Geometry and Space Groups

richard.cooper@chem.ox.ac.uk

ECACOMSIG school 2013

What will we learn to do?

- Convert crystal data to orthogonal coordinates
- Compute geometry in crystallographic coordinate systems

- Change coordinate systems (abc)
- Change coordinates (xyz)
- Change reflection indices (hkl / MTZ)

Libraries (*e.g.* cctbx)

uc_sym_mat3 const &	metrical_matrix () const Access to metrical matrix.
uc_sym_mat3 const &	reciprocal_metrical_matrix () const Access to reciprocal metrical matrix
uc_mat3 const &	fractionalization_matrix () const Matrix for the conversion of cartesian to fractional coordinates.
uc_mat3 const &	orthogonalization_matrix () const Matrix for the conversion of fractional to cartesian coordinates.
FloatType	distance (fractional< FloatType > const &site_frac_1, fractional< FloatType > const &site_frac_2) const
uc_mat3	matrix_cart (sgtbx::rot_mx const &rot_mx) const
unit_cell	change_basis (sgtbx::rot_mx const &c_inv_r) const
unit_cell	change_basis (sgtbx::change_of_basis_op const &cb_op) const Transformation (change-of-basis) of unit cell parameters.

Bibliography / Further reading

- International Tables Volume A: Space-group symmetry (IUCr)

- IUCr Teaching Pamphlet #22: Matrices, mappings, and crytsallographic symmetry, Hans Wondratschek (IUCr, 1997); iucr.org/education/pamphlets/22

- Fundamentals of Crystallography 3rd ed., edited by C. Giacovazzo (IUCr, 2006)

- Computing Methods in Crystallography, John Rollett (Pergamon Press, 1965)

Notation

Very generally:

x	italic lower case	a scalar
$\mathbf{a}$	bold lower case	a vector
$\mathbf{M}$	bold upper case	a matrix

Orthogonalization

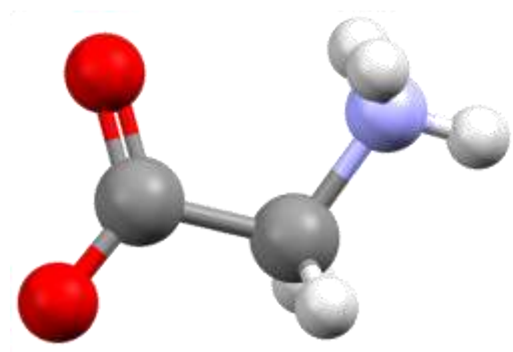
COORDINATE SYSTEMS

Why orthogonalize?

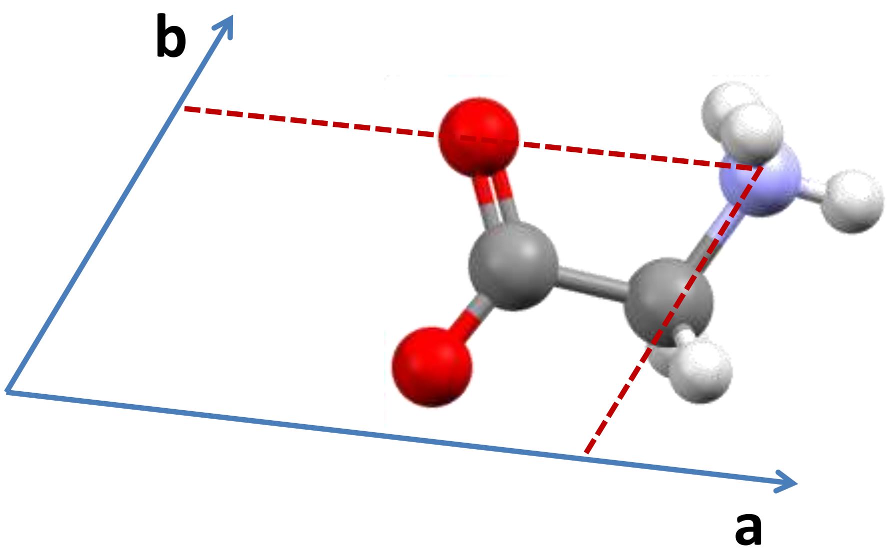
- **Compare two structures across a phase transition, or that have been published in different non-conventional settings.**
- Relate macroscopic orthogonally described properties (e.g. elasticity, piezoelectricity, etc.) to the crystallographic coordinate system.
- **Put part of a crystal structure into PDB format.**

Representing coordinates

- Crystallography typically uses fractional, non-Cartesian co-ordinates
  - Good for symmetry and diffraction formulae
  - Bad for geometrical calculation (distance, etc.)



Representing coordinates

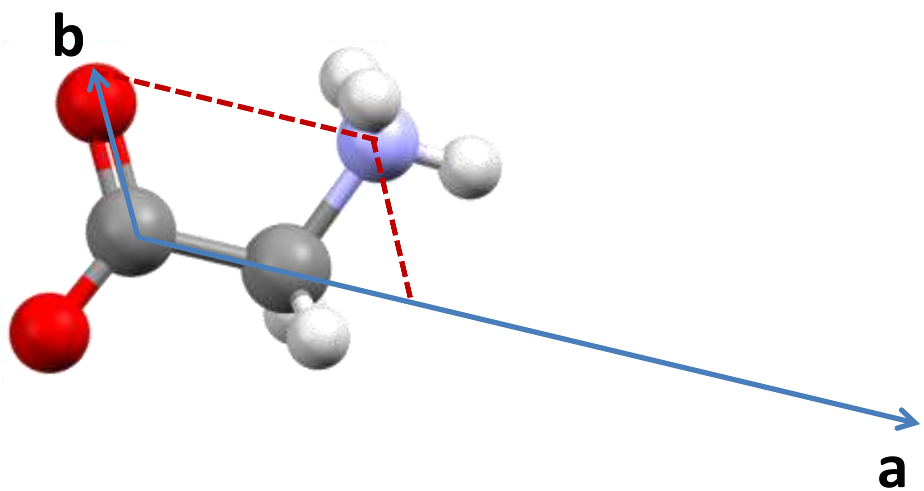


$x = 0.7\mathbf{a}$

$y = 0.8\mathbf{b}$

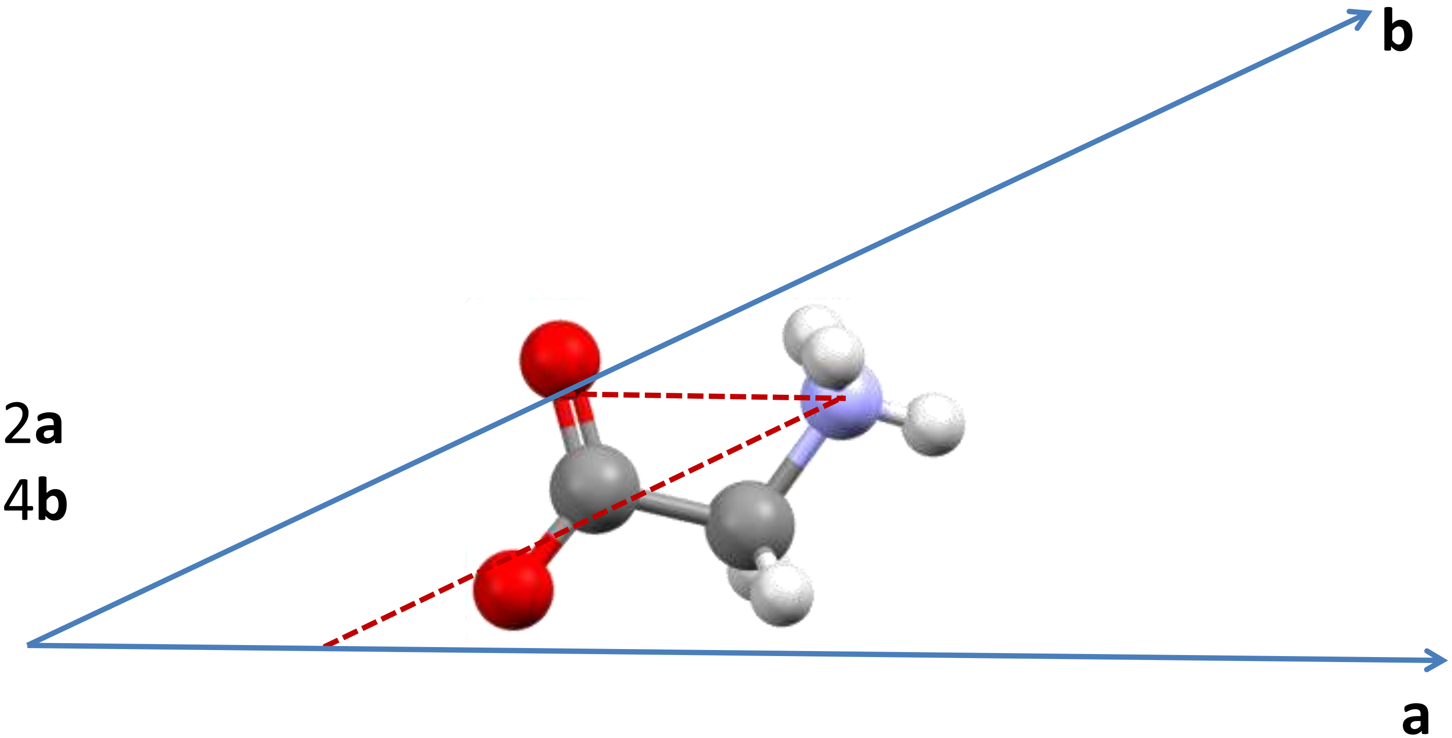
Representing coordinates

$x = 0.4\mathbf{a}$
$y = 1.0\mathbf{b}$



Representing coordinates

$x = 0.2\mathbf{a}$
$y = 0.4\mathbf{b}$

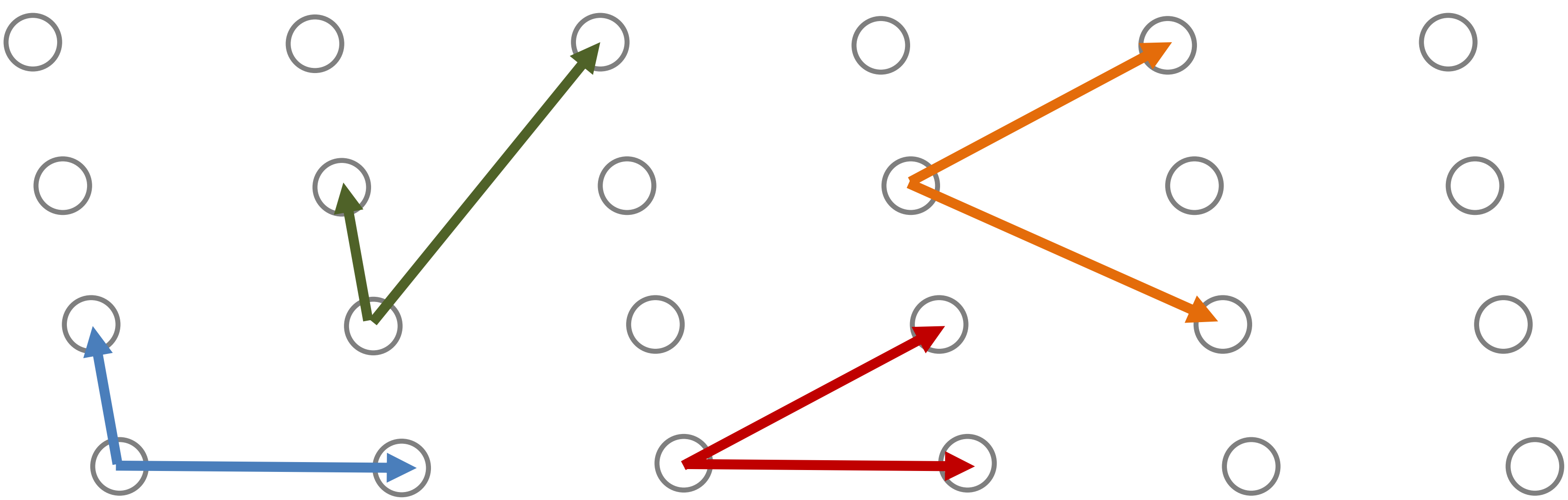


Basis vectors

- In the previous examples **a** and **b** are the basis vectors.
- They define coordinate axes for a particular reference frame.
- The molecule didn’t move. The way we refer to it changed.
- We can choose them to point wherever we like (provided they are linearly independent) – **but some ways are more sensible than others.**

Lattices

- A three-dimensional lattice can be described by 3 basis vectors. If basis vectors begin and end on lattice points the coordinates remain the same from one cell to the next.



Describing unit cell edges as vectors

General (triclinic) case. Construct a Cartesian coordinate system $\boldsymbol{e_1}$, $\boldsymbol{e_2}$, $\boldsymbol{e_3}$ for **a**, **b**, and **c** :

- $\boldsymbol{e_1}$ is (1,0,0); $\boldsymbol{e_2}$ is (0,1,0); $\boldsymbol{e_3}$ is (0,0,1)
- Express (**a**, **b**, **c**) as linear combinations of ($\boldsymbol{e_1}$, $\boldsymbol{e_2}$, $\boldsymbol{e_3}$):

$$\mathbf{a} = k_{11}\, \boldsymbol{e_1}. + k_{12}.\, \boldsymbol{e_2} + k_{13}.\, \boldsymbol{e_3}$$

$$\mathbf{b} = k_{21}.\, \boldsymbol{e_1} + k_{22}.\, \boldsymbol{e_2} + k_{23}.\, \boldsymbol{e_3}$$

$$\mathbf{c} = k_{31}.\boldsymbol{e_1} + k_{32}.\, \boldsymbol{e_2} + k_{33}.\, \boldsymbol{e_3}$$

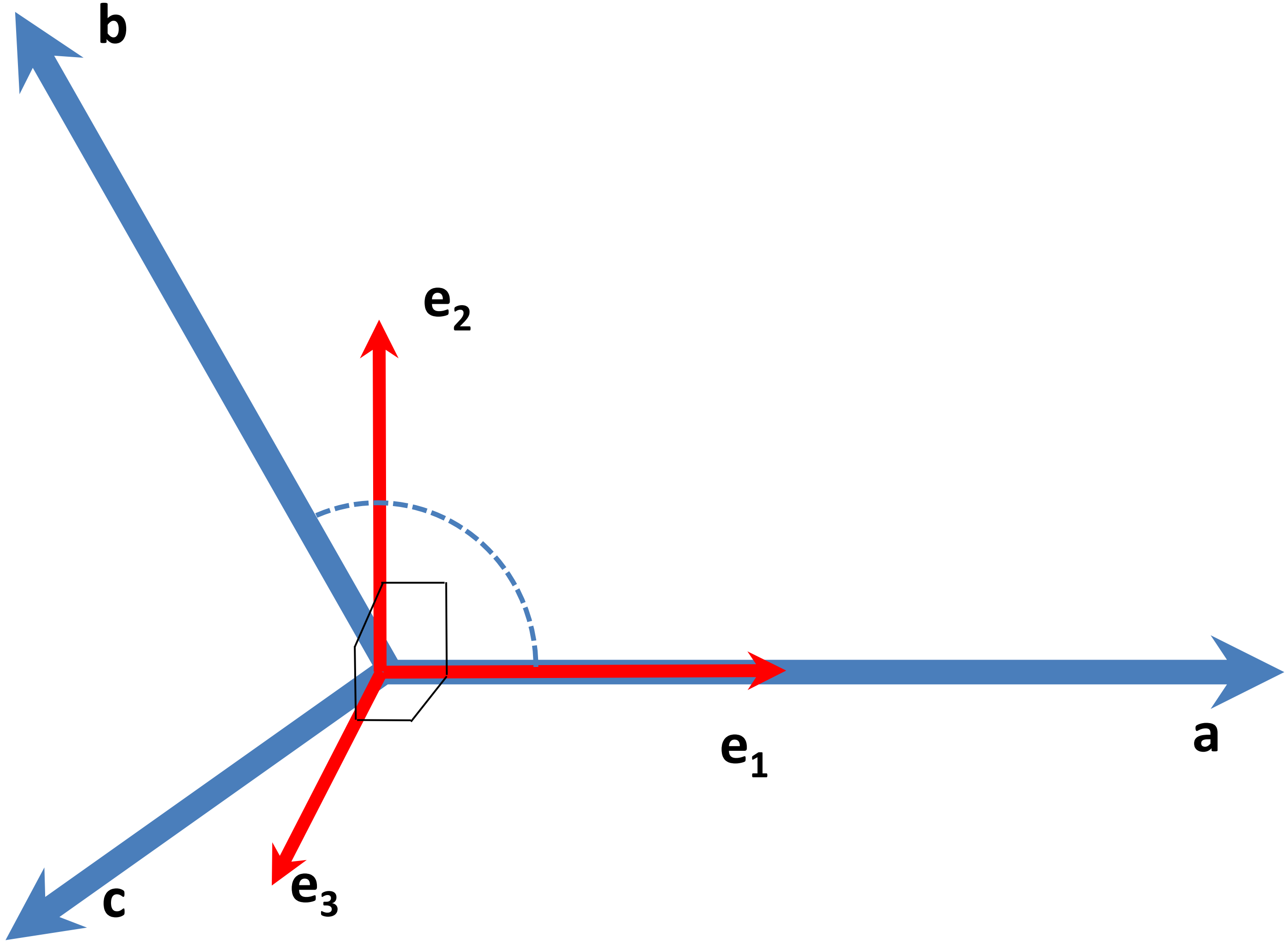
$$(\mathbf{a,b,c})^{\mathbf{T}} = \mathbf{M}^{-1}\, (\mathbf{e_1,e_2,e_3})^{\mathbf{T}}$$

Express new axes in old system

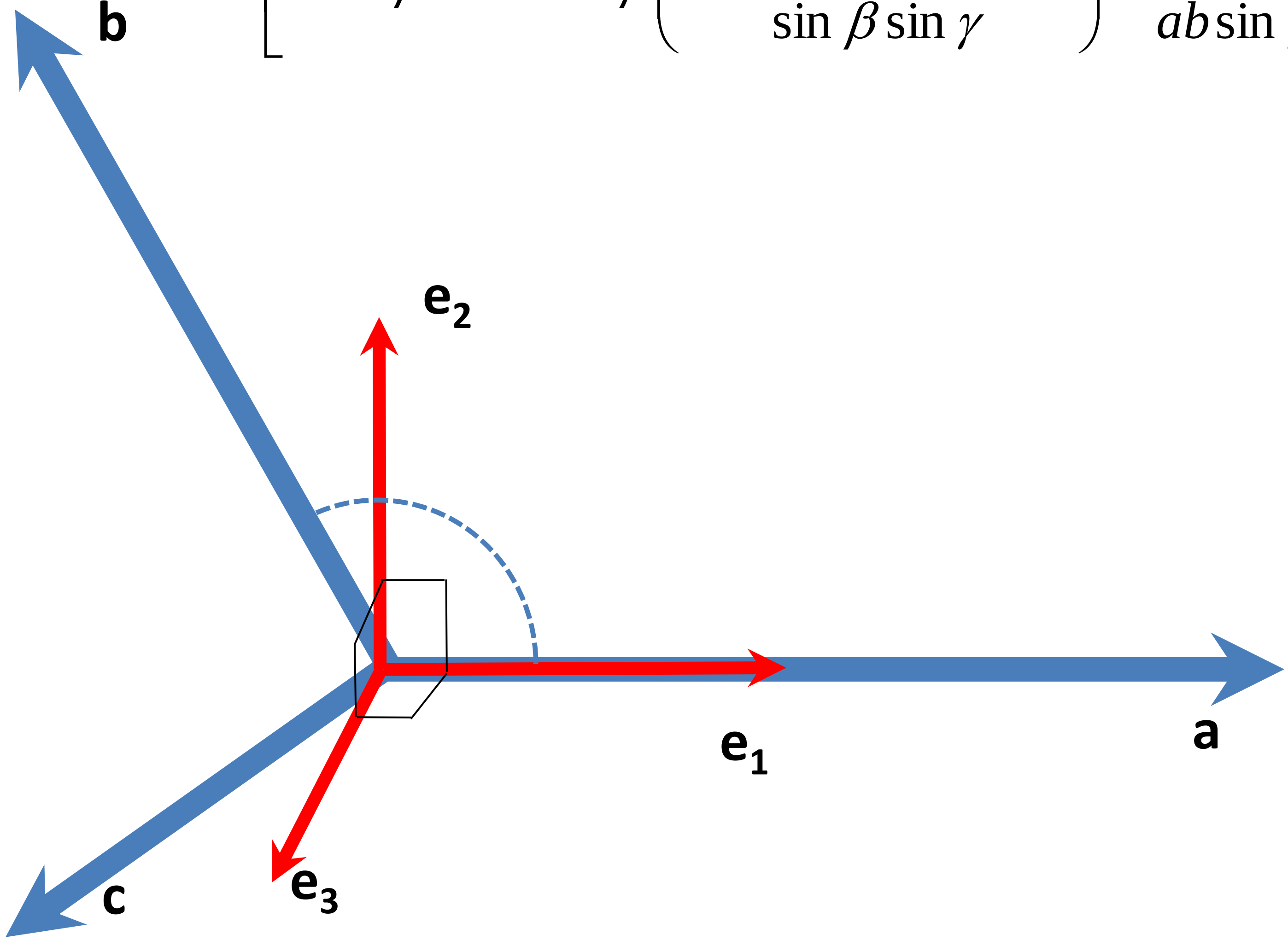
$\mathbf{e_1}$ is parallel to crystallographic **a** axis .

$\mathbf{e_2}$ is in **ab** plane,a perpendicular to $\mathbf{e_1}$: $\mathbf{a} \times \mathbf{b} \times \mathbf{e_1}$.

$\mathbf{e_3}$ perpendicular to **ab** plane: $\mathbf{a} \times \mathbf{b}$

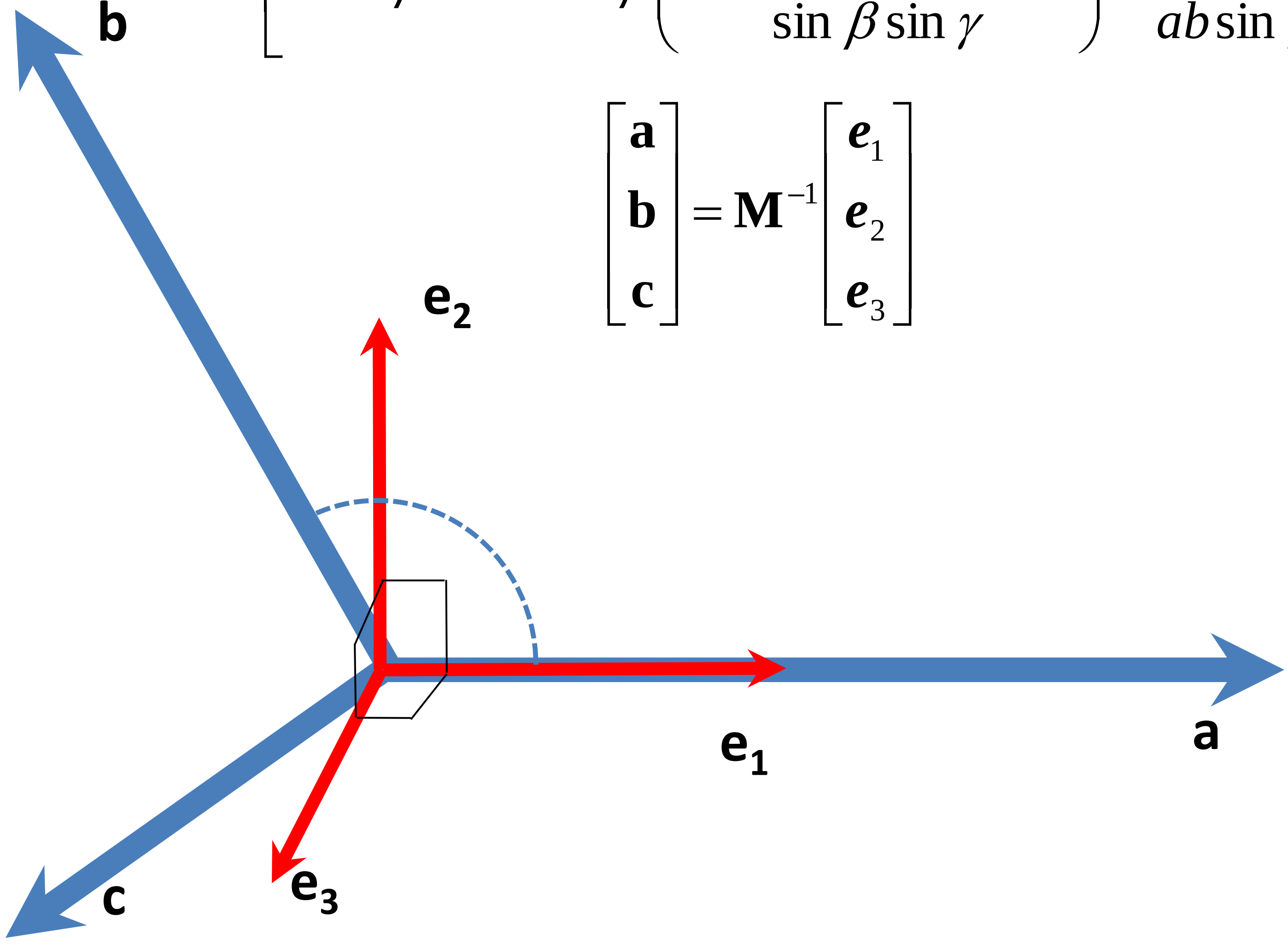


$$\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix} = \begin{bmatrix} a & 0 & 0 \\ b\cos\gamma & b\sin\gamma & 0 \\ c\cos\beta & -c\sin\beta\left(\dfrac{\cos\beta\cos\gamma - \cos\alpha}{\sin\beta\sin\gamma}\right) & \dfrac{V}{ab\sin\gamma} \end{bmatrix} \begin{bmatrix} \boldsymbol{e}_1 \\ \boldsymbol{e}_2 \\ \boldsymbol{e}_3 \end{bmatrix}$$



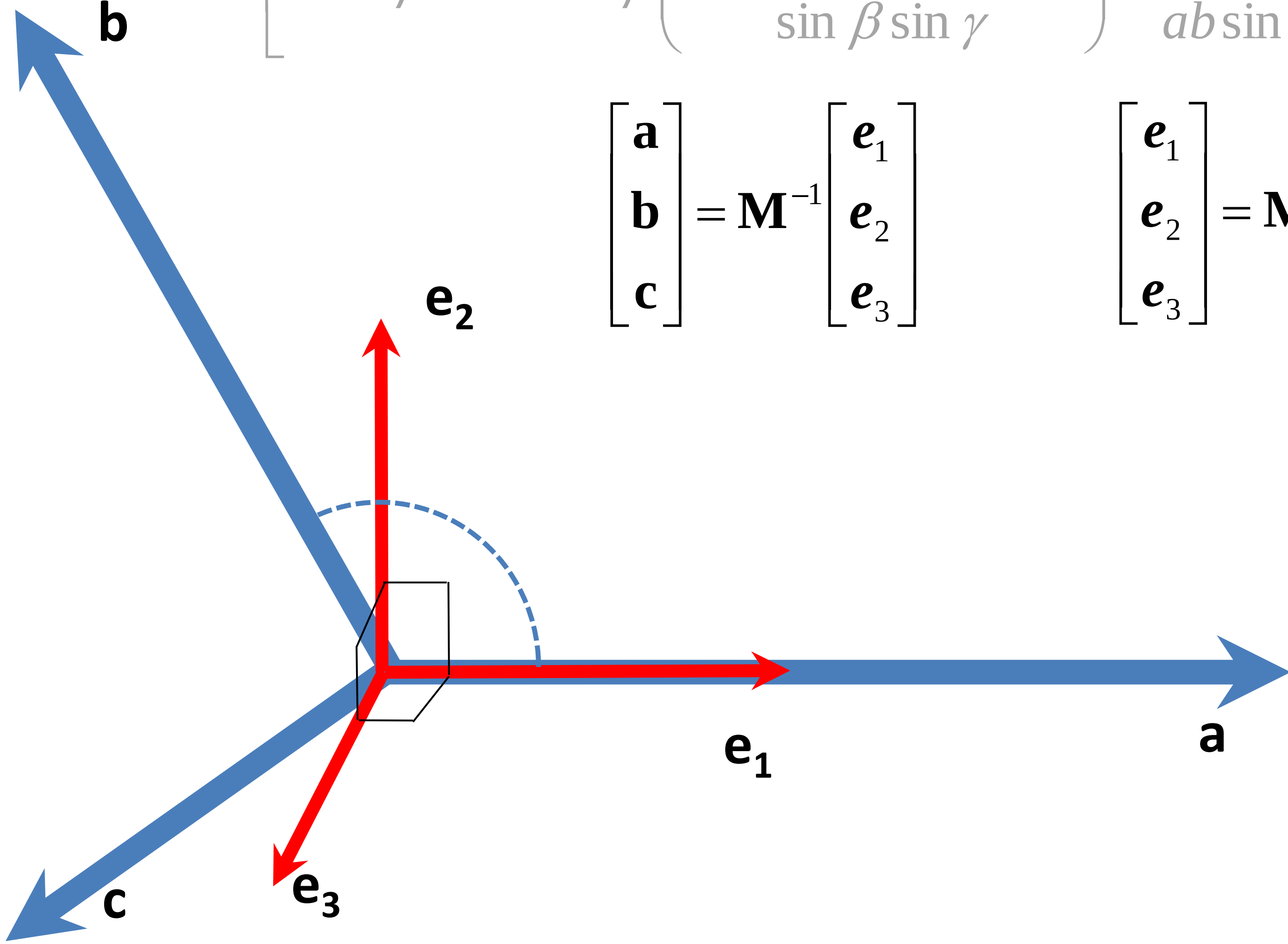
$$
\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix} = \begin{bmatrix} a & 0 & 0 \\ b\cos\gamma & b\sin\gamma & 0 \\ c\cos\beta & -c\sin\beta\left(\dfrac{\cos\beta\cos\gamma-\cos\alpha}{\sin\beta\sin\gamma}\right) & \dfrac{V}{ab\sin\gamma} \end{bmatrix} \begin{bmatrix} \boldsymbol{e}_1 \\ \boldsymbol{e}_2 \\ \boldsymbol{e}_3 \end{bmatrix}
$$

$$
\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix} = \mathbf{M}^{-1} \begin{bmatrix} \boldsymbol{e}_1 \\ \boldsymbol{e}_2 \\ \boldsymbol{e}_3 \end{bmatrix}
$$



$$\begin{bmatrix}\mathbf{a}\\\mathbf{b}\\\mathbf{c}\end{bmatrix}=\begin{bmatrix} a & 0 & 0 \\ b\cos\gamma & b\sin\gamma & 0 \\ c\cos\beta & -c\sin\beta\left(\dfrac{\cos\beta\cos\gamma-\cos\alpha}{\sin\beta\sin\gamma}\right) & \dfrac{V}{ab\sin\gamma}\end{bmatrix}\begin{bmatrix}\boldsymbol{e}_1\\\boldsymbol{e}_2\\\boldsymbol{e}_3\end{bmatrix}$$

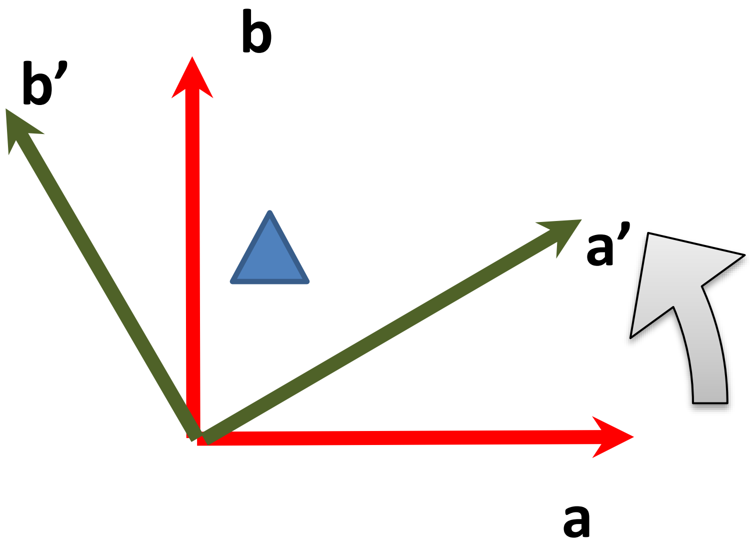
$$\begin{bmatrix}\mathbf{a}\\\mathbf{b}\\\mathbf{c}\end{bmatrix}=\mathbf{M}^{-1}\begin{bmatrix}\boldsymbol{e}_1\\\boldsymbol{e}_2\\\boldsymbol{e}_3\end{bmatrix} \qquad \begin{bmatrix}\boldsymbol{e}_1\\\boldsymbol{e}_2\\\boldsymbol{e}_3\end{bmatrix}=\mathbf{M}\begin{bmatrix}\mathbf{a}\\\mathbf{b}\\\mathbf{c}\end{bmatrix}$$



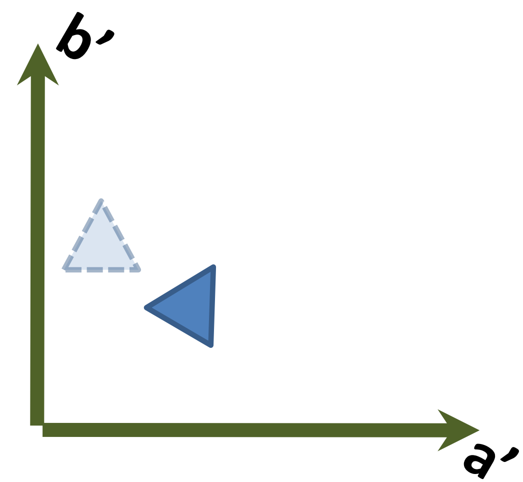
Transforming coordinates

b

- Coordinates transform differently to the cell axes:



Axes rotate 30° anti-clockwise from ab to a'b'.



Object stands still, but relative to axes it appears to have moved clockwise.

Transforming coordinates

- Basis transforms covariantly; coordinates transform contravariantly

$$\begin{bmatrix} x_{cart} \\ y_{cart} \\ z_{cart} \end{bmatrix} = \left(\mathbf{M}^{-1}\right)^{T} \begin{bmatrix} x_{frac} \\ y_{frac} \\ z_{frac} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{e}_1 \\ \boldsymbol{e}_2 \\ \boldsymbol{e}_3 \end{bmatrix} = \mathbf{M} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix}$$

$$\begin{bmatrix} x_{frac} \\ y_{frac} \\ z_{frac} \end{bmatrix} = \mathbf{M}^{T} \begin{bmatrix} x_{cart} \\ y_{cart} \\ z_{cart} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix} = \mathbf{M}^{-1} \begin{bmatrix} \boldsymbol{e}_1 \\ \boldsymbol{e}_2 \\ \boldsymbol{e}_3 \end{bmatrix}$$

Transforming X-ray data

Find matrix $\mathbf{U}$ to convert *hkl* vector to reciprocal Cartesian basis... $\mathbf{Uh} = \mathbf{h'}$

1. We know that $\mathbf{h}^T\mathbf{x} = \phi$. Therefore $\mathbf{h'}^T\mathbf{x'} = \phi$.

2. If a matrix $\mathbf{L}$ transforms $\mathbf{x}$ to $\mathbf{x'}$, then:

$$\mathbf{h'}^T\mathbf{x'} = (\mathbf{Uh})^T \mathbf{L}\,\mathbf{x} = \mathbf{h}^T\mathbf{U}^T \mathbf{L}\,\mathbf{x} = \phi$$

3. Therefore $\mathbf{U}^T\mathbf{L} = \mathbf{I}$ and $\mathbf{U} = (\mathbf{L}^T)^{-1}$

Transforming coordinates

Reciprocal space coordinates transform in the same way as the cell vectors

$$\begin{bmatrix} x_{cart} \\ y_{cart} \\ z_{cart} \end{bmatrix} = \left(\mathbf{M}^{-1}\right)^T \begin{bmatrix} x_{frac} \\ y_{frac} \\ z_{frac} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{e}_1 \\ \boldsymbol{e}_2 \\ \boldsymbol{e}_3 \end{bmatrix} = \mathbf{M} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix} \quad \begin{bmatrix} h_c \\ k_c \\ l_c \end{bmatrix} = \mathbf{M} \begin{bmatrix} h \\ k \\ l \end{bmatrix}$$

$$\begin{bmatrix} x_{frac} \\ y_{frac} \\ z_{frac} \end{bmatrix} = \mathbf{M}^T \begin{bmatrix} x_{cart} \\ y_{cart} \\ z_{cart} \end{bmatrix} \quad \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix} = \mathbf{M}^{-1} \begin{bmatrix} \boldsymbol{e}_1 \\ \boldsymbol{e}_2 \\ \boldsymbol{e}_3 \end{bmatrix} \quad \begin{bmatrix} h \\ k \\ l \end{bmatrix} = \mathbf{M}^{-1} \begin{bmatrix} h_c \\ k_c \\ l_c \end{bmatrix}$$

THE METRIC TENSOR

The Metric Tensor

$$\begin{bmatrix} \Delta x_{cart} \\ \Delta y_{cart} \\ \Delta z_{cart} \end{bmatrix} = \left(\mathbf{M}^{-1}\right)^{T} \begin{bmatrix} \Delta x_{frac} \\ \Delta y_{frac} \\ \Delta z_{frac} \end{bmatrix}$$

$$\begin{bmatrix} \Delta x_{cart} & \Delta y_{cart} & \Delta z_{cart} \end{bmatrix} \begin{bmatrix} \Delta x_{cart} \\ \Delta y_{cart} \\ \Delta z_{cart} \end{bmatrix} = \Delta x^{2}{}_{cart} + \Delta y^{2}{}_{cart} + \Delta z^{2}{}_{cart} = d^{2}$$

$$\begin{bmatrix} \Delta x_{frac} & \Delta y_{frac} & \Delta z_{frac} \end{bmatrix} \mathbf{M}^{-1} \left(\mathbf{M}^{-1}\right)^{T} \begin{bmatrix} \Delta x_{frac} \\ \Delta y_{frac} \\ \Delta z_{frac} \end{bmatrix} = \Delta \mathbf{x} \mathbf{G} \Delta \mathbf{x} = d^{2}$$

The Metric Tensor

- Easy to calculate.
- Useful for computing.
- Easy to transform:

  G' = MGM^T

$$G = \begin{bmatrix} \mathbf{a.a} & \mathbf{a.b} & \mathbf{a.c} \\ \mathbf{a.b} & \mathbf{b.b} & \mathbf{b.c} \\ \mathbf{a.c} & \mathbf{b.c} & \mathbf{c.c} \end{bmatrix}$$

$$G = \begin{bmatrix} aa & ab\cos\gamma & ac\cos\beta \\ ab\cos\gamma & bb & bc\cos\alpha \\ ac\cos\beta & bc\cos\alpha & cc \end{bmatrix}$$

The Metric Tensor

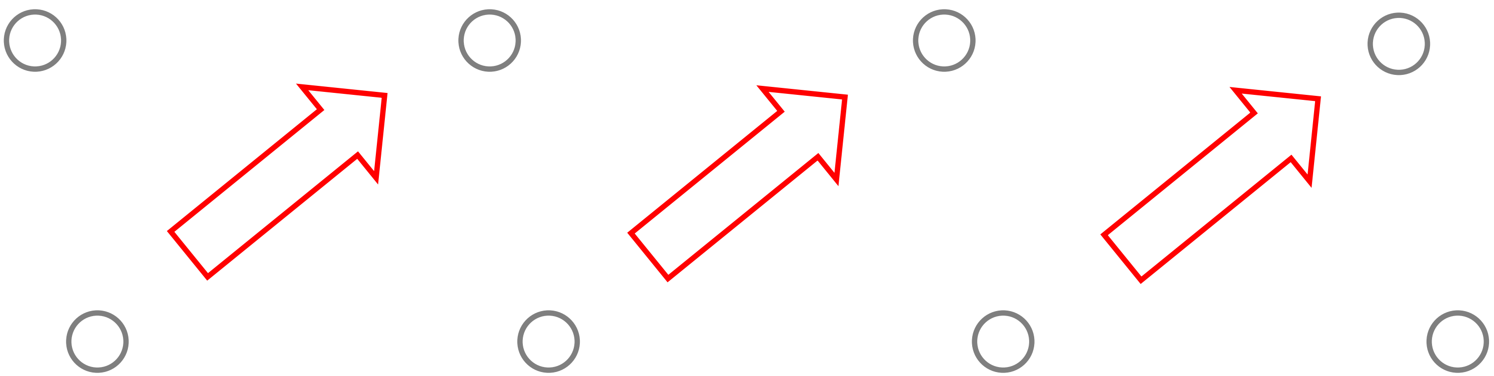
- **G** is very useful in programming – it is much less error prone that the explicit orthogonalisation matrix full of sines and cosines.
- The determinant of **G** is the volume of the unit cell squared.
- The inverse of **G** is denoted **G***.
- **G** transforms reciprocal (**a***,**b***,**c***) to real lattice directions (**a**,**b**,**c**), and **G*** does the reverse.
- $d^2 = \Delta x \; G \; \Delta x^T$

Implementation comment from Prof Neder: if comparing many distances (e.g. all atoms pairwise, then it may be quicker (fewer operations) to convert all coordinates to orthogonal basis in one pass, then compute distances between pairs.

TRANSFORMATIONS

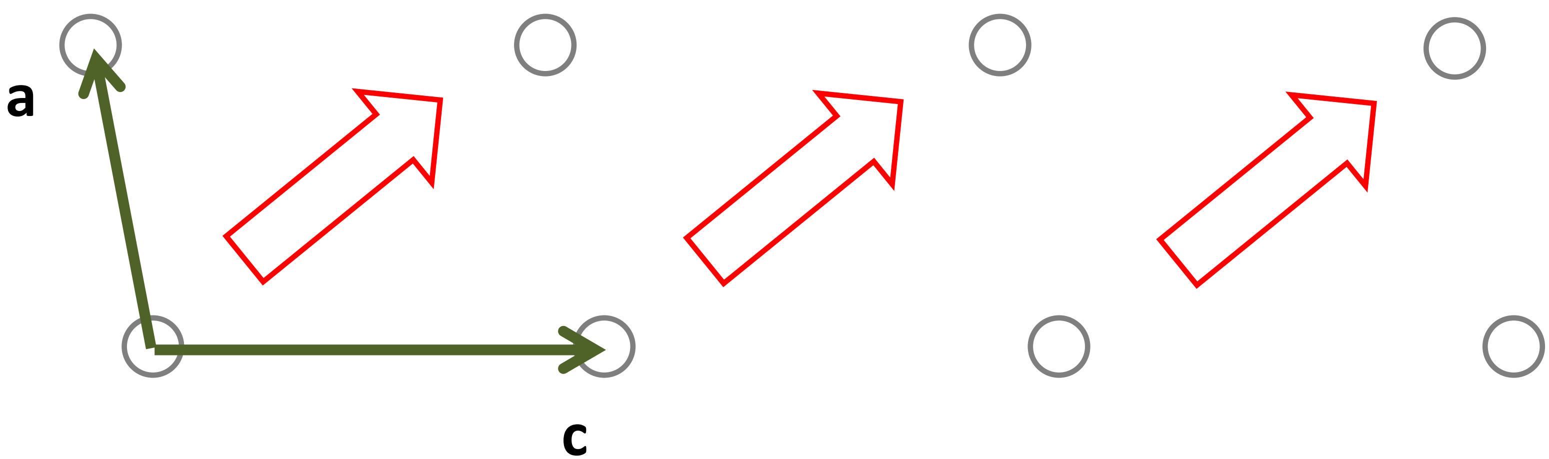
Transforming cells

- Classic example is changing space group $P2_1/n$ to $P2_1/c$
- The space groups are the same.



Transforming cells

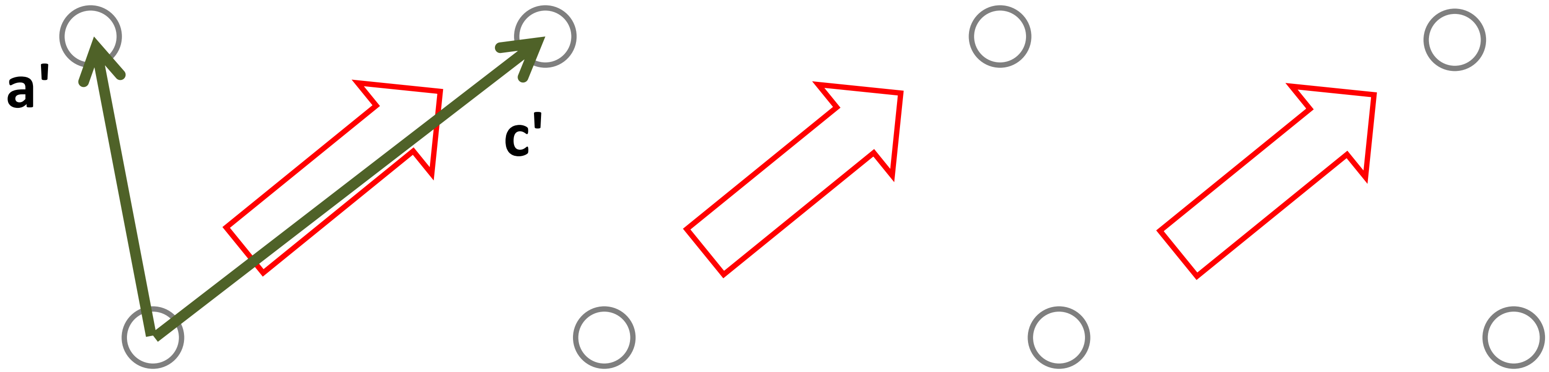
- Classic example is changing space group $P2_1/n$ to $P2_1/c$
- The space groups are the same.



$P2_1/n$

Transforming cells

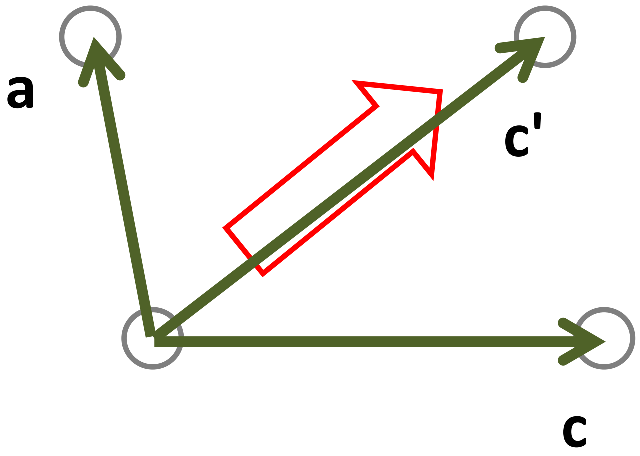
- Classic example is changing space group $P2_1/n$ to $P2_1/c$
- The space groups are the same.



$P2_1/c$

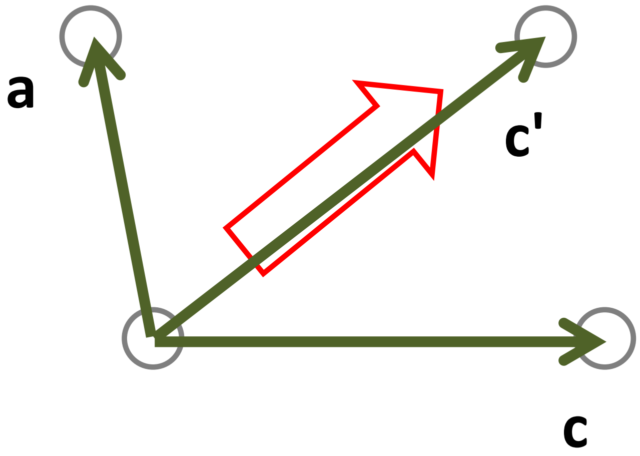
Transforming cells

- Transform basis vectors from $P2_1/n$ to $P2_1/c$



Transforming cells

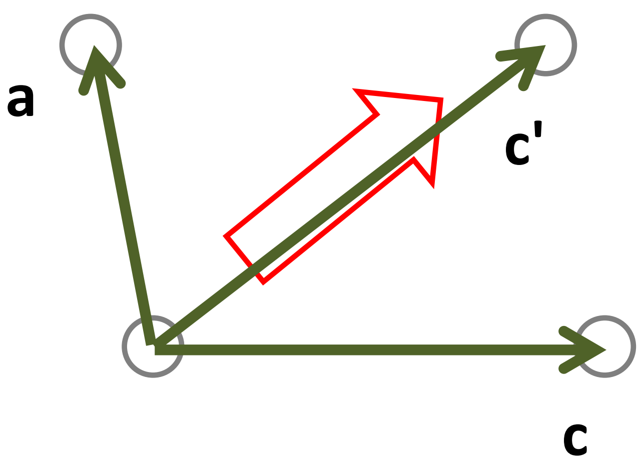
- Transform basis vectors from $P2_1/n$ to $P2_1/c$



$$\boxed{\begin{aligned} \mathbf{a}' &= \mathbf{a} \\ \mathbf{b}' &= \mathbf{b} \\ \mathbf{c}' &= \mathbf{a} + \mathbf{c} \end{aligned}}$$

Transforming cells

- Transform basis vectors from $P2_1/n$ to $P2_1/c$



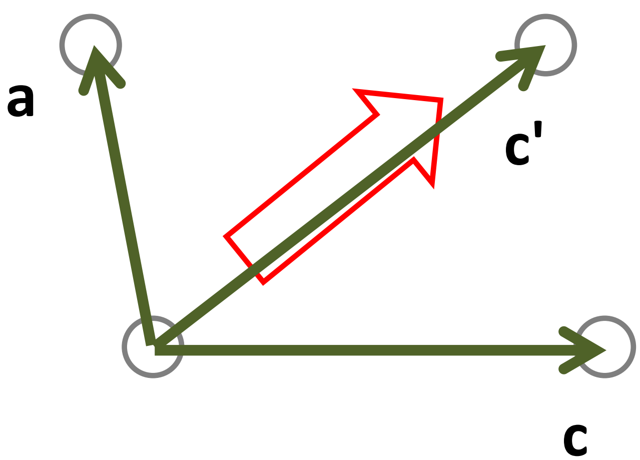
a’ = **a**
b’ = **b**
c‘ = **a** + **c**

a’ = 1.**a** + 0.**b** + 0.**c**
b’ = 0.**a** + 1.**b** **+ 0.c**
c‘ = 1.**a** + 0.**b** + 1.**c**

Transforming cells

- Transform basis vectors from $P2_1/n$ to $P2_1/c$

Transform basis vectors from $P2_1/n$ to $P2_1/c$



$$\mathbf{a}' = \mathbf{a}$$
$$\mathbf{b}' = \mathbf{b}$$
$$\mathbf{c}' = \mathbf{a} + \mathbf{c}$$

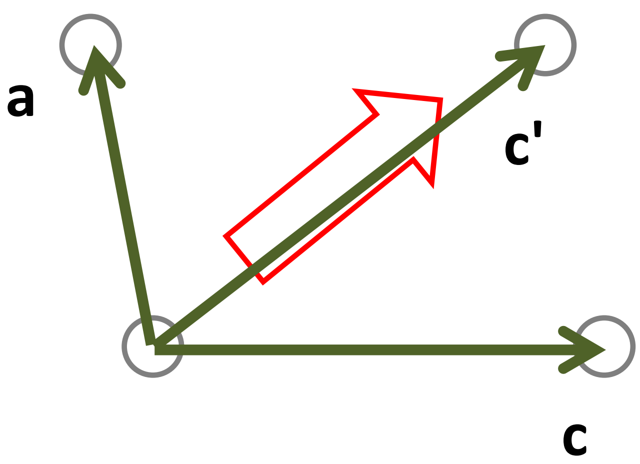
$$\mathbf{a}' = 1.\mathbf{a} + 0.\mathbf{b} + 0.\mathbf{c}$$
$$\mathbf{b}' = 0.\mathbf{a} + 1.\mathbf{b} + \mathbf{0.c}$$
$$\mathbf{c}' = 1.\mathbf{a} + 0.\mathbf{b} + 1.\mathbf{c}$$

$$\begin{bmatrix} \mathbf{a}' \\ \mathbf{b}' \\ \mathbf{c}' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix}$$

Transforming cells

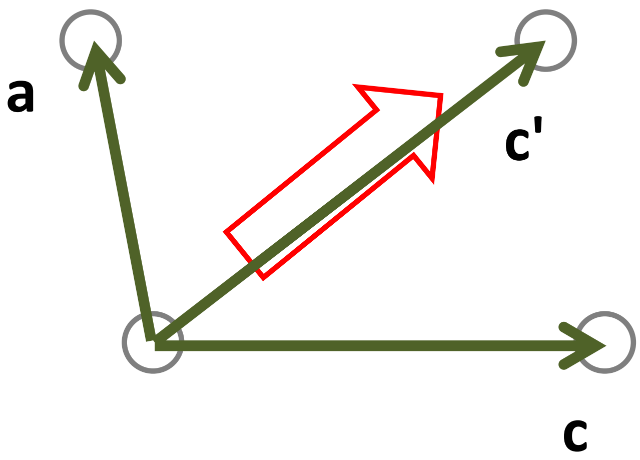
- Transform metric tensor from $P2_1/n$ to $P2_1/c$
- Avoids referring to cell vectors in Cartesian basis (**a**,**b**,**c**)

$$\mathbf{G'} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \mathbf{G} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}^T$$



Transforming cells

- Transform coordinates from $P2_1/n$ to $P2_1/c$



$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}^{-1} \right)^T \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Tutorials

Work through coordinate transform and geometry calculations in IPython notebook (or Python) using numpy matrices.

Thanks

- See bibliography slide above
- Dr David Watkin
- ECACOMSIG delegates and organisers