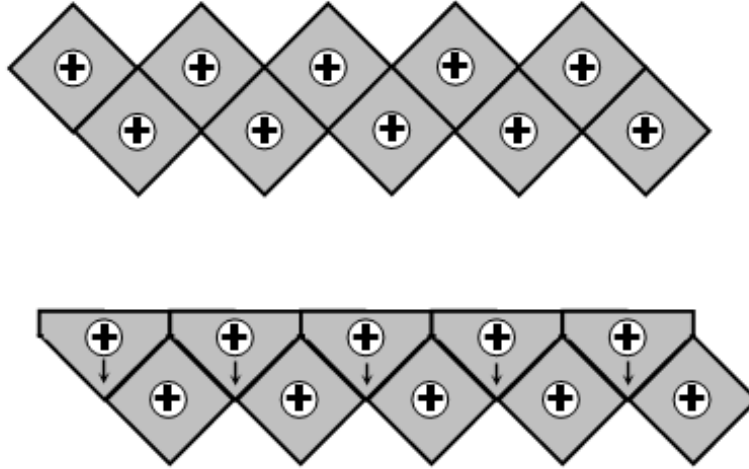
- All possible structure can be viewed on the web at the site:

<http://surfexp.fhi-berlin.mpg.de/>

- It is important to realize that the steps have certain structure to them also and that they can be thought of as planes that project up through the surface.
- The fcc(511) surface can be thought of as (111) steps combined with the (100) terraces.
- When thinking about the step structure it is important to remember the atoms within the original unit cell.

3.6 Surface Relaxation

- Cleavage of a 3D solid removes one degree of translational periodicity. The semi-infinite solid is now only periodic in the plane parallel to the cut.
- Clearly this must have some effect on the positions of atoms at the surface. They are now only bonded to atoms on one side and must react to minimize total energy.
- The universally observed effect is that there is a contraction of the layers at the surface.
- The contraction is due to the contraction of electron density into the bulk of the solid.

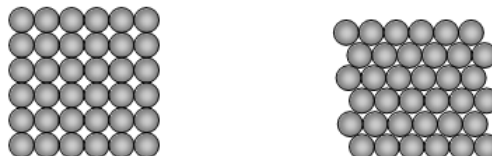


- For the close packed surfaces the contraction is minimal; 1-3% of the plane spacing.
- For non-close packed low Miller index surfaces the contraction can be 5-10% of the plane spacing.
- For rough surfaces the contraction is greater as a percentage of the surface plane spacing because the surface plane spacing is lower. The absolute contraction is not necessarily much different from that of a low Miller index surface.
- Relaxation preserves the 2D translational periodicity of the bulk.

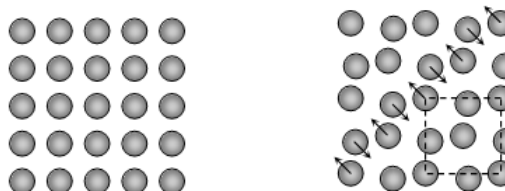
3.7 Reconstruction

- The need to minimize the total energy of the surface can drive the surface to much more drastic rearrangements than relaxation.
- Reconstruction involves an atomic rearrangement that changes the 2D translational periodicity of the surface.

Pt(100) - hex



W(100) - c(2x2)



Au(110) - (2x1)



- The tendency is that the close packed or low Miller index surfaces of metals are thermodynamically stable.
- A number of the reconstructions that have been observed can be rationalized in terms of a surface taking on close packed structures. The Au(110)-(2 × 1) forms facets of (111) surfaces.
- One of the types of reconstruction that should be obvious is the coalescence of steps. Steps are high energy features but are prevented from coalescence by dipole repulsion.
- At high enough step density, however, these become unstable and do coalesce into polyatomic steps.

4 Computational crystallography

In this section we show some general tools for computing crystallographic properties, such as unit cell volume, distances, distance between planes, etc. . . We will only consider the linear algebra approach to these problems.

First, we consider that every vector has a set of basis vectors it is associated with. For example, the vector $2\hat{e}_1 + 3\hat{e}_2 + 4\hat{e}_3$ can be compactly represented by $[2, 3, 4]$ with the implied cartesian vector basis set:

$$\hat{e}_1 = 1\vec{x} + 0\vec{y} + 0\vec{z} \quad (4)$$

$$\hat{e}_2 = 0\vec{x} + 1\vec{y} + 0\vec{z} \quad (5)$$

$$\hat{e}_3 = 0\vec{x} + 0\vec{y} + 1\vec{z} \quad (6)$$

This special set of vectors is orthonormal (each vector has unit length and is orthogonal to the other vectors).

4.1 Computing distances between two points

We know that the length of a vector is simply the square root of the dot-product of the vector with itself, so we can express the distance between two points as the length of the vector connecting them. Let point 1 be defined by a position vector \vec{p} and point 2 by a position vector \vec{q} . We know the distance between point 1 and point 2 is:

$$D = \sqrt{(\vec{q} - \vec{p}) \cdot (\vec{q} - \vec{p})}$$

```

1 import numpy as np
2 p = np.array([0,0,0])
3 q = np.array([1,1,0])
4
5 D = np.sqrt(np.dot((q-p),(q-p)))
6 print D

```

1.41421356237

This only works, however, when the basis vectors are orthonormal! In many unit cells, the lattice vectors are not orthonormal, and it is more difficult to compute distances. To generalize the formula, we need to introduce the metric tensor. [Graef and McHenry \[2007\]](#)

$$g = \begin{bmatrix} \vec{a} \cdot \vec{a} & \vec{a} \cdot \vec{b} & \vec{a} \cdot \vec{c} \\ \vec{b} \cdot \vec{a} & \vec{b} \cdot \vec{b} & \vec{b} \cdot \vec{c} \\ \vec{c} \cdot \vec{a} & \vec{c} \cdot \vec{b} & \vec{c} \cdot \vec{c} \end{bmatrix}$$

Note that for an orthonormal basis, this matrix is simply the identity matrix. With this definition, we define the distance simply as:

$$D = \sqrt{(q - p) \cdot g \cdot (q - p)}$$

Here is an example of computing the distance between two points defined in a unit cell basis, i.e. $[1/2, 1/3, 1/4]$ means $1/2\vec{a} + 1/3\vec{b} + 1/4\vec{c}$ where each lattice vector is in turn defined, e.g. $\vec{a} = 2\vec{x}$, $\vec{b} = 2\vec{y}$ and $\vec{c} = 3\vec{z}$.

```

1 import numpy as np
2
3 # define lattice vectors
4 a = np.array([2, 0, 0])
5 b = np.array([0, 2, 0])

```

```

6  c = np.array([0, 0, 3])
7
8  # define metric tensor
9  g = np.array([[np.dot(a,a), np.dot(a,b), np.dot(a,c)],
10                [np.dot(b,a), np.dot(b,b), np.dot(b,c)],
11                [np.dot(c,a), np.dot(c,b), np.dot(c,c)]])
12
13 # define points
14 p = np.array([0.5, 1./3., 0.25])
15 q = np.array([1./3., 0.5, 0.75])
16
17 # compute distance
18 D = np.sqrt(np.dot((p-q), np.dot(g, (p-q))))
19
20 print('The distance between the points is {0:1.2f}'.format(D))

```

The distance between the points is 1.57.

4.2 Computing the angle between two vectors

When vectors are given in cartesian coordinates, we can easily calculate the length and angles between vectors using the `Scientific.Geometry.Vector` class. Here is an example:

```

1  import numpy as np
2  from Scientific.Geometry import Vector
3
4  A = Vector([ 4, 0, 0])
5  B = Vector([ 0, 6, 0])
6  C = Vector([-2.5, 0, 5*np.sqrt(3)/2.])
7
8  print('a = {0} angstroms'.format(A.length()))
9  print('b = {0} angstroms'.format(B.length()))
10 print('c = {0} angstroms'.format(C.length()))
11 print('alpha = {0:1.1f} degrees'.format(B.angle(C)*180./np.pi))
12 print('beta = {0:1.1f} degrees'.format(A.angle(C)*180./np.pi))
13 print('gamma = {0:1.1f} degrees'.format(A.angle(B)*180./np.pi))

```

```

a = 4.0 angstroms
b = 6.0 angstroms
c = 5.0 angstroms
alpha = 90.0 degrees
beta = 120.0 degrees
gamma = 90.0 degrees

```

In the example above, we show how to construct the $a, b, c, \alpha, \beta, \gamma$ representation of a 3D unit cell.

When the vectors are not given in cartesian coordinates, but rather in some basis set of vectors that is not orthonormal, the example above does not work. We need a different approach to calculate angles between vectors. We define [Graef and McHenry \[2007\]](#) without derivation, that

$$\theta = \cos^{-1} \frac{\vec{a} \cdot \vec{g} \cdot \vec{b}}{|\vec{a}| |\vec{b}|}$$

Let us recalculate the angles between the vectors in the previous example with this equation.

```

1 import numpy as np
2 A = [1, 0, 0]
3 B = [0, 1, 0]
4 C = [0, 0, 1]
5
6 # these are the basis vectors
7 a = np.array([ 4, 0, 0])
8 b = np.array([ 0, 6, 0])
9 c = np.array([-2.5, 0, 5*np.sqrt(3)/2.])
10
11 # define metric tensor
12 g = np.array([[np.dot(a,a), np.dot(a,b), np.dot(a,c)],
13               [np.dot(b,a), np.dot(b,b), np.dot(b,c)],
14               [np.dot(c,a), np.dot(c,b), np.dot(c,c)]])
15
16 alpha = np.arccos(np.dot(B, np.dot(g, C))
17                  /(np.sqrt(np.dot(B, np.dot(g, B)))
18                  * np.sqrt(np.dot(C, np.dot(g, C))))) * 180./np.pi
19
20 beta = np.arccos(np.dot(A, np.dot(g, C))
21                 /(np.sqrt(np.dot(A, np.dot(g, A)))
22                 * np.sqrt(np.dot(C, np.dot(g, C))))) * 180./np.pi
23
24 gamma = np.arccos(np.dot(A, np.dot(g, B))
25                  /(np.sqrt(np.dot(A, np.dot(g, A)))
26                  * np.sqrt(np.dot(B, np.dot(g, B))))) * 180./np.pi
27
28 print alpha
29 print beta
30 print gamma

```

90.0

120.0

90.0

As an example, we consider the vectors $[1, 0, 1]$ and $[-2, 0, 1]$ in the basis vectors $[4, 0, 0]$, $[0, 6, 0]$, and $[-2.5, 0, 5\sqrt{3}/2]$. What is the angle between these vectors?

```

1 import numpy as np
2
3 a = np.array([ 4, 0, 0])
4 b = np.array([ 0, 6, 0])
5 c = np.array([-2.5, 0, 5*np.sqrt(3)/2.])
6
7 # define metric tensor
8 g = np.array([[np.dot(a,a), np.dot(a,b), np.dot(a,c)],
9               [np.dot(b,a), np.dot(b,b), np.dot(b,c)],
10              [np.dot(c,a), np.dot(c,b), np.dot(c,c)]])
11
12 p = np.array([1, 0, 1])
13 q = np.array([-2, 0, 1])
14
15 lp = np.sqrt(np.dot(p, np.dot(g, p)))
16 lq = np.sqrt(np.dot(q, np.dot(g, q)))
17
18 theta = np.arccos(np.dot(p, np.dot(g, q))/(lp * lq))
19 print 'The angle between the vectors is {0:1.2f} degrees.'.format(theta * 180. / np.pi)

```

The angle between the vectors is 86.70 degrees.

4.3 Computing unit cell volumes

The volume of a unit cell is particularly simple to compute from the unit cell vectors: $V = |\det U|$ where U is the unit cell vector matrix in cartesian coordinates. We must take the absolute value because the order of vectors is arbitrary, and some orders will result in negative determinants.

Let us consider a trivial example of the volume of a cube first.

```
1 import numpy as np
2
3 a = np.array([5, 0, 0])
4 b = np.array([0, 5, 0])
5 c = np.array([0, 0, 5])
6
7 U = np.vstack([a, b, c])
8 print U
9 print('V = {0:1.3f} ang^3'.format(np.abs(np.linalg.det(U))))
```

```
[[5 0 0]
 [0 5 0]
 [0 0 5]]
V = 125.000 ang^3
```

Now, consider the primitive lattice vectors of an fcc lattice with a lattice constant of 5 and ask what is the volume?

```
1 import numpy as np
2
3 U = 5.0 * np.array([[0.5, 0.5, 0.0],
4                    [0.5, 0.0, 0.5],
5                    [0.0, 0.5, 0.5]])
6
7 print('V = {0:1.3f} ang^3'.format(np.abs(np.linalg.det(U))))
```

```
V = 31.250 ang^3
```

No geometry required! Note that the volume of the primitive cell is 1/4 of the conventional cubic cell. Naturally, the primitive cell has one atom in it, whereas the conventional cell has four atoms in it.

```
1 import numpy as np
2
3 U = 5.0 * np.array([[0.5, 0.5, 0.0],
4                    [0.5, 0.0, 0.5],
5                    [0.0, 0.5, 0.5]])
6
7 print('V = {0:1.3f} ang^3'.format(np.abs(np.linalg.det(U))))
8
9 a = U[0]
10 b = U[1]
11 c = U[2]
12
13
14 print np.abs(np.dot(a, np.cross(b, c)))
15 print np.abs(np.dot(b, np.cross(a, c)))
16 print np.abs(np.dot(c, np.cross(b, a)))
```

```
V = 31.250 ang^3
31.25
31.25
31.25
```

4.4 The reciprocal lattice in crystallography

The reciprocal lattice is a convenient, alternative set of basis vectors that are defined by:

$$A \cdot A^{*T} = I$$

Or,

$$A^* = (A^{-1})^T$$

Let us consider a brief example:

```
1 import numpy as np
2
3 # primitive fcc lattice
4 A = 3.61*np.array([[0.0, 0.5, 0.5],
5                   [0.5, 0.0, 0.5],
6                   [0.5, 0.5, 0.0]])
7
8 Astar = np.linalg.inv(A).T
9
10 print Astar
```

```
[[ -0.27700831  0.27700831  0.27700831]
 [ 0.27700831 -0.27700831  0.27700831]
 [ 0.27700831  0.27700831 -0.27700831]]
```

It is of some interest that the reciprocal lattice of the fcc lattice has the same symmetry as the bcc lattice!

The reciprocal lattice is of particular significance because a vector in the reciprocal lattice basis set with components (h, k, l) is normal to the surface plane with Miller indices (h, k, l). Furthermore, the distance between the planes with Miller indices is related inversely to the length of the reciprocal lattice vector. Thus, we find ourselves needing to conveniently calculate the length of a reciprocal lattice vector, which may be defined in a non-orthonormal set of basis vectors. As we did in real space, we define the reciprocal metric tensor:

$$g^* = \begin{bmatrix} a^* \cdot a^* & a^* \cdot b^* & a^* \cdot c^* \\ b^* \cdot a^* & b^* \cdot b^* & b^* \cdot c^* \\ c^* \cdot a^* & c^* \cdot b^* & c^* \cdot c^* \end{bmatrix}$$

As we saw in the real space computational crystallography section, the reciprocal metric tensor is useful in computing quantities in the reciprocal basis vector space.

For example, given a cubic crystal with a lattice constant of 2, compute the distance between the (110) planes. It was stated that the distance between two planes is inversely related to the length of the reciprocal lattice vector with components of the Miller indices that defines the plane of interest.

```

1  import numpy as np
2
3  A = np.array([[2, 0, 0],
4                [0, 2, 0],
5                [0, 0, 2]])
6
7  Astar = np.linalg.inv(A).T
8
9  astar = Astar[0, :]
10 bstar = Astar[1, :]
11 cstar = Astar[2, :]
12
13 gstar = np.array([[np.dot(astar, astar), np.dot(astar, bstar), np.dot(astar, cstar)],
14                   [np.dot(bstar, astar), np.dot(bstar, bstar), np.dot(bstar, cstar)],
15                   [np.dot(cstar, astar), np.dot(cstar, bstar), np.dot(cstar, cstar)]])
16
17 # MUCH more compact definition of g*
18 gstar = np.dot(Astar, Astar.T)
19
20 v = [1, 1, 0]
21
22 d_110 = np.sqrt(np.dot(v, np.dot(gstar, v)))
23
24 spacing = 1./d_110
25 print 'The spacing between the planes is {0}.'.format(spacing)

```

```

[[ 0.25  0.    0. ]
 [ 0.    0.25  0. ]
 [ 0.    0.    0.25]]

```

The spacing between the planes is 1.41421356237.

Let us consider another example, with a more complex unit cell. Find the spacing between (111) planes for the following unit cell.

```

1  import numpy as np
2
3  # these are the basis vectors
4  a = np.array([ 3, 0, 0])
5  b = np.array([-2, 0, 4*np.sqrt(3)/2.])
6  c = np.array([ 0, 6, 0])
7
8
9  A = np.vstack([a, b, c])
10
11 Astar = np.linalg.inv(A).T
12
13 # define reciprocal metric tensor
14 gstar = np.dot(Astar, Astar.T)
15
16 v = [1, 1, 1]
17 dstar_111 = np.sqrt(np.dot(v, np.dot(gstar, v)))
18 print 'd = ', 1./dstar_111

```

d = 1.64316767252

The linear algebra notation we use here is very compact, and avoids the need to memorize the special equations that are valid for specific crystal systems.

4.5 Coordinate system transformations

It is frequently convenient to work in different coordinate systems. For example, the cartesian coordinate system is intuitive, but it can be difficult to tell if a position is inside a unit cell that if the unit cell vectors are not orthogonal. In this case it can be convenient to work in fractional coordinates, where the point uses the unit cell vectors as a basis. Alternatively, if there is a lattice, it may be convenient to use the primitive lattice vectors as a basis, because then all points will have vectors with integer components. It is not difficult to interchange between these different basis sets if we understand some linear algebra.

Let us consider computing the fractional coordinates of a position in the unit cell basis set. Given a point at $\vec{p} = p_1\hat{x} + p_2\hat{y} + p_3\hat{z}$, and the unit cell vectors

$$\underline{\underline{A}} = \begin{bmatrix} \vec{a} \\ \vec{b} \\ \vec{c} \end{bmatrix}, \text{ we can express the fractional coordinates as}$$

$$\vec{s} = (\underline{\underline{A}}^T)^{-1} \vec{p}^T$$

\vec{s} conventionally has components between 0 and 1, and anything less than zero or greater than one is outside the unit cell.

Let us consider another example. Let us compute the unit cell vectors of the conventional fcc unit cell in the primitive lattice vector basis set.

```

1: import numpy as np
2:
3: # conventional vectors
4: A = np.array([[1, 0, 0],
5:               [0, 1, 0],
6:               [0, 0, 1]])
7:
8: P = np.array([[0.0, 0.5, 0.5],
9:               [0.5, 0.0, 0.5],
10:              [0.5, 0.5, 0.0]])
11:
12: a = np.dot(np.linalg.inv(P.T), A.T).T
13:
14: print a
```

```

[[-1.  1.  1.]
 [ 1. -1.  1.]
 [ 1.  1. -1.]]
```

The vectors should have components that are all integers (every lattice point must be an integer linear combination of the primitive lattice vectors). There is an important note to consider about line 12 and that is the final transpose of the dot product. Without this, the components of the transformed vectors are in *columns*, and we have constructed everything in *rows*. In this specific case, it does not matter because the matrix is highly symmetric, and the transpose does

not change it. In less symmetric cases, it can matter a lot, and it is important to keep the linear algebra in mind. The vector of the point being transformed must be a column vector, or a matrix with the components going column-wise.

```
1 import numpy as np
2
3 A = np.array([[2, 0, 0],
4               [0, 3, 0],
5               [0, 0, 4]])
6
7 p = np.array([[1, 1.5, 2],
8               [2./4, 3./4, 4./4]])
9
10 print np.dot(np.linalg.inv(A.T), p.T).T
```

```
[[ 0.5  0.5  0.5 ]
 [ 0.25 0.25 0.25]]
```

5 Low Energy Electron Diffraction (LEED)

- As mentioned the structures of surface are not just what they would appear to be from cleaving of the bulk crystal. Reconstructions can occur that change the periodicity of the surface lattice. Relaxations cause changes in the layer spacings at solid surfaces. Furthermore, adsorbed atoms and molecules can create new surface lattices, induce reconstructions or even remove the reconstructions of the clean surfaces.
- There are several types of structural experiments that have been developed. The most common are Low Energy Electron Diffraction (LEED) and Scanning Tunneling Microscopy (STM).
- LEED is the primary way in which the structures of surfaces and the adsorbed layers have been determined quantitatively and is quite commonly done by most research groups in the field of surface science.
- LEED can be used in one of two modes :
 - determination of unit cells of surfaces and adsorbed overlayers, and
 - true quantitative structure determination.
- The quantitative determination of structure is very difficult and highly computationally intensive.

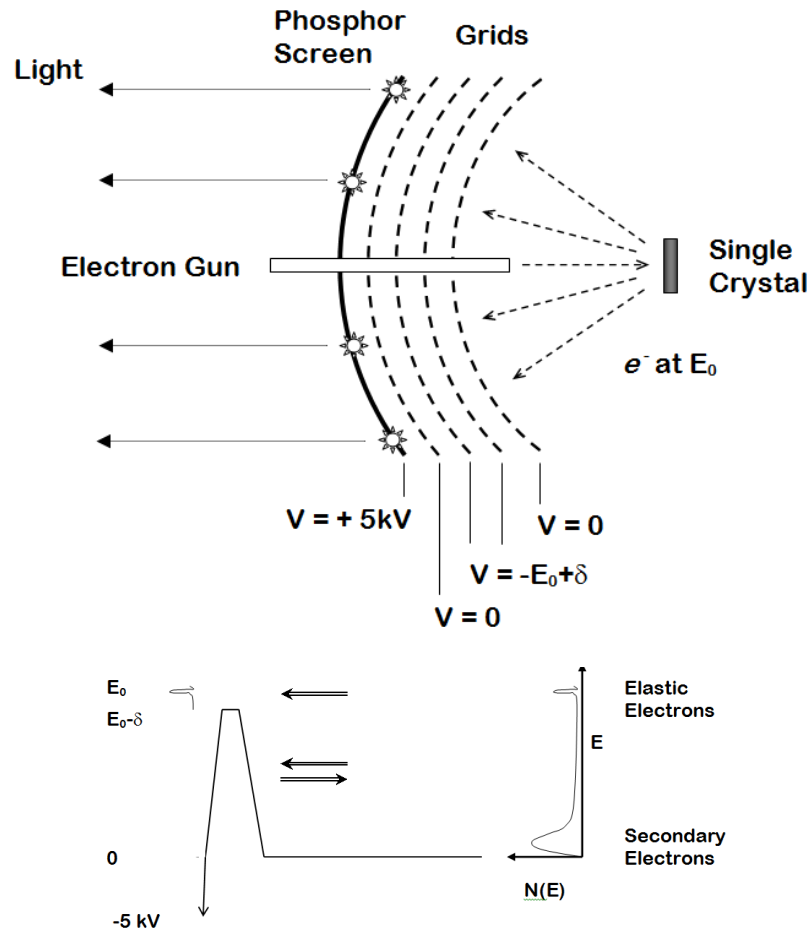


Figure 15: Potential energy profile seen by the electron beam.

- The phosphorus screen allows easy and cheap visualization of the LEED pattern. Usually a photograph is taken of the back side of the screen. This projects the diffraction pattern onto a plane.
- More complex methods of detection allow measurement of the diffraction spot profiles and the intensities.

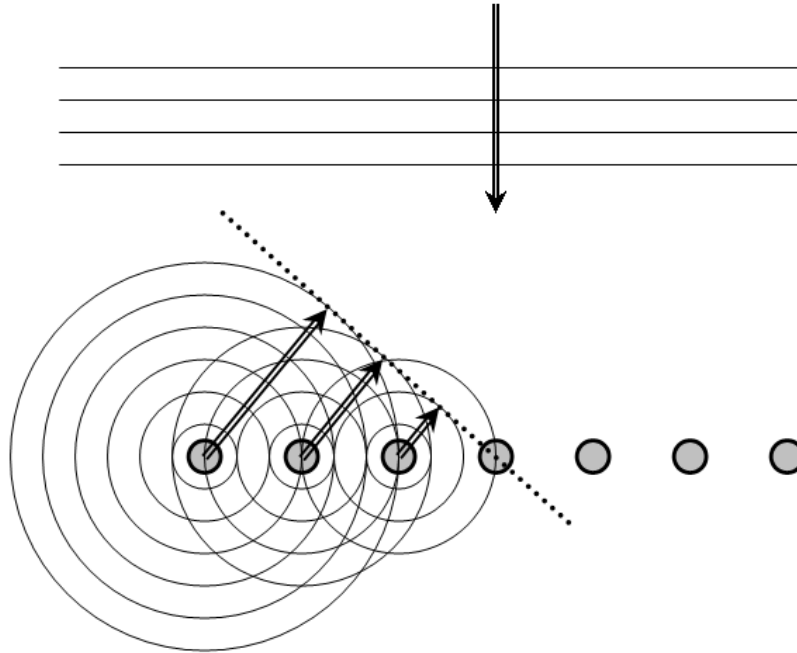
5.1 Diffraction

- Diffraction requires wavelike behavior of the electron. This was postulated by de Broglie.

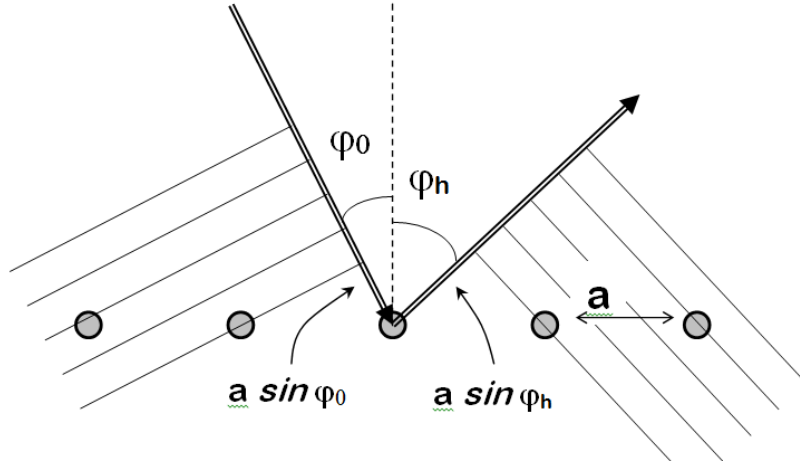
$$\lambda = \frac{h}{p} = \sqrt{\frac{150}{E_{kin}}}$$

This gives the wavelength in Å for kinetic energy in eV.

- Let us review the rules of diffraction from a lattice in 1-D. Basically diffraction is a process in which a wave is incident on an array of atoms. To first order the waves scattered by each atom are spherical waves propagating with the same wavelength as the incident wave. At long distances from the array the scattered waves can be considered to be plane waves from each source atom.



- Diffraction occurs when the distances traveled by the scattered plane waves all differ by some integral multiple of the wavelength. A wave with wavelength λ is incident on a 1D lattice with spacing a at an angle ϕ_0 .

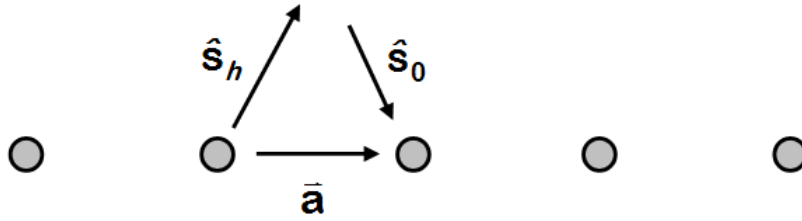


- In order to have constructive interference and obtain a diffracted beam the waves scattering off adjacent lattice points must travel distances which differ by integral multiples of the wavelength.
- The diffraction condition must be that

$$a(\sin \phi_h - \sin \phi_0) = h\lambda$$

for $h \in I$. For a given incident angle ϕ_0 this determines the angles ϕ_h at which diffracted beams appear.

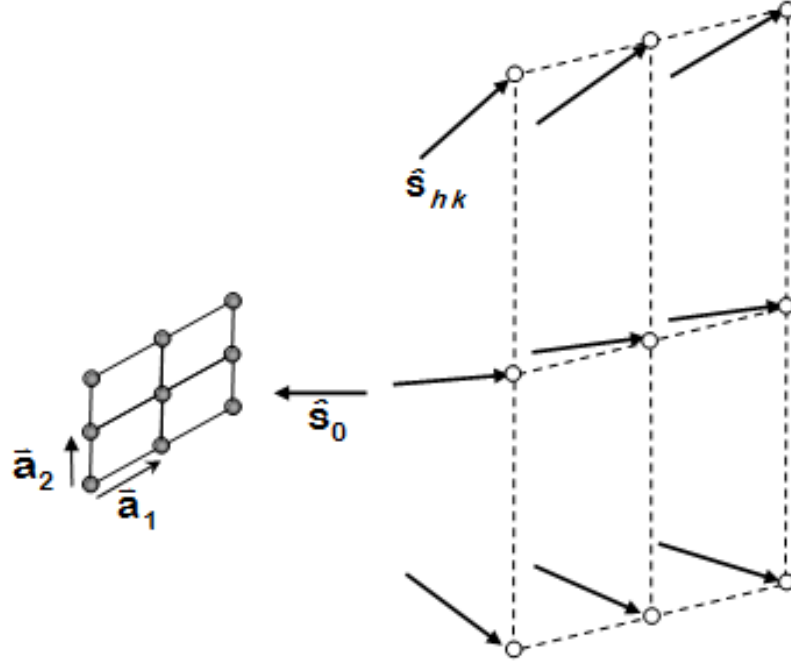
- Another way of writing this equation is to think of the lattice spacing as a vector \vec{a} and the incident and scattered waves as traveling along directions given by the unit vectors \hat{s}_0 and \hat{s}_h .



- Then the diffraction condition can be written as

$$\vec{a} \cdot (\hat{s}_h - \hat{s}_0) = h\lambda$$

Consider the 2D diffraction problem from a lattice with vectors \vec{a}_1 and \vec{a}_2 . The beam is incident along a vector \hat{s}_0 and diffracting in a direction \hat{s}_{hk} .



- Now the diffraction conditions are given by

$$\vec{a}_1 \cdot (\hat{s}_{hk} - \hat{s}_0) = h\lambda$$

and

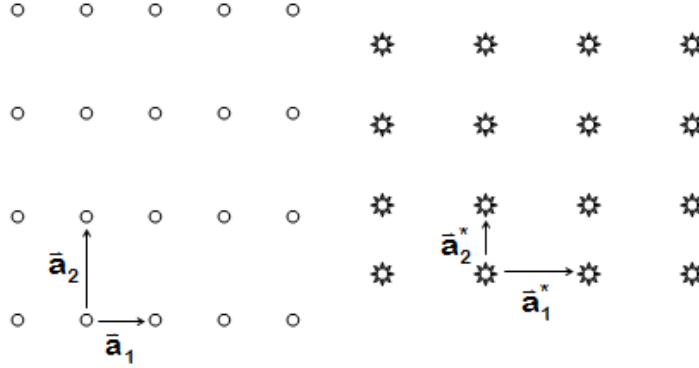
$$\vec{a}_2 \cdot (\hat{s}_{hk} - \hat{s}_0) = k\lambda.$$

Both must be satisfied in order to get diffraction

5.2 The reciprocal lattice

For any lattice vectors, (\vec{a}_1, \vec{a}_2) , we can define a new set of vectors, which we call the reciprocal lattice $(\vec{a}_1^*, \vec{a}_2^*)$ based on these definitions:

$$\begin{aligned}\bar{\mathbf{a}}_1 \cdot \bar{\mathbf{a}}_1^* &= 1 & \bar{\mathbf{a}}_1 \cdot \bar{\mathbf{a}}_2^* &= 0 \\ \bar{\mathbf{a}}_2 \cdot \bar{\mathbf{a}}_1^* &= 0 & \bar{\mathbf{a}}_2 \cdot \bar{\mathbf{a}}_2^* &= 1\end{aligned}$$



- Note that the reciprocal lattice is periodic and has symmetry just like the real space lattice.
- The power of this construction is that it provides us with the solutions to the diffraction problem. The directions in which diffraction occurs are given by

$$\hat{s} - \hat{s}_0 = h\lambda\bar{\mathbf{a}}_1^* - k\lambda\bar{\mathbf{a}}_2^*$$

Let us consider a general approach to computing the reciprocal lattice. First, we consider how to represent the vectors in the form of a matrix. For the two lattice vectors \vec{a}_1 and \vec{a}_2 , we express them in a matrix as: $A = \begin{bmatrix} \vec{a}_1 \\ \vec{a}_2 \end{bmatrix}$. We next consider the reciprocal lattice vectors in the columnwise matrix, $A^* = \begin{bmatrix} \vec{a}_1^* \\ \vec{a}_2^* \end{bmatrix}$. Now, we have the following equation: $A \cdot A^{*T} = I$, where I is the identity matrix. We can readily solve for the reciprocal lattice vectors as $A^* = (I \cdot A^{-1})^T = (A^{-1})^T$.

Let us consider a specific example where $a_1 = 1x + 0y$ and $a_2 = 0x + 2y$.

```

1: import numpy as np
2:
3: a1 = [1, 0]           #(a1)
4: a2 = [0, 2]
5:
6: A = np.vstack([a1, a2]) #(vstack)
7: print('A = \n{0}'.format(A))
8:
9: Astar = np.linalg.inv(A).T #(inv)
10:
11: a1star= Astar[0]        #(a1star)
12: a2star = Astar[1]
13: print 'a1* = {0}'.format(a1star)
```

```

14 14: print 'a2* = {}'.format(a2star)
15 15:
16 16: # Verify the requirements.
17 17: print np.dot(a1, a1star)    #(dot)
18 18: print np.dot(a1, a2star)
19 19: print np.dot(a2, a1star)
20 20: print np.dot(a2, a2star)

```

```

A =
[[1 0]
 [0 2]]
a1* = [ 1.  0.]
a2* = [ 0.  0.5]
1.0
0.0
0.0
1.0

```

This method works generally for 2 and 3 dimensional lattices. Some notes about the code above:

1. In line 3 we represent the components of the vector as a *list*.
2. In line 6 we create an array by using `numpy.vstack`. This makes the columns of the array the components of the vector. An array is not the same as a matrix!
3. In line 9 we calculate the inverse of the array. Note that now the components of the reciprocal lattice vectors are in the *rows* of the resulting array.
4. In line 11 we use indexing to extract the first row. In Python, indexing starts at 0.
5. In line 17 we must use the `numpy.dot` function to get the matrix multiplication, or dot product of the two vectors. The `*` operator will perform element-wise multiplication.

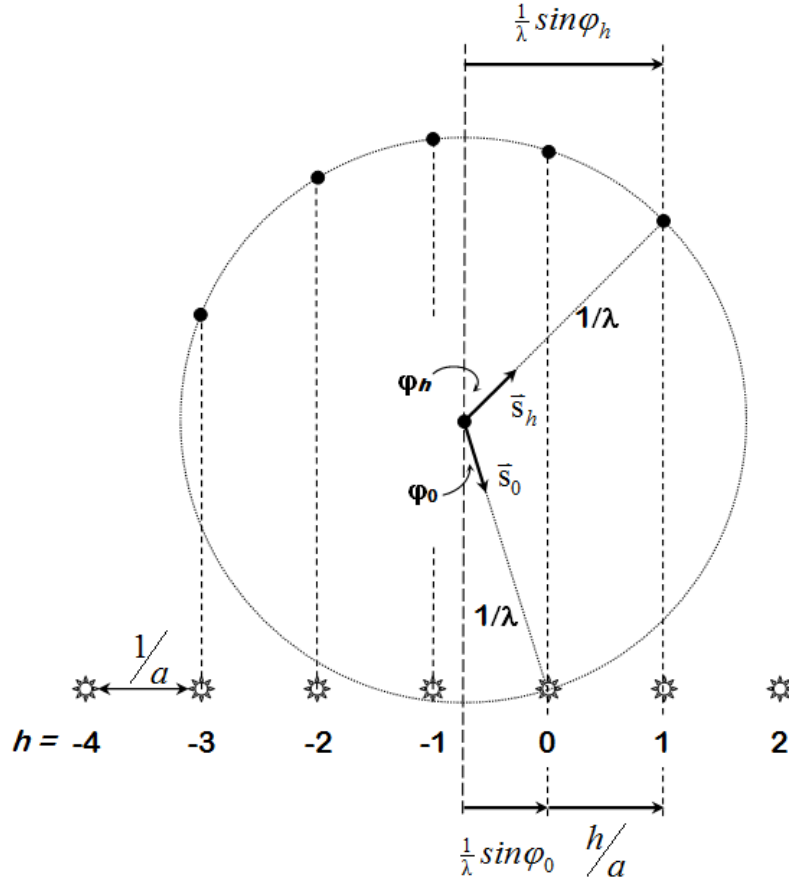
5.3 The LEED Pattern

- It will turn out that the LEED pattern is actually an image of the reciprocal lattice.
- Interpretation of the LEED pattern requires transforming the image of the reciprocal lattice back into the real lattice.

5.3.1 Ewald Sphere

- At the crystal surface there is a lattice of real atoms and a reciprocal lattice.

- Define an origin for the reciprocal lattice and then imagine a set of rods drawn perpendicular to the surface through the reciprocal lattice points.
- Finally, draw a sphere of radius $r = \frac{1}{\lambda}$ over the lattice centered at a point $\frac{-\hat{s}_0}{\lambda}$ from the origin of the reciprocal lattice.



- The directions in which one gets diffraction beams are given by the intersections of the reciprocal lattice rods with the sphere.

$$\frac{1}{\lambda} \sin \varphi_h = \frac{1}{\lambda} \sin \varphi_0 + \frac{h}{a} \quad (7)$$

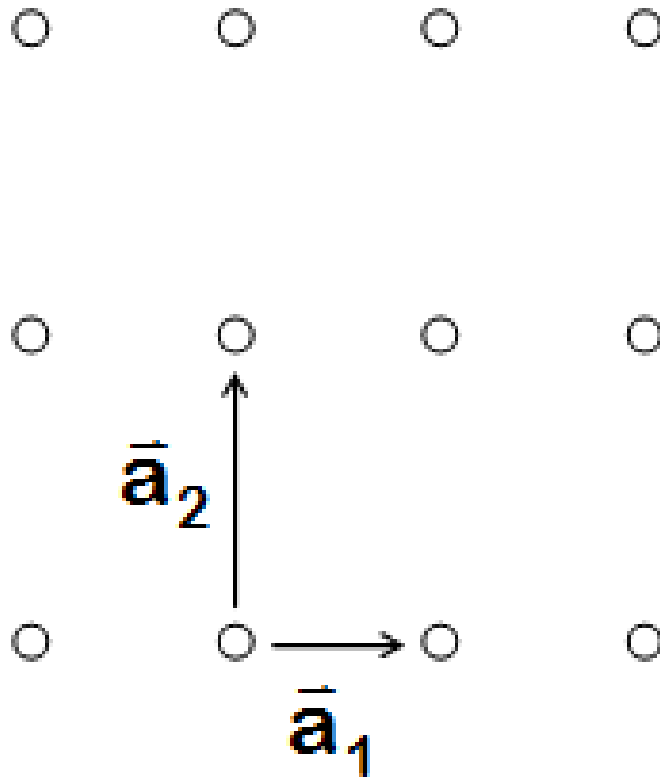
$$a \cdot (\sin \varphi_h - \sin \varphi_0) = h\lambda \quad (8)$$

- If you imagine a much larger sphere (the LEED) screen, then the diffracted beams are intersecting the screen and the image that one obtains is the projection of the screen onto a plane. The screen is a magnified version of the Ewald sphere and the image is then a magnified view of the reciprocal lattice.

- As the crystal is rotated the image on the screen will rotate also.
- As the energy of the beam increases, $\frac{1}{\lambda}$ increases and the number of beams intersecting the screen increases. What one sees is that the reflected beam stays put but the diffracted beams converge onto the reflected beam.
- From the image one can determine the real space lattice but nothing more. True structural information comes from the beam intensities.

5.3.2 LEED Patterns from Simple Clean Surfaces

- LEED patterns are simply images of the reciprocal lattice of the real space surface lattice.
- Given the rectangular lattice below, what does the LEED pattern look like?



The lengths of the real space lattice vectors are:

$$|\vec{a}_1| = 1 \text{ and } |\vec{a}_2| = 2$$

Let us use python to compute the reciprocal lattice vectors.

```

1 import numpy as np
2

```

```

3 A = np.matrix([[1, 0],
4               [0, 2]])
5
6 Astar = A.T.I          #(ati)
7 print Astar

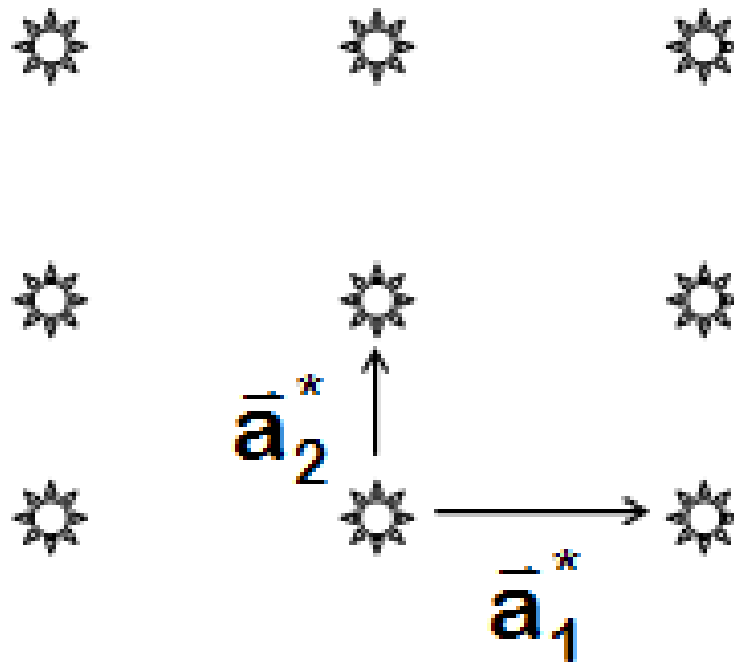
```

```

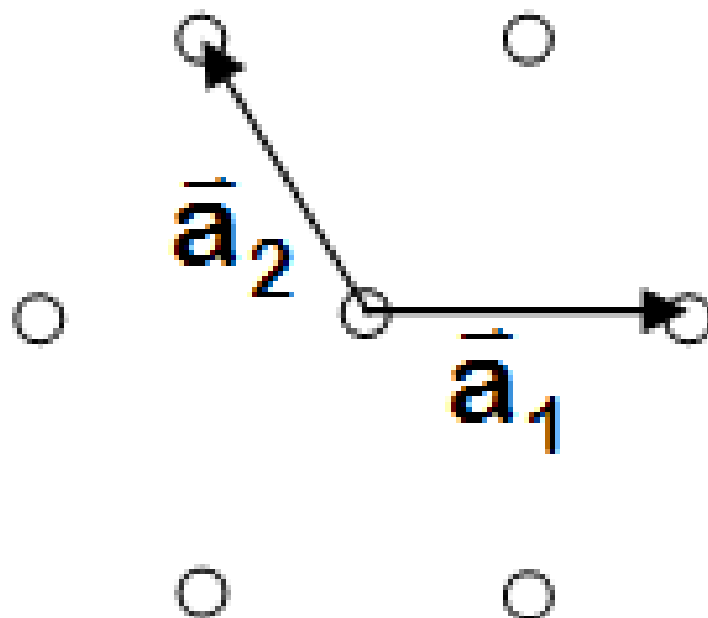
[[ 1.  0. ]
 [ 0.  0.5]]

```

The reciprocal lattice and therefore the LEED pattern look like:



- Given a hexagonal lattice below, what does the LEED pattern look like?



```

1 import numpy as np
2
3 A = np.matrix([[1, 0],
4                [-0.5, np.sqrt(3)/2]]) # note the vectors are in row form
5
6 Astar = A.T.I
7 print Astar

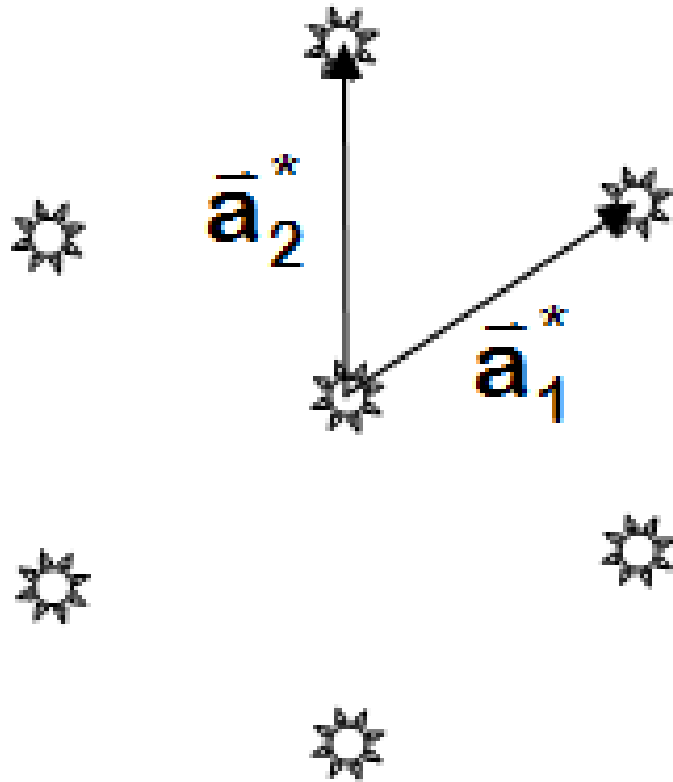
```

```

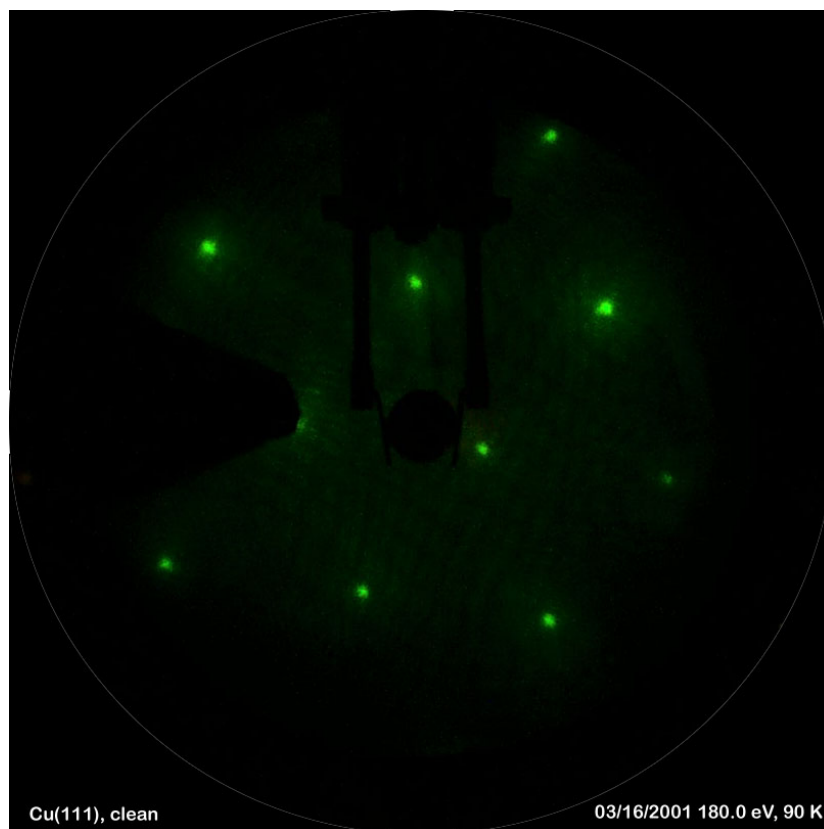
[[ 1.          0.57735027]
 [ 0.          1.15470054]]

```

The reciprocal lattice and therefore the LEED pattern look like



- Note that the reciprocal lattice is a true lattice. It extends infinitely in both directions, is periodic, and all points are equivalent.
A real LEED image of a clean Cu(111) surface looks like this:



5.4 Overlayers and Superlattices

- As noted earlier some surfaces can reconstruct such that the periodicity of the surface is different from that of the bulk termination plane.
 - When atoms or molecules are adsorbed on surfaces they tend to form well defined structures. They will adsorb in well defined site on the surface and often with long range order or periodicity which is commensurate with the substrate but with longer periodicity.
 - Adsorption sites are often high symmetry sites which optimize the bonding between adsorbate and the surface: on-top, bridging, fcc hollow, hcp hollow etc.
 - The ordered overlayers are actually lattices of atoms or molecules. If they are commensurate with the substrate this means that they have periodicity or lattice vectors that are linear combinations of the surface lattice vectors.

5.4.1 Overlayer Notations

- Adsorbed atoms also produce ordered overlayers with lattice parameters different from those of the substrate.

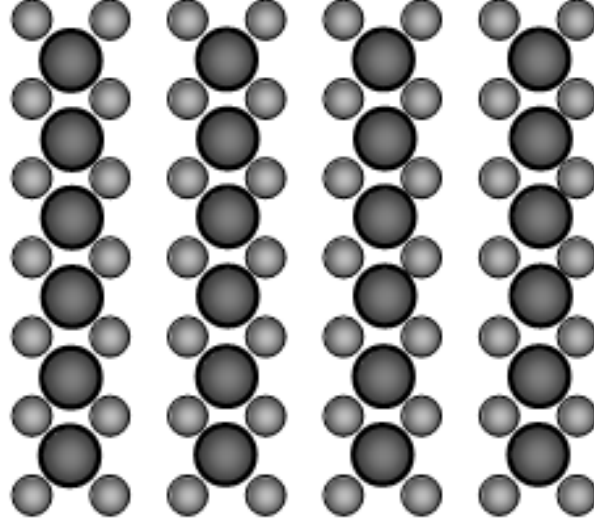


Figure 16: $p(2 \times 1)$ overlayer on a square lattice.

- The superlattice will have its own lattice vectors (\vec{b}_1, \vec{b}_2) which may or may not have some simple relationship with the substrate lattice vectors (\vec{a}_1, \vec{a}_2) .
- Those superlattices with lattice vectors that are separated by the same angle as the substrate lattice vectors can be referred to by Wood's notation.
- If the lengths are related by
 $|\vec{b}_1| = x|\vec{a}_1|$ and $|\vec{b}_2| = y|\vec{a}_2|$
and the angle between the vectors is φ , then the notation for the overlayer

is

$$(x \times y)R\varphi.$$

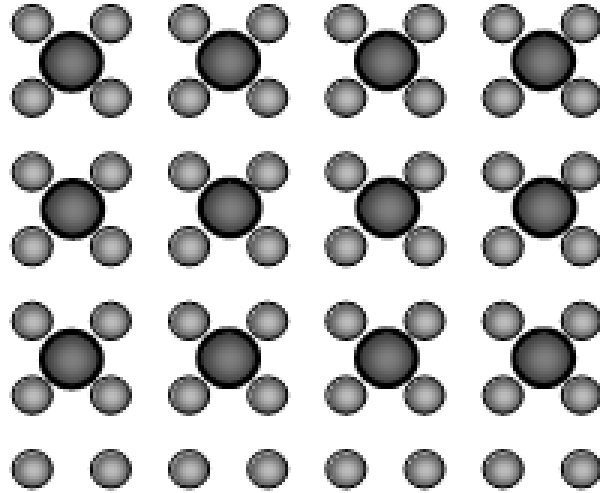


Figure 17: Example of $p(2 \times 2)$ on a square lattice.

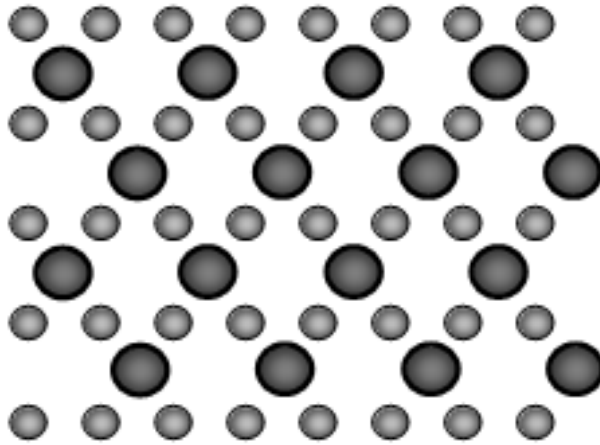


Figure 18: Example of a $c(2 \times 2)$ on a rectangular lattice.

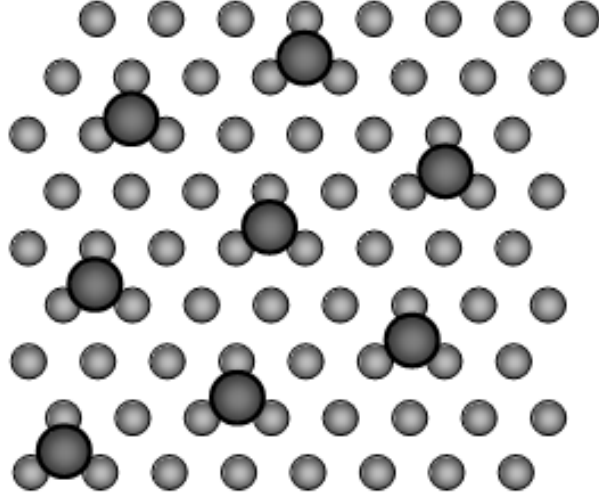


Figure 19: Example of a $(\sqrt{7} \times \sqrt{7})$ R19° on a hexagonal lattice.

- If the angle between the two sets of lattice vectors is not the same then it is not possible to define a Wood's notation for the superlattice.
- The relationship between the two lattices can always be given as

$$\vec{b}_1 = m_{11}\vec{a}_1 + m_{12}\vec{a}_2 \quad (9)$$

$$\vec{b}_2 = m_{21}\vec{a}_1 + m_{22}\vec{a}_2 \quad (10)$$

- The overlayer can then be denoted by a simple matrix which transforms the overlayer lattice into the substrate lattice.

$$\begin{pmatrix} \vec{b}_1 \\ \vec{b}_2 \end{pmatrix} = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{pmatrix} \vec{a}_1 \\ \vec{a}_2 \end{pmatrix} = \underline{\underline{M}} \begin{pmatrix} \vec{a}_1 \\ \vec{a}_2 \end{pmatrix}$$

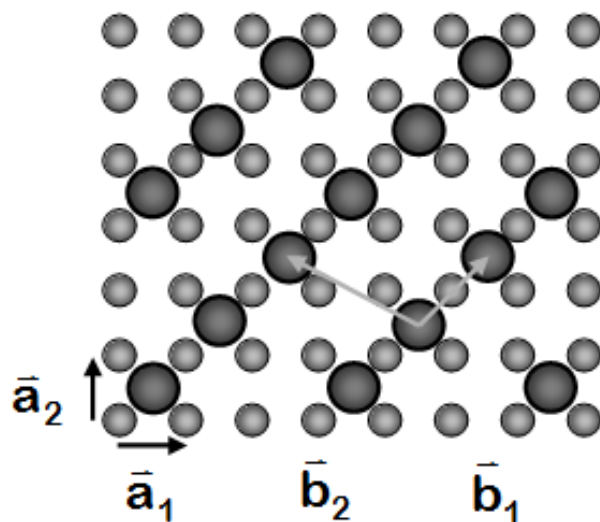


Figure 20: Example of a lattice with a structure requiring matrix notation.

In this example: $\vec{b}_1 = 1 \cdot \vec{a}_1 + 1 \cdot \vec{a}_2$
 and $\vec{b}_2 = -2 \cdot \vec{a}_1 + 1 \cdot \vec{a}_2$
 so:

$$M = \begin{bmatrix} 1 & 1 \\ -2 & 1 \end{bmatrix}$$

- Notice that there are several different choices of lattice vectors that can be used. All describe the same lattice.

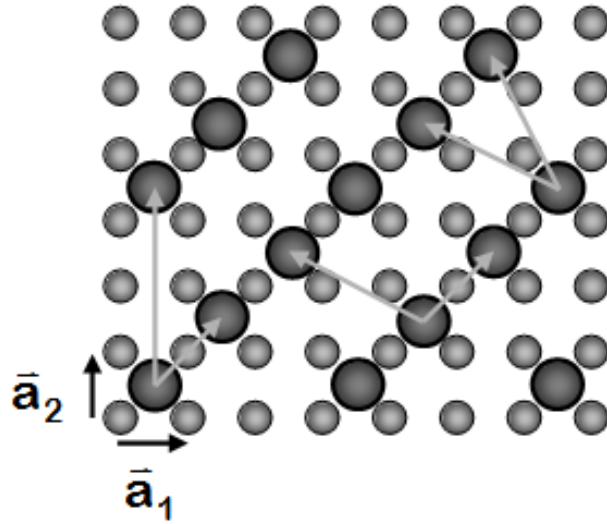
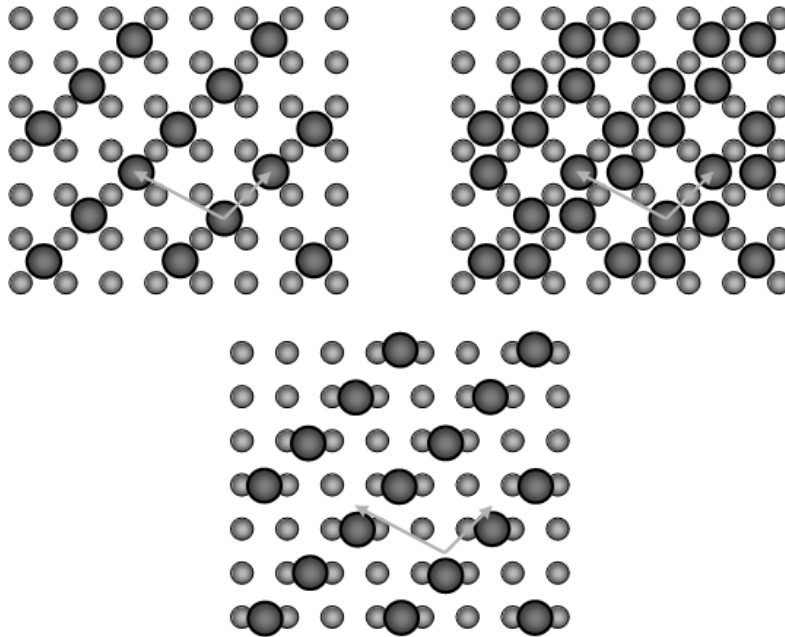


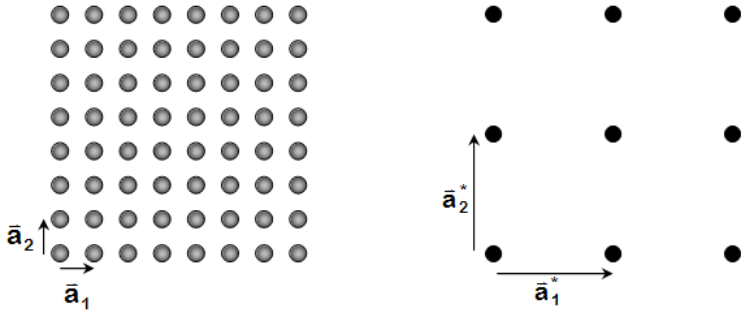
Figure 21: Equivalent lattice vectors for describing the same lattice.

- Also note that there are several different adsorbate overlayers that yield the same lattice. They may have different bases however.



5.4.2 LEED Patterns from superlattices

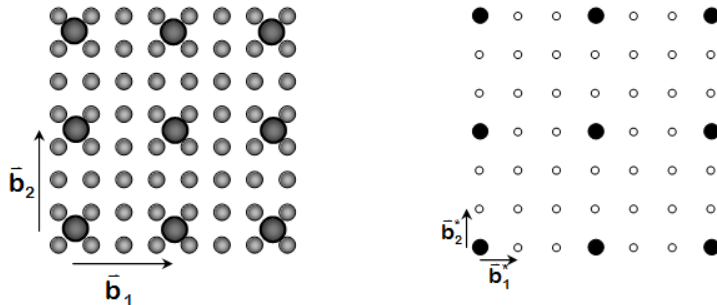
- LEED patterns are produced by diffraction from lattices at surfaces. The clean unreconstructed surface is said to give a (1×1) pattern. The LEED pattern would be the reciprocal lattice of the surface lattice vectors determined by termination of the bulk lattice.
- The LEED pattern is actually a picture of the reciprocal lattice.
- One of the primary objectives of the LEED experiment is to identify the periodicity of the overlayer lattice.
- Since there is a well defined relationship between the reciprocal lattice and the real lattice it is not hard to obtain the real space lattice parameters from the reciprocal space lattice.
- Example - $p(3 \times 3)$ on an fcc(100) surface.
 - The fcc(100) lattice is a square array and thus gives a diffraction pattern



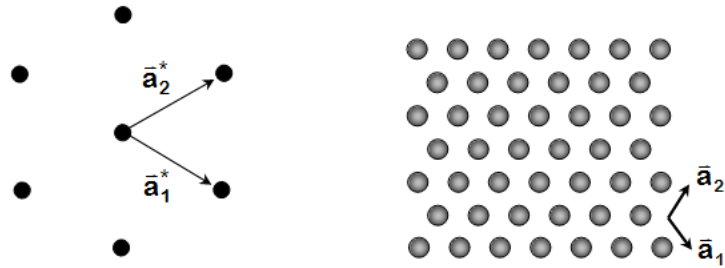
which is square.

$$\vec{b}_1 = 3\vec{a}_1$$

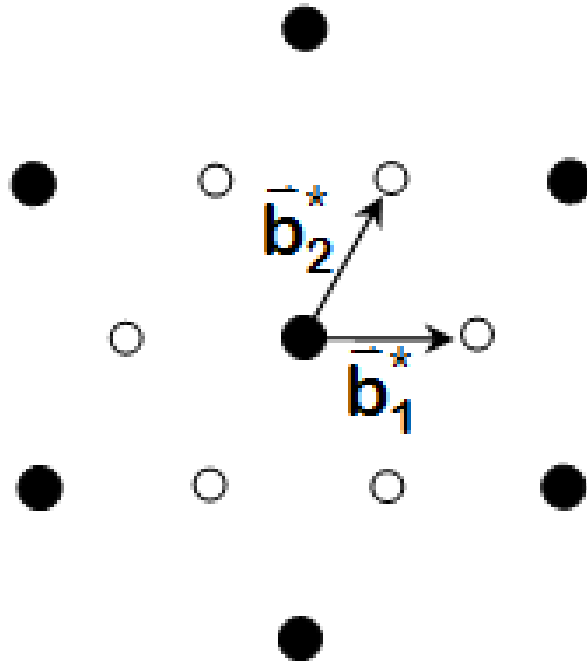
$$|\vec{b}_1^*| = \frac{1}{|\vec{b}_1|} = \frac{1}{3|\vec{a}_1|} = \frac{1}{3}|\vec{a}_1^*|$$



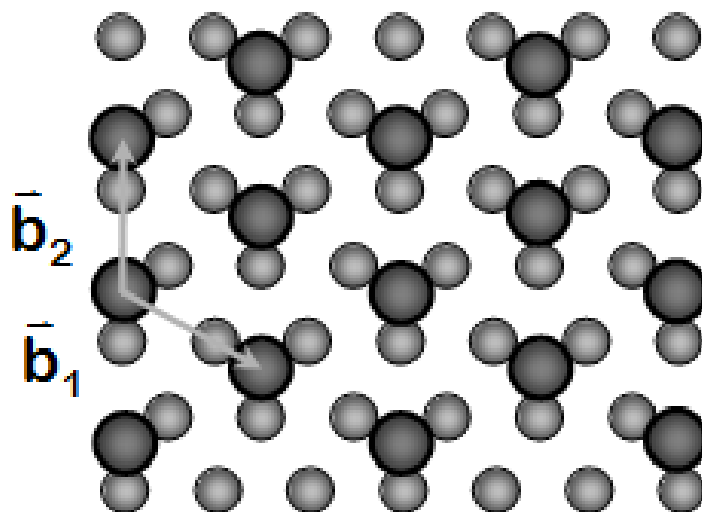
- Example - $(\sqrt{3} \times \sqrt{3})R30^\circ$ on an fcc(111) lattice.
- The fcc(111) lattice is hexagonal so its diffraction pattern has hexagonal symmetry.



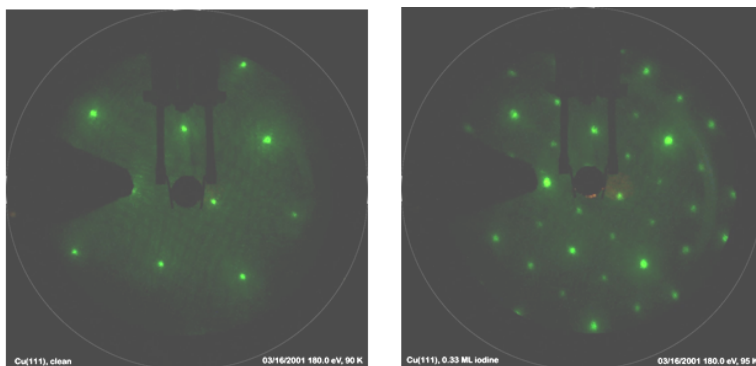
- An adsorbed layer of atoms on the surface gives the LEED pattern shown below. Note that the spots would all appear the same except that there are some new ones due to diffraction from the overlayer.



- This gives a structure such as the following for the real space lattice. It is a $(\sqrt{3} \times \sqrt{3})R30^\circ$ lattice.



Here is what an actual LEED pattern looks like for a clean surface of Cu(111) and a $(\sqrt{3} \times \sqrt{3})R30^\circ$ adsorbate overlayer.

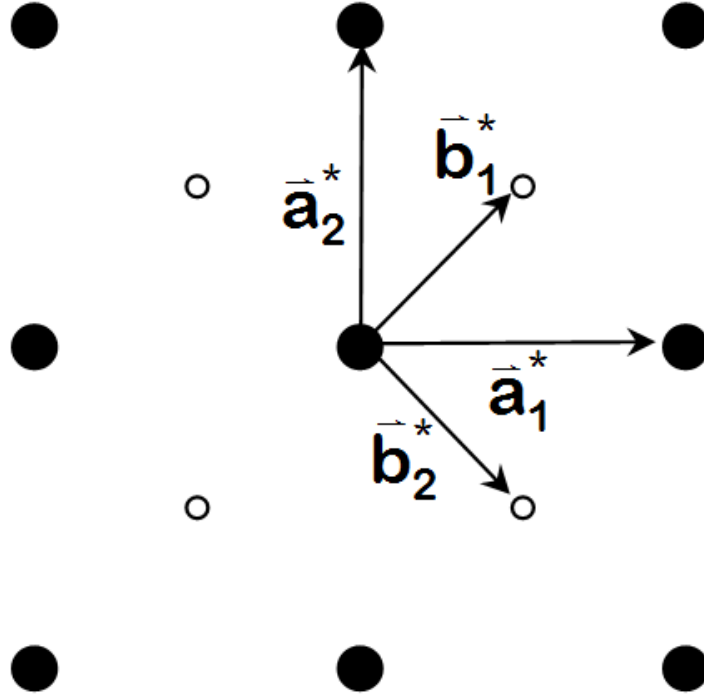


- Remember that the reciprocal lattice only gives you the real space lattice and not the positions of the atoms in the unit cells or the registry of the lattice with respect to the substrate.

5.5 General Reciprocal Space to Real Space Conversion

- There is a general means for converting a reciprocal lattice into a real space lattice.
 - The usual type of problem is one in which the substrate lattice (1×1) is known and what one is trying to do is to determine the unit cell of the overlayer in terms of the substrate lattice vectors.
 - Basically the problem is to find M , the matrix which relates the overlayer lattice to the surface lattice.

5.5.1 Example - LEED of $(\sqrt{2} \times \sqrt{2})R45^\circ$ on an fcc(100) lattice.



- Find the reciprocal lattice vectors from the LEED pattern. You can usually tell which pattern belongs to the surface atoms; they are the ones with the longest spacings (because they are the closest together).

$$\vec{b}_1^* = \frac{1}{2}\vec{a}_1^* + \frac{1}{2}\vec{a}_2^*$$

$$\vec{b}_2^* = \frac{1}{2}\vec{a}_1^* - \frac{1}{2}\vec{a}_2^*$$

- Find the reciprocal space overlayer matrix.

$$M^* = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

- Find the real space overlayer matrix from the reciprocal space overlayer matrix.

$$M = [(M^*)^T]^{-1}$$

- Use the real space matrix to find the unit cell of the overlayer lattice.

$$b = M \begin{bmatrix} \vec{a}_1 \\ \vec{a}_2 \end{bmatrix}$$

```

1 import numpy as np
2
3 Mstar = 0.5 * np.array([[1, 1], [1, -1]])
4
5 M = np.linalg.inv(Mstar.T)
6 print M

```

```

[[ 1.  1.]
 [ 1. -1.]]

```

- Note that all we know is the scaling and the orientation of the overlayer unit cell with respect to the substrate lattice. We do not know the registry or the contents of the unit cell.

- Note that the area of the unit cell is $A = 2$. In other words it contains two substrate atoms.

- In terms of the area of the substrate lattice the superlattice area is

$$area = |\vec{b}_1 \times \vec{b}_2|$$

- The coverage of the adsorbate must be some multiple of the inverse unit cell area.

$$\theta = n/A$$

- In the case of the $(\sqrt{2} \times \sqrt{2})R45^\circ$ lattice the coverage is most probably

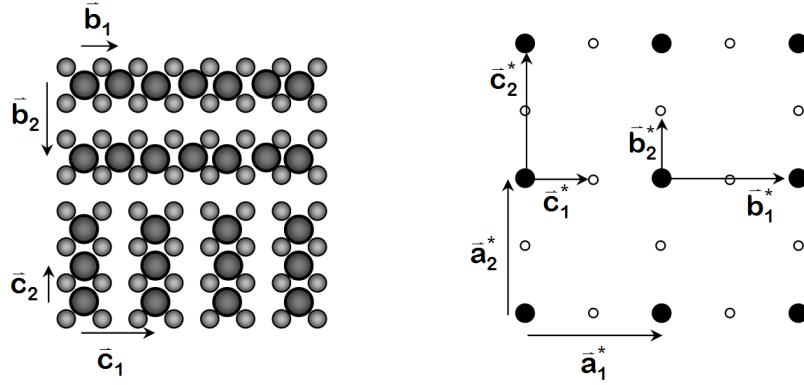
$$\theta = 1/2$$

although some independent means is needed to determine this.

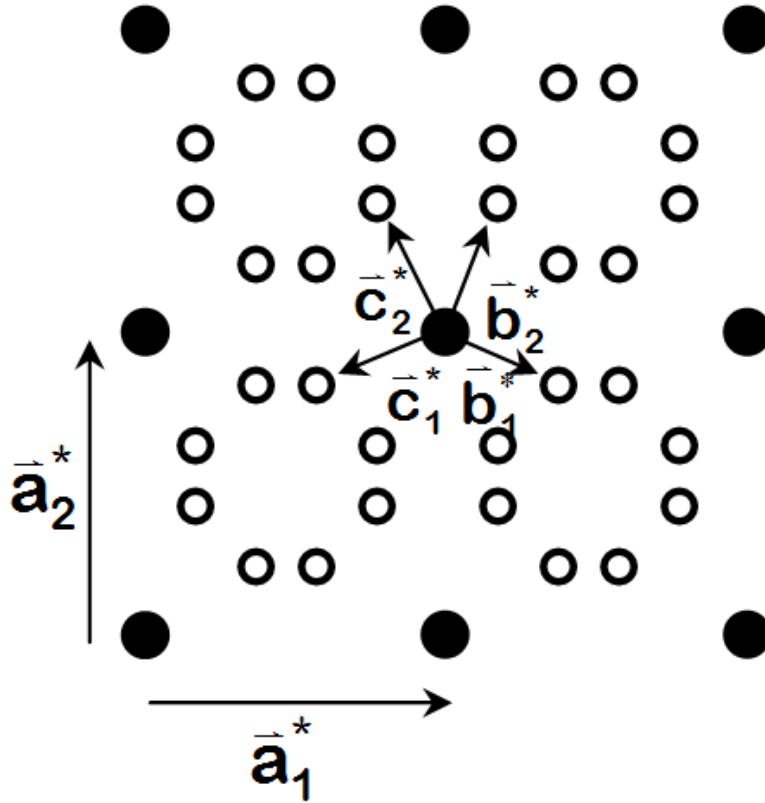
5.6 Domains

- It is very often the case that because of symmetry of the substrate multiple orientations of the superlattice are energetically equivalent.
- The presence of one or more domains can give two superimposed LEED patterns. This complicates the analysis of the LEED pattern to get the reciprocal lattice.

5.6.1 Example - (2×1) overlayer on a square fcc(100) lattice.



5.6.2 Example - $(\sqrt{5} \times \sqrt{5})R26^\circ$ on square lattice. Oxygen on Mo(100) at $\theta = 0.6$.



The reciprocal lattice vectors are given by:

$$\vec{b}_1^* = \frac{2}{5}\vec{a}_1^* - \frac{1}{5}\vec{a}_2^*$$

$$\vec{b}_2^* = \frac{1}{5}\vec{a}_1^* + \frac{2}{5}\vec{a}_2^*$$

This gives the reciprocal lattice matrix as:

```

1 import numpy as np
2
3 Mstar = 1. / 5. * np.array([[2, -1],
4                             [1,  2]])
5
6 M = np.linalg.inv(Mstar.T)
7 print M

```

```

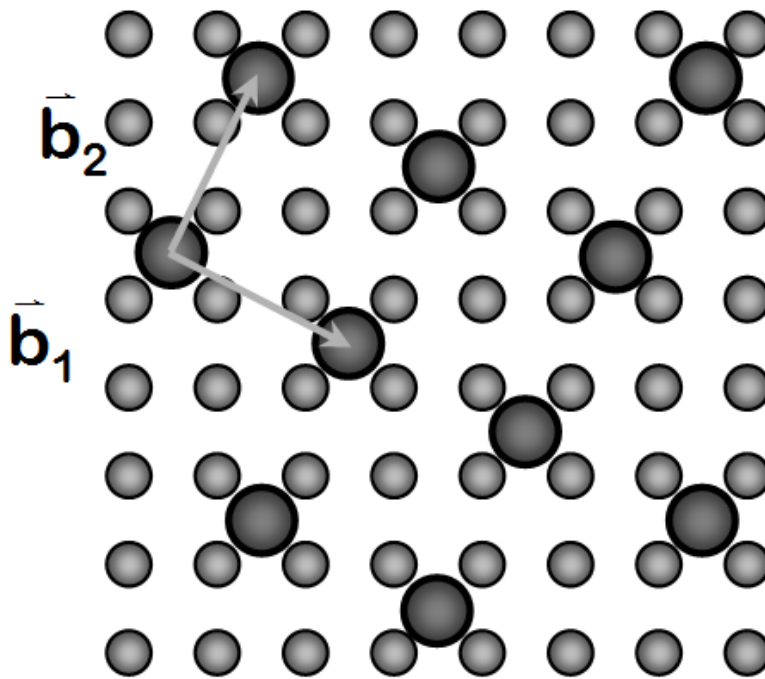
[[ 2. -1.]
 [ 1.  2.]]

```

The real space lattice vectors are:

$$\vec{b}_1 = 2\vec{a}_1 - \vec{a}_2$$

$$\vec{b}_2 = \vec{a}_1 + 2\vec{a}_2$$



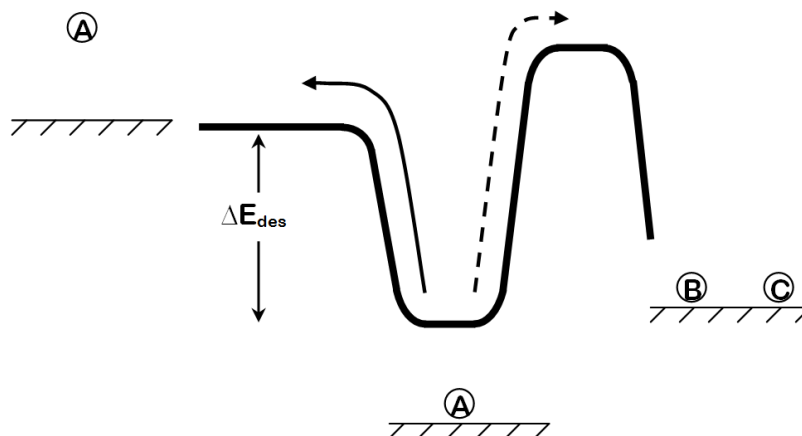
6 Thermal desorption methods

- Thermal desorption methods are the most widely performed experiment in surface science to study the energetics of adsorption and the kinetics of surface reactions.
- Thermal desorption methods are also used widely in the field of catalysis and in the study of other surface processes such as adsorption into porous solids.

6.1 TPD - Reversible Adsorption / Desorption

- The measurement of desorption kinetics is usually performed using an experiment called Temperature Programmed Desorption (TPD).

- When molecules adsorb and then desorb reversibly without reacting on the surface, TPD can be used to measure the desorption energy.
- Reversible desorption basically assumes that the desorption energy is lower than any barrier to reaction and thus it is the preferred reaction channel for a molecule on a surface.



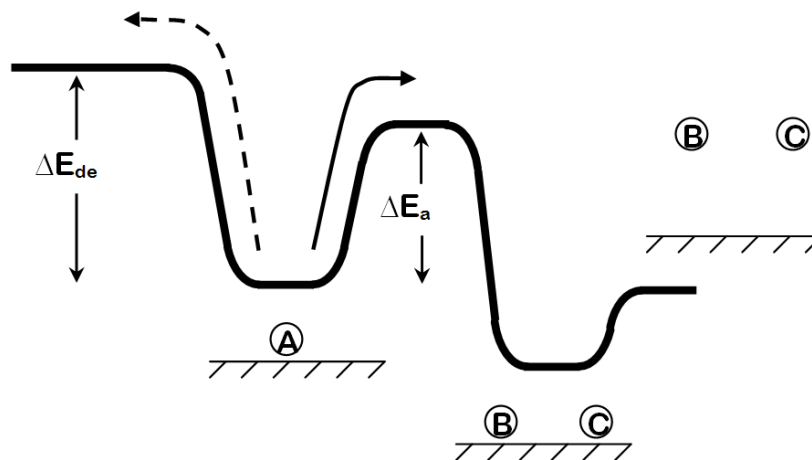
- The TPD experiment measures the rate of desorption. With some analysis, we can estimate the rate constant for desorption.

$$k_{des} = \nu \exp\left(\frac{-\Delta E_{des}^{\ddagger}}{RT}\right)$$

If there is no activation barrier to *adsorption* then the barrier to desorption, $\Delta E_{des}^{\ddagger}$, is the same as the desorption energy, ΔE_{des} . In many cases the barriers to *adsorption* are quite negligible.

6.2 TPRS – Irreversible Surface Reactions

- Temperature Programmed Reaction Spectroscopy (TPRS) is used to measure reaction kinetics and identify products of surface reactions other than simple desorption.
- If adsorbed molecules react rather than desorb from the surface, then it is not possible to use the TPD experiment to measure desorption energies. This occurs when the barriers to reaction are lower than the barriers to desorption.

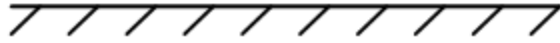


- Many adsorbed molecules do react and it is possible that one can detect the desorption of products from the surface. This can provide both an identification of the reaction products and a measure of the barrier to surface reaction ΔE_a .

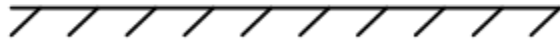
6.3 TPD Experiment

- The typical TPD experiment is performed in four steps.
 1. Surface preparation
 2. Adsorption of species at T_0 .
 3. Heating of surface at constant rate, $T = T_0 + \beta t$.
 4. Detection of desorbing species by mass spectrometry.

1

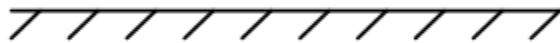


2



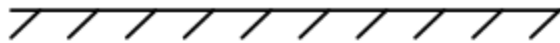
T_0

3



$T = T_0 + \beta t$

4



high



- The TPD experiment is performed in a vacuum so that the mass spectrometer can operate. If one does not use a mass spectrometer as a detector the experiment can also be done in flowing gas.

- The mass spectrometer is tuned to the mass or masses of species that are desorbing from the surface and measures a signal versus time (or temperature). The signal that is measured by the mass spectrometer is proportional to the partial pressure of the desorbing material which in turn is proportional to the desorption rate (provided that the desorbing species are being pumped away rapidly.)

$$signal(T) \propto P(T) \propto r(T)$$

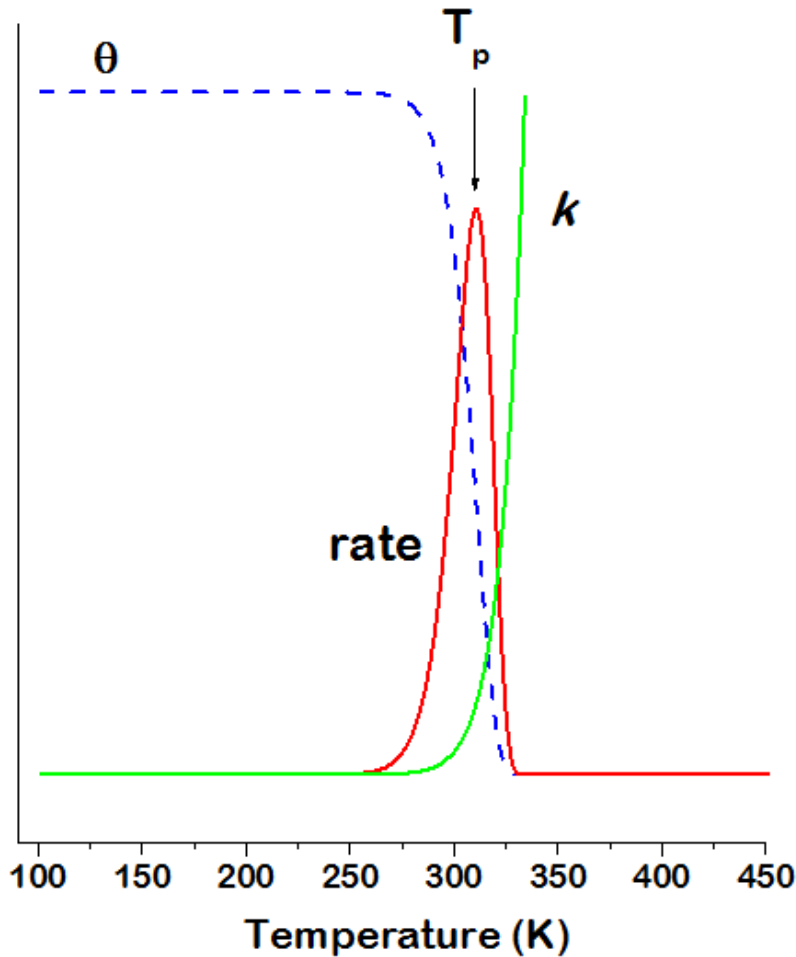
- The rate of desorption is defined as the change in coverage with respect to time.

$$r = -\frac{d\theta}{dt} = -\beta \frac{d\theta}{dT}$$

- For most process one thinks about the desorption rate simply as a reaction which can be zero-, first-, or second-order in the reactant concentration.

$$r = k \cdot \theta^n$$

- One can think about the desorption experiment physically.
- Initially one has a high coverage θ_0 of adsorbates but the rate constant $k(T_0)$ is very low. Thus the desorption rate is low.
- As the temperature increases the rate constant increases and thus the desorption rate increases.
- Because desorption is occurring the coverage drops. At some temperature the rate of desorption passes through a maximum and then drops back to zero.



- The most common form of desorption is for a first-order process. This is what one would expect for a molecule that simply adsorbs and then desorbs from the surface.
- Zero-order desorption can occur for thick films which in essence sublime from the surface.
- Second-order desorption occurs in cases where molecules adsorb dissociatively but then recombine in order to desorb. ($\text{H}_{2,\text{g}} \rightarrow 2\text{H}_{\text{ad}} \rightarrow \text{H}_{2,\text{g}}$).

6.4 Analysis of First-Order Desorption Kinetics

- We will solve for the peak desorption temperature for a first-order desorption process. This can be used to determine the desorption energy.
 - Begin with the definition of the desorption rate as

$$r_d = -\frac{d\theta}{dt} = k\theta = \nu \exp\left(\frac{E_d}{RT}\theta\right)$$

- In terms of temperature we have

$$-\frac{d\theta}{dT} = \frac{\nu}{\beta} \exp\left(\frac{E_d}{RT}\theta\right)$$

- The maximum in the desorption rate is defined by the expression

$$\frac{dr_d}{dT}\bigg|_{T_p} = 0$$

- This can be evaluated using the form of the expression for the desorption rate.

$$-\frac{dr_d}{dT} = \nu \left[\exp\left(\frac{-E_d}{RT}\right) \frac{d\theta}{dT} + \frac{E_d}{RT^2} \exp\left(\frac{-E_d}{RT}\right) \theta \right]$$

At the peak desorption temperature this is equal to zero and gives

$$\frac{E_d}{RT_p^2} \theta_p = -\frac{d\theta}{dT}\bigg|_{T_p}$$

which leads to this final equation relating the peak temperature, desorption barrier and pre-exponential factor.

$$\frac{E_d}{RT_p^2} - \frac{\nu}{\beta} \exp\left(\frac{E_d}{RT_p}\right)$$

Which must be solved numerically. Note we have a single equation, and two unknowns: E_d and ν .

- In order to solve for E_d from a single measured value of T_p one must estimate the value of the pre-exponent. Formally this comes from transition state theory and can be given by the expression:

$$\nu = \left(\frac{k_b T}{h} \frac{q_{\ddagger}}{q_A} \right)$$

The values of q are the partition functions for the adsorbed species (q_A) and for the transition state for desorption (q_{\ddagger}). In many instances these roughly cancel out and one can assume that the pre-exponent is given by

$$\nu \approx \left(\frac{k_b T}{h} \right) \approx 10^{13} \text{ sec}^{-1}$$

- One can estimate the values of E_d and ν independently if one measures the desorption profiles over a range of different heating rates. The peak desorption temperature will depend upon the heating rate and increases for faster heating rates.

- Examining the expression for the relationship between desorption energy and peak temperature gives the following.

$$\frac{E_d}{RT_p^2} = \frac{\nu}{\beta} \exp\left(\frac{-E_d}{RT_p}\right) \quad (11)$$

$$\frac{\beta}{T_p^2} = \frac{\nu R}{E_d} \exp\left(\frac{-E_d}{RT_p}\right) \quad (12)$$

$$\ln\left(\frac{\beta}{T_p^2}\right) = \ln\left(\frac{\nu R}{E_d}\right) - \frac{E_d}{RT_p} \quad (13)$$

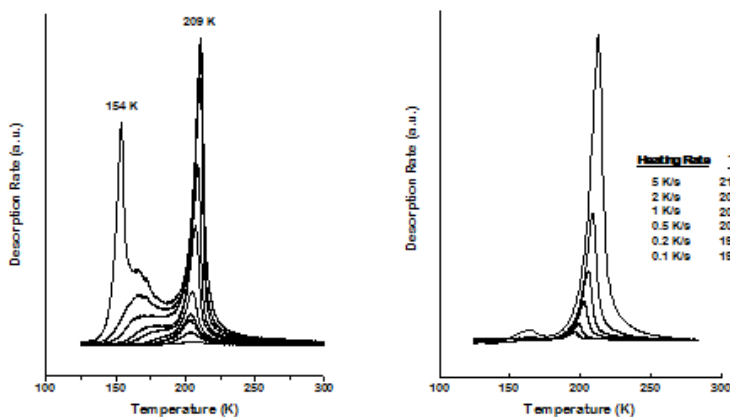
If one obtains T_p for various different heating rates and then plots

$\ln\left(\frac{\beta}{T_p^2}\right)$ vs $\frac{1}{T_p}$

this will yield a straight line of slope E_d/R and the intercept contains information about the preexponential factor.

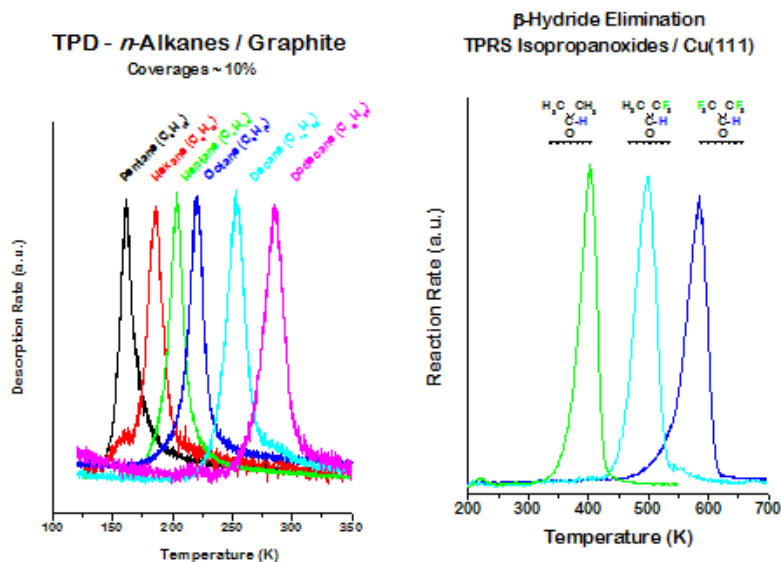
6.4.1 Examples of first-order desorption curves.

- The plot is for heptane adsorbed on a graphite surface. The desorption is first order and reversible. Note that the peak desorption temperature is independent of coverage. Once the monolayer desorbs one has desorption from the multilayer which occurs at lower temperatures.
- The second plot shows the desorption of heptane from graphite using variable heating rates. Note that the desorption temperature increases with increasing heating rate. This can be used to determine the desorption energy independent of the pre-exponent.



- The third plot shows the desorption peaks for alkanes of increasing chain length. Clearly the temperature increases roughly linearly with the chain length and indicates that the alkanes adsorb with desorption energies that increase linearly with chain length.

- The fourth plot is actually a TPRS curve for a reaction of isopropanoxides that produce acetaldehydes that desorb from the surface during heating.

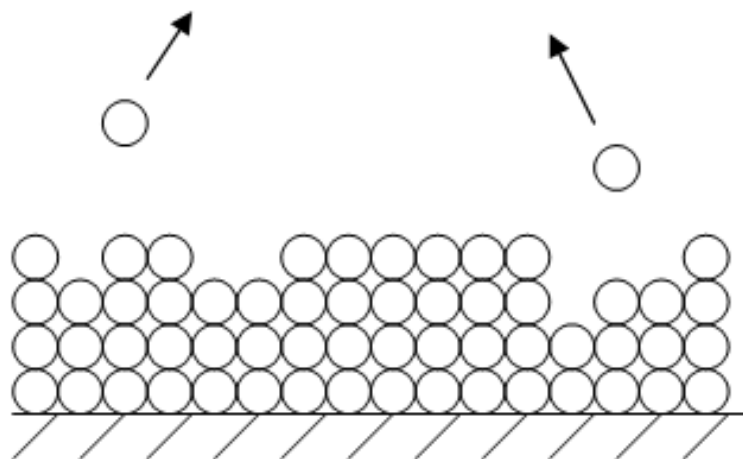


6.5 Zero order Desorption Kinetics

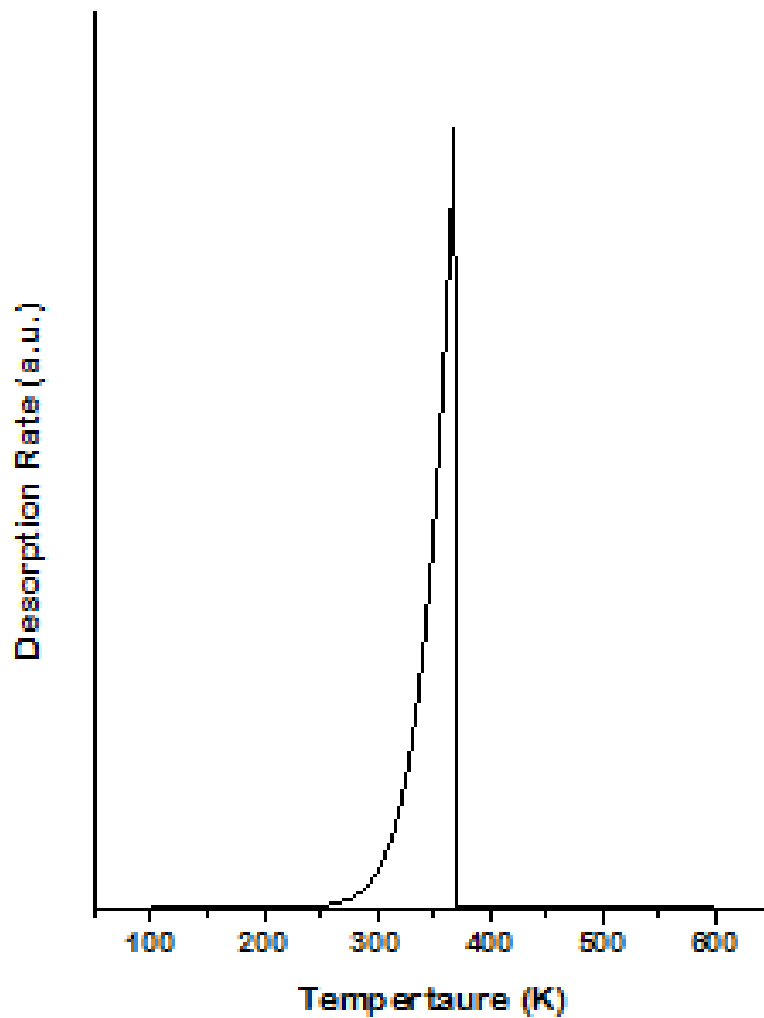
- Zero-order desorption kinetics are described by an equation of the form

$$r = k\theta^0 = \nu \exp\left(-\frac{E_d}{RT}\right)$$

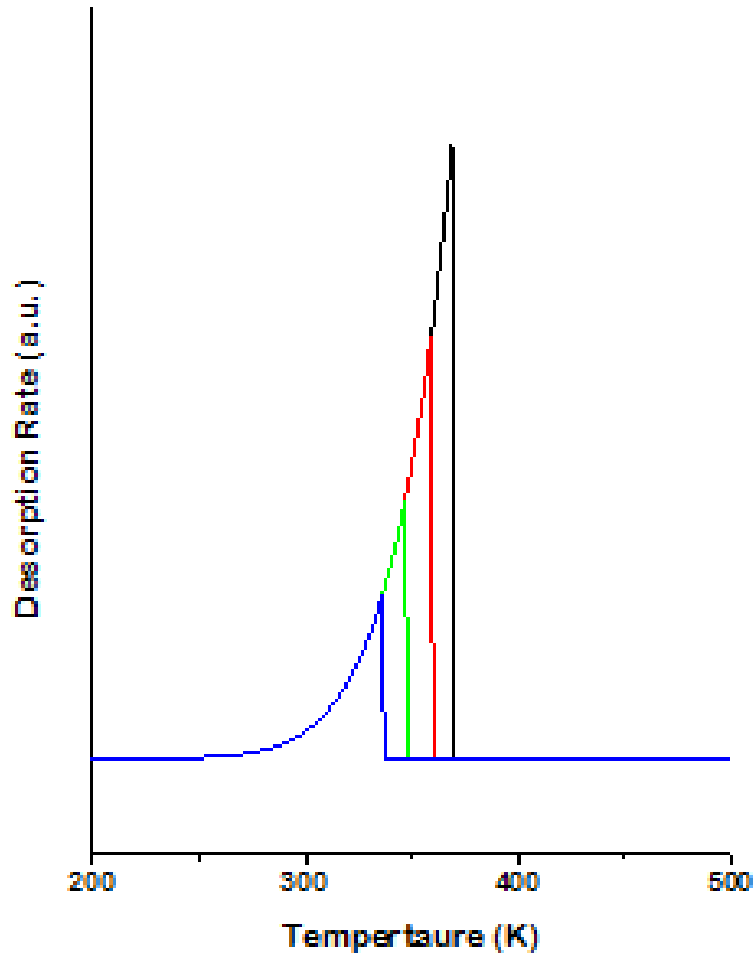
- Zero-order desorption kinetics usually occur under conditions where the concentration of species at the surface does not change during the desorption process. As an example, consider the sublimation of a solid or of a thin film. The desorption of one molecule from the surface merely exposes a molecule in the next layer for desorption.



- During a TPD experiment one measures the desorption rate directly. Because there is no coverage dependence, the TPD curve is directly proportional to the desorption rate constant plotted versus temperature.



- During a zero-order desorption process the desorption rate continues to increase with temperature until all the adsorbate has been removed from the surface. At that temperature the desorption rate drops to zero.
- The area under the desorption curve is proportional to the amount of material on the surface initially. As a result increasing the coverage on the surface simply causes the peak temperatures to increase. At low temperatures the curves all overlap one another because the desorption rate is just proportional to the desorption rate constant.



6.6 Second-Order Desorption Kinetics

- Second order desorption usually describes recombinative desorption of adsorbates such as H_2 , O_2 , N_2 , etc. . .
 - The desorption rate will be second-order if the desorption of the molecule is fast with respect to the recombination rate.
 $\$ \text{insert math here} \$$
 - Second order desorption kinetics can be analyzed in much the same way as first order kinetics to give a relationship between the peak desorption temperature, T_p , and the desorption energy, E_d .

$$r_d = k\theta^2 \quad (14)$$

$$= -\beta \frac{d\theta}{dT} \quad (15)$$

$$TODO : muchmoretoadd \quad (16)$$

- Note that the relationship between the peak temperature and the desorption energy now depends on the coverage of the adsorbate.

- The second order desorption peak is symmetric about the peak and so it is easy to show that

$$\theta_p = \frac{1}{2}\theta_0$$

and as a consequence that the relationship between the peak desorption temperature and the desorption energy depends on the initial coverage on the surface.

$$\left(\frac{-E_{des}}{RT_p^2}\right) = \frac{\nu}{\beta} \theta_0 \exp\left(\frac{E_{des}}{RT_p^2}\right)$$

6.7 Assumptions in TPD Analysis

- There are several assumptions that go into the analysis of the TPD spectra that must be recognized.

- *Equilibrium.* It is assumed that the molecules on the surface reach their equilibrium state and thus that the desorption energy that is measured represent a true thermodynamic property of the system. Because the system is heated slowly during the process it is fairly easy for the system to reach equilibrium but not always the case.

- *Pre-exponent.* It is often the case that it is difficult to get good data for the variable heating rate experiment and that one has to then analyze the data under the assumption that the pre-exponent is 10^{13} sec^{-1} . This is a rough estimate but the important thing is that the measured desorption energy is not very sensitive to errors in this value. One order of magnitude error in ν will only change E_{des} by ~5%.

- *Coverage dependent desorption energy.* The assumption is that there is one value of E_{des} in the experiment. Often there can be repulsive interactions between the adsorbates that result in an increase in the E_{des} as the molecules desorb from the surface.

[./ insert tpd3 here

7 Worked examples

7.1 Analysis of XPS spectra

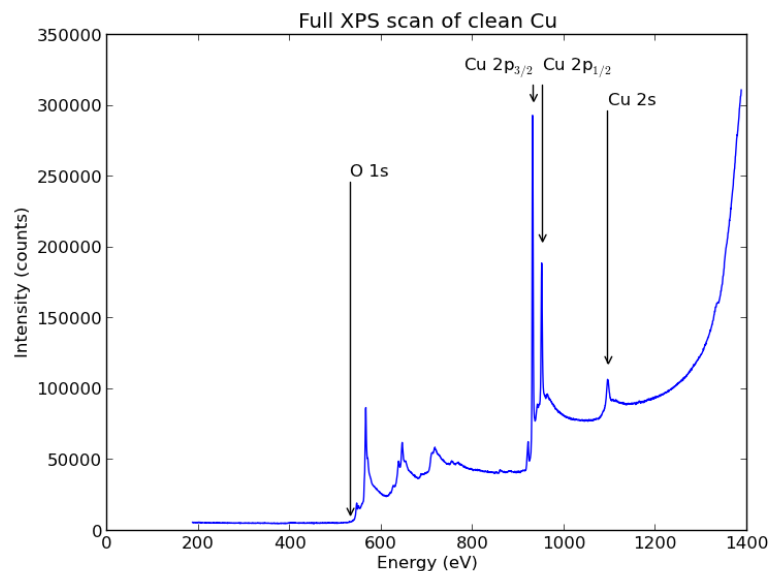
In this example, we examine XPS spectra for a clean Cu surface, and one that has been exposed to 30L of oxygen. Our objective is to calculate the atomic

% of O on the surface. First, we will examine the full spectrum of the clean surface, and two subregions of the clean spectrum. The data for these spectra is stored in text files.

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 E, C = np.loadtxt('data/Cu-clean-full.txt', skiprows=5, unpack=True)
5 plt.plot(E, C)
6 plt.xlabel('Energy (eV)')
7 plt.ylabel('Intensity (counts)')
8 plt.title('Full XPS scan of clean Cu')
9 adict = dict(arrowstyle="->",
10             connectionstyle="angle,angleA=0,angleB=90,rad=10")
11 plt.annotate('Cu 2s', (1096, 115000), (1096, 300000), arrowprops=adict)
12 plt.annotate('Cu 2p1/2', (952.20, 201000), (952.2, 325000), arrowprops=adict)
13 #plt.annotate('Cu 2p', (942.42, 0), (942.42, 225000), arrowprops=adict)
14 plt.annotate('Cu 2p3/2', (932.67, 300000), (932.67, 325000), arrowprops=adict, ha='right')
15
16 plt.annotate('O 1s', (532, 8000), (532, 250000), arrowprops=adict)
17
18 plt.savefig('images/XPS-Cu-clean-full.png')
19 plt.show()

```



Now, we examine the region associated with the Cu 2p signals.

```

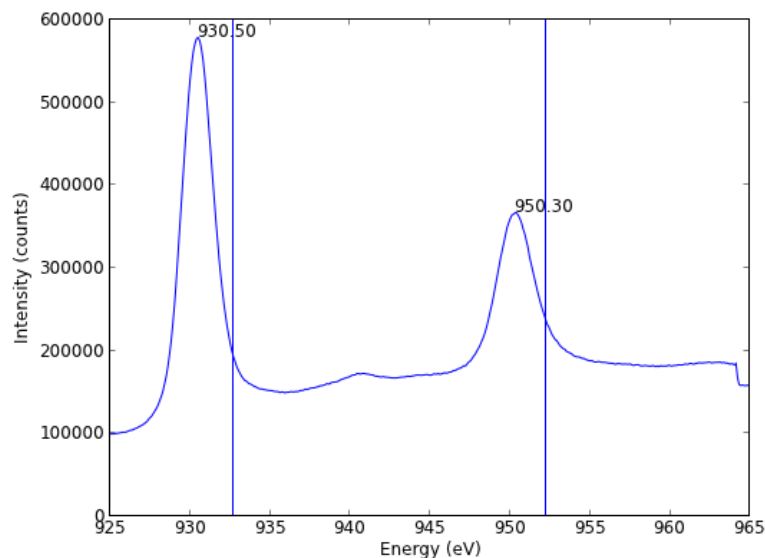
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 E, C = np.loadtxt('data/Cu-clean-region1.txt', skiprows=5, unpack=True)
5 plt.plot(E, C)
6 plt.axvline(952.20, 0, 600000, 'k--')
7 plt.axvline(932.67, 0, 600000, 'k--')
8
9 p1_ind = E < 935

```

```

10 p2_ind = (E > 945) & (E < 955)
11
12 # Annotate the two peaks
13 i1max = np.argmax(C[p1_ind])
14 plt.annotate('{0:1.2f}'.format(E[p1_ind][i1max]), (E[p1_ind][i1max], C[p1_ind][i1max]))
15
16 i2max = np.argmax(C[p2_ind])
17 plt.annotate('{0:1.2f}'.format(E[p2_ind][i2max]), (E[p2_ind][i2max], C[p2_ind][i2max]))
18
19 # plot the data
20 plt.xlabel('Energy (eV)')
21 plt.ylabel('Intensity (counts)')
22 plt.savefig('images/XPS-Cu-clean-region1.png')
23 plt.show()

```



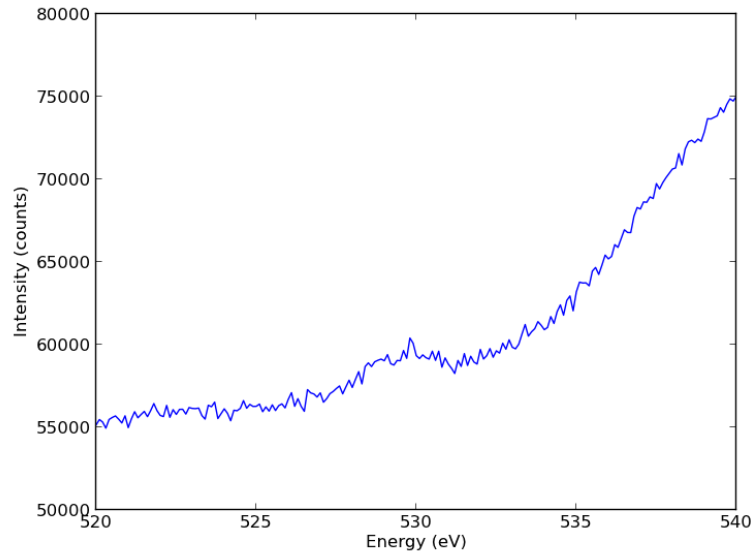
Note that when we zoom in, and look at the spectrum, the peaks are not exactly at the place predicted by the Handbook. This is because the energy scale is not calibrated for the work function of the spectrometer, which introduces a constant shift in the energy scale. To be quantitative in energy scale, we must use a standard to calibrate this energy scale. Here it does not matter, so we skip that step.

Finally, we examine the region where the oxygen peak is expected.

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 E, C = np.loadtxt('data/Cu-clean-region2.txt', skiprows=5, unpack=True)
5 plt.plot(E,C)
6 plt.xlabel('Energy (eV)')
7 plt.ylabel('Intensity (counts)')
8 plt.savefig('images/XPS-Cu-clean-region2.png')

```

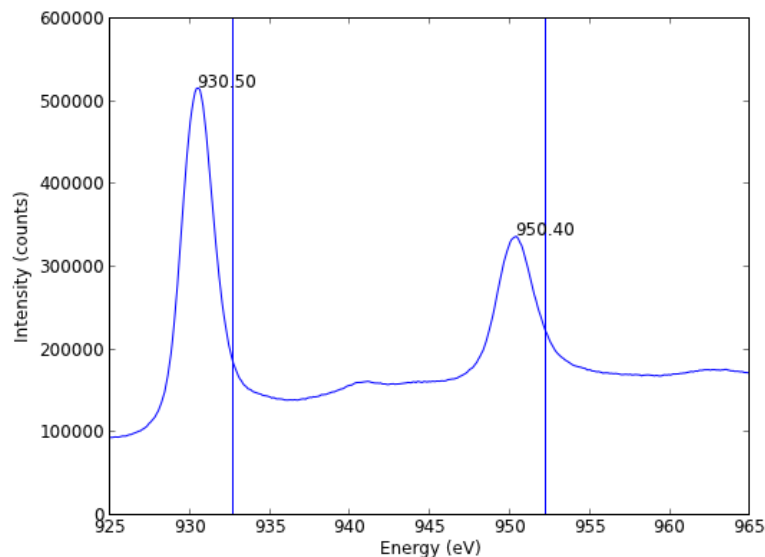


There is only a trace amount of oxygen present in this spectrum. Now we examine the spectra after exposing the surface to 30L of oxygen.

```

1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  E, C = np.loadtxt('data/Cu-0-30L-region1.txt', skiprows=5, unpack=True)
5  plt.plot(E,C)
6
7  plt.axvline(952.20, 0, 600000, 'k--')
8  plt.axvline(932.67, 0, 600000, 'k--')
9
10 # Annotate the two peaks
11 p1_ind = E < 935
12 p2_ind = (E > 945) & (E < 955)
13
14 i1max = np.argmax(C[p1_ind])
15 plt.annotate('{0:1.2f}'.format(E[p1_ind][i1max]), (E[p1_ind][i1max], C[p1_ind][i1max]))
16
17 i2max = np.argmax(C[p2_ind])
18 plt.annotate('{0:1.2f}'.format(E[p2_ind][i2max]), (E[p2_ind][i2max], C[p2_ind][i2max]))
19
20
21 plt.xlabel('Energy (eV)')
22 plt.ylabel('Intensity (counts)')
23 plt.savefig('images/XPS-Cu-0-30L-region1.png')
24 plt.show()

```



You can see there is hardly any change in the position of the Cu peaks, suggesting no surface oxidation has taken place, even though there is oxygen on the surface. The peak labels above are simply the energy of the highest intensity data point, not the energy of the maximum of a fitted peak. We need to integrate the areas under these curves to quantify the amount of Cu present. We choose the first peak because it is larger. We have to define a background to subtract off. We do that by fitting a polynomial to the region before and after the peak. Then we subtract the area under the polynomial from the area under the curve.

```

1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  E, C = np.loadtxt('data/Cu-0-30L-region1.txt', skiprows=5, unpack=True)
5
6  # peak 1
7  ind = E < 938
8
9  E1 = E[ind]
10 C1 = C[ind]
11
12 bg = (E1 < 927) | (E1 > 936) # indices of the background region
13
14 pb = np.polyfit(E1[bg], C1[bg], 3)
15 bgfit = np.polyval(pb, E1)
16
17
18 plt.plot(E1, C1)
19 plt.plot(E1[bg], C1[bg], 'g. ')
20 plt.plot(E1, bgfit, 'g-')
21 plt.savefig('images/XPS-Cu-0-bg-1.png')
22
23 I1 = -np.trapz(C1, E1)
24 Ib = -np.trapz(bgfit, E1)

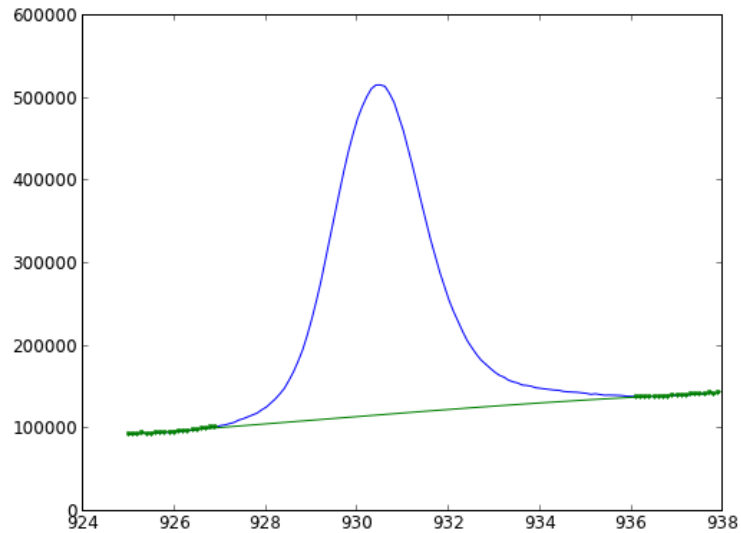
```

```

25
26 A1 = I1 - Ib
27 print 'The integrated peak area is: ',A1

```

The integrated peak area is: 1049187.8531

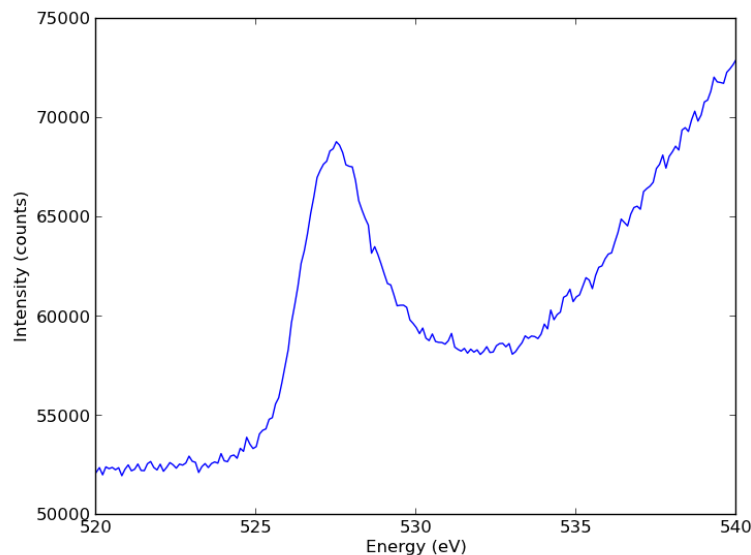


Next, we consider the region where the oxygen peak is.

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 E, C = np.loadtxt('data/Cu-0-30L-region2.txt', skiprows=5, unpack=True)
5 plt.plot(E,C)
6 plt.xlabel('Energy (eV)')
7 plt.ylabel('Intensity (counts)')
8 plt.savefig('images/XPS-Cu-0-30L-region2.png')

```



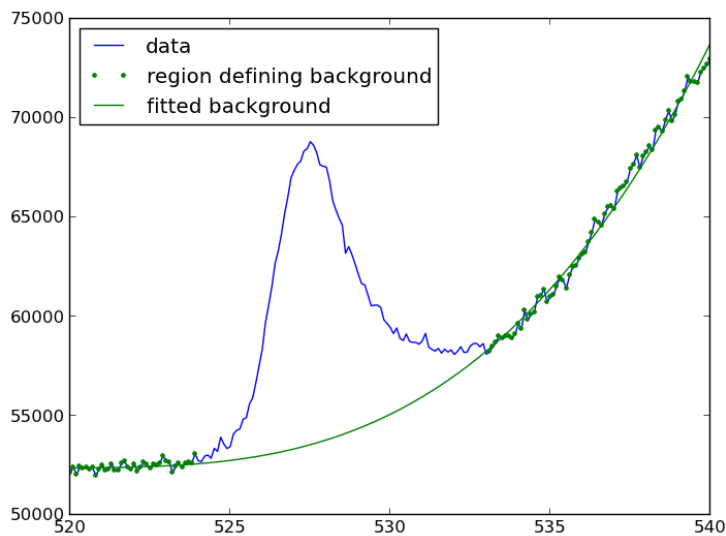
We need to integrate the area under this curve to quantify the oxygen. There is obviously a background we do not want to consider. We say the peak is between 524 and 533 eV, and we will fit a function to the region outside it, e.g. a third order polynomial. Then, we can simply integrate the data, and the background, and take the difference. The choice of polynomial order affects the area by up to 10%. This is a source of uncertainty in the analysis, as there is no definitive background function to use. You should be explicit in what you do so others can reproduce it.

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 E, C = np.loadtxt('data/Cu-0-30L-region2.txt', skiprows=5, unpack=True)
5
6 ind = (E < 524) | (E > 533) # / is a logical "or"
7
8 Eb = E[ind]
9 Cb = C[ind]
10
11 pb = np.polyfit(Eb, Cb, 3)
12 Cbfit = np.polyval(pb, E)
13
14 plt.plot(E, C, label='data')
15 plt.plot(E[ind], C[ind], 'g. ', label='region defining background')
16 plt.plot(E, Cbfit, label='fitted background')
17 plt.legend(loc='best')
18 plt.savefig('images/XPS-Cu-0-peak-area.png')
19
20 I1 = -np.trapz(C, E)      # Note energies are decreasing in datafile so we take the negative
21 Ib = -np.trapz(Cbfit, E)
22
23 print 'The area under the curve is {0}'.format(I1 - Ib)

```

The area under the curve is 50387.380641



Finally to get atomic percent, we normalize each area by its respective sensitivity factor.

$$x_A = \frac{Area_A / S_A}{\sum Area_i / S_i}$$

```

1 A_Cu = 1049187.85313
2 A_O = 50387.3806319
3
4 s_Cu = 4.2
5 s_O = 2.1
6 print (A_O / s_O) / (A_O / s_O + A_Cu / s_Cu)
7
8 # in vector form:
9 import numpy as np
10 A = np.array([1049187.85313, 50387.3806319])
11 s = np.array([4.2, 2.1])
12
13 x = (A / s) / (A / s).sum()
14 X = (A / s) / np.sum(A / s)
15 print x
16 print X

```

```

0.0876330760691
[ 0.91236692  0.08763308]
[ 0.91236692  0.08763308]

```

Dedicated software solutions exist for analysing XPS spectra (e.g. <http://www.casaxps.com/>, Single Academic License: Euro 830.00!) Additional detailed information can be found at <http://www.casaxps.com/ebooks/ebooks.htm>. See also <http://www.xpsfitting.com/2011/09/background-selection.html>.

7.2 BET surface area

In BET analysis, an equation is derived that relates the volume of gas adsorbed to the relative pressure of gas. This equation is derived from the following assumptions:

1. Adsorption occurs in multilayers
2. The first monolayer has one heat of adsorption
3. Subsequent layers have a different and constant heat of adsorption
4. There is no capillary condensation

With those assumptions, the isotherm takes on this form:

$$\frac{P/P_0}{V(1 - P/P_0)} = \frac{1}{cV_m} + \frac{c-1}{cV_m}P/P_0$$

In this equation c is a constant that is material specific, and V_m is the volume of an adsorbed *monolayer* of gas. From this equation, you can compute the quantities c and V_m by fitting a line to a plot of $\frac{P/P_0}{V(1-P/P_0)}$ vs P/P_0 . The slope of the line will be equal to $(c-1)/(cV_m)$, and the intercept will be equal to $1/(cV_m)$. The volume of the adsorbed monolayer is: $V_m = 1/(\text{slope} + \text{intercept})$. We do not need to know what c is.

If we know the cross-sectional area of the adsorbed molecules, we compute the surface area as:

$A_s = A_m \frac{V_m}{V_{T,P}} N_A$ where A_m is the cross-sectional area of the molecule, $V_{T,P}$ is the standard molar volume of the gas at (T,P), and N_A is Avogadro's number.

Here is a worked example. Typically, one measures the volume of adsorbed gas as a function of pressure. When P/P_0 is less than about 0.35, the analysis is valid. Above that range, the data usually not linear.

P/P0	Vads (cc/g)
0.017	19
0.041	31
0.07	39
0.11	48
0.162	56
0.209	63
0.319	72
0.381	79
0.44	84
0.53	95

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 data = np.array(data)
5
```

```

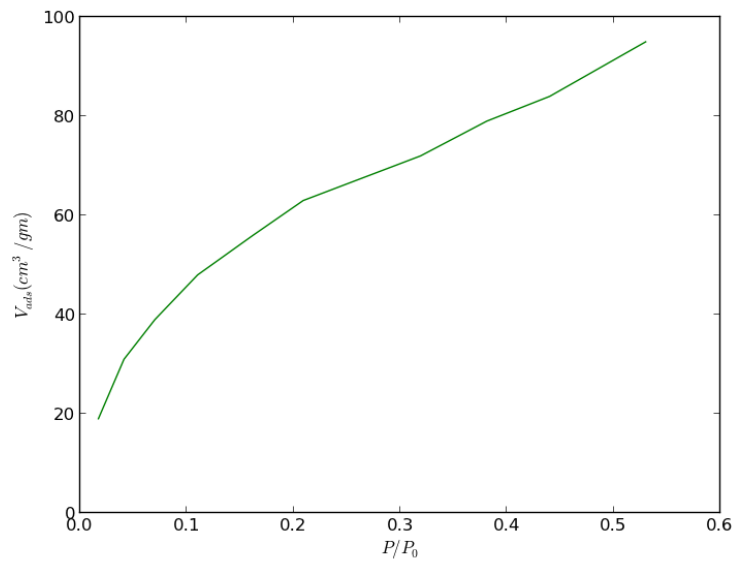
6 relP = np.array(data[:, 0])
7 Vads = np.array(data[:, 1])
8
9 plt.plot(relP, Vads)
10 plt.xlabel('$P/P_0$')
11 plt.ylabel('$V_{ads}$ (cm3/gm)')
12 plt.savefig('images/bet-0.png')

```

```

>>> >>> >>> >>> >>> >>> >>> >>> >>> [ <matplotlib.lines.Line2D object at 0x043DC1F0>]
<matplotlib.text.Text object at 0x0438BD50>
<matplotlib.text.Text object at 0x043C4070>

```



```

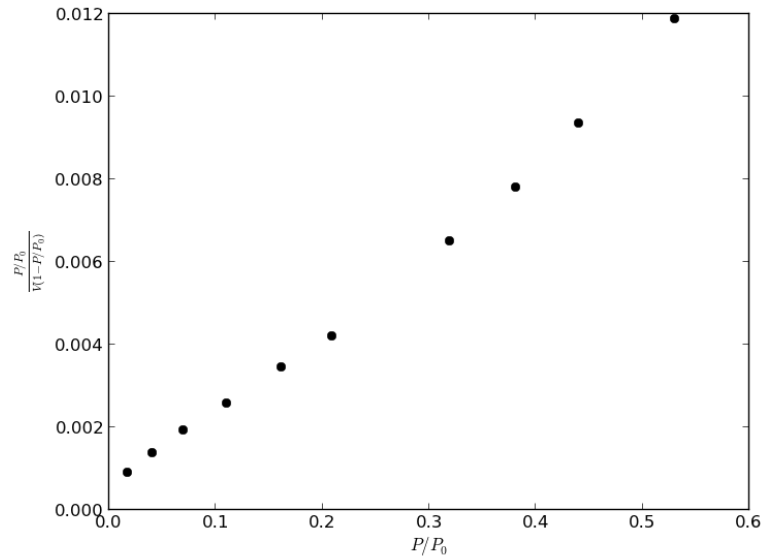
1 y = relP / (Vads * (1 - relP))
2
3 plt.figure()
4 plt.plot(relP, y, 'ko')
5 plt.xlabel('$P/P_0$')
6 plt.ylabel('$\frac{P/P_0}{V(1-P/P_0)}$')
7 plt.savefig('images/bet-1.png')

```

```

>>> <matplotlib.figure.Figure object at 0x043C4B90>
[ <matplotlib.lines.Line2D object at 0x044ED170>]
<matplotlib.text.Text object at 0x044C40B0>
<matplotlib.text.Text object at 0x044C7870>

```

Now, we compute confidence intervals on the parameters.

```

1  from scipy.stats.distributions import t
2
3  X = np.column_stack([relP**0, relP])
4  p, res, rank, s = np.linalg.lstsq(X, y)
5  intercept, slope = p
6
7  print 'intercept = ', p[0]
8  print 'slope = ', p[1]
9
10 plt.plot(relP, np.dot(X, p), 'r-')
11 plt.legend(['data', 'fit'], loc='best')
12 plt.savefig('images/bet-2.png')
13
14 # compute the confidence intervals
15 n = len(y) # number of data points
16 k = len(p) # number of parameters
17
18 sigma2 = np.sum((y - np.dot(X, p))**2) / (n - k) # RMSE
19
20 C = sigma2 * np.linalg.inv(np.dot(X.T, X)) # covariance matrix
21 se = np.sqrt(np.diag(C)) # standard error
22
23 alpha = 0.05 # 100*(1 - alpha) confidence level
24
25 sT = t.ppf(1.0 - alpha/2.0, n - k) # student T multiplier
26 CI = sT * se
27
28 print
29 for beta, ci in zip(p, CI):
30     print '{2: 1.4f} [{0: 1.4f} {1: 1.4f}]'.format(beta - ci, beta + ci, beta)

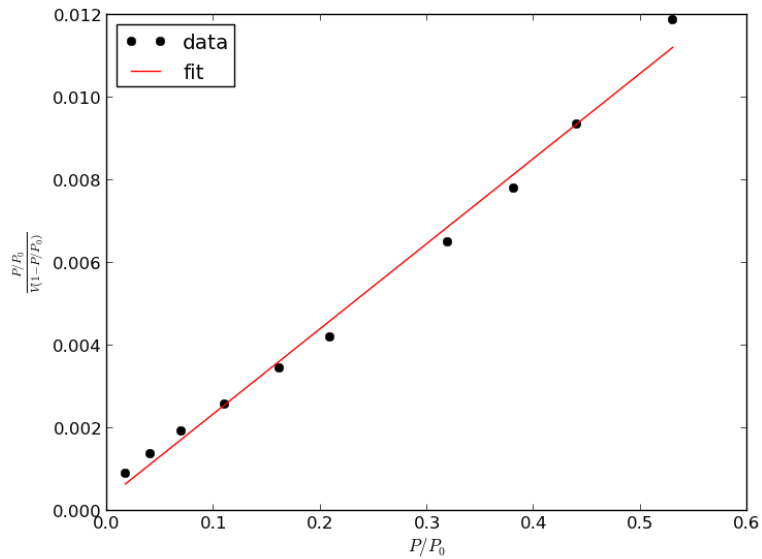
```

```

>>> >>> >>> intercept = 0.000303436320825
>>> slope = 0.0205910496331

```

```
>>> [<matplotlib.lines.Line2D object at 0x043D0370>]
<matplotlib.legend.Legend object at 0x04550A30>
>>> >>> ... >>> >>> >>> >>> >>> >>> >>> >>> >>> >>> >>>
... .. 0.0003 [-0.0001 0.0007]
0.0206 [ 0.0190 0.0221]
```



Interestingly, the intercept contains zero. The intercept is also very small, so we do not worry about it here. This may be due to fitting over too broad a range of P/P_0 . There is clear evidence of non-linearity past $P/P_0 = 0.1$.

Finally, we compute the surface area, and do a rough calculation on the uncertainty in the surface area based on the confidence interval of the slope.

```
1 for slope in [0.0190, 0.0206, 0.0221]:
2     Vmonolayer = 1.0 / (slope + intercept)
3     m = 1
4     Angstrom = 1e-10*m
5     Na = 6.022e23 # Avogadro's number
6     Am = 20.2 * Angstrom**2 # This is for Krypton
7     Vmolar = 22400 # cm^3/mol at STP
8     SurfaceArea = Am * Vmonolayer / Vmolar * Na
9     print 'Surface area = {0:1.2f} m^2/gm'.format(SurfaceArea)
```

```
... .. Surface area = 281.33 m^2/gm
Surface area = 259.79 m^2/gm
Surface area = 242.40 m^2/gm
```

That is a pretty broad range for the surface area estimate.

7.3 Summary notes

BET surface areas are very commonly reported.

- Molecule cross-sectional areas may be material dependent (<http://pubs.acs.org/doi/abs/10.1021/la00038a>)

8 References

References

Richard I. Masel. *Principles of adsorption and reaction on solid surfaces*. John Wiley & Sons, Inc., 1996.

Gabor A. Somorjai and Yimin Li. *Introduction to surface chemistry and catalysis*. John Wiley & Sons, Inc., second edition, 2010.

Robert L. Carter. *Molecular Symmetry and group theory*. John Wiley & Sons, Inc., 1998.

David M. Bishop. *Group theory and Chemistry*. Dover Publications, Inc., 1973.

Marc De Graef and Michael E. McHenry. *Structure of Materials - An introduction to Crystallography, diffraction and symmetry*. Cambridge University Press, 2007.

9 GNU Free Documentation License

GNU Free Documentation License
Version 1.3, 3 November 2008

Copyright (C) 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
<<http://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero

Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements",

"Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the

- terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
 - H. Include an unaltered copy of this License.
 - I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
 - J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
 - K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
 - L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
 - M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
 - N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
 - O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list

of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual

title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not

as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.