modelkit-crystal.docx

Bob Hanson, 2022.03.08

Notes regarding the use of the Jmol Model Kit with crystal structures.

Jmol 14.32.31 completes a set of functionality centered around the MODELKIT command. This functionality allows symmetry-aware editing of a crystal structure. The functionality is accessed with or without user interaction with the popup menu.

Some terminology:

**symmetry-equivalent atoms** These are atoms that are all generated from the same atom record of a CIF file by application of one (of possibly several) of the space group’s symmetry operations. Symmetry-equivalent atoms have the same *site* property.

All of the following methods select all of the oxygen atoms of quartz, as there is only one oxygen entry in the CIF file (@2).

LOAD =ams/quartz 1 PACKED

SELECT site=2 and {thismodel}

SELECT within(site, @2 and {thismodel})

x = @@2.site; SELECT site=x and {thismodel}

x = @@2.find(“equivalent”); SELECT x

Note the use of {thismodel} and @@ to limit the selection to the current model in cases where more than one frame is loaded.

**asymmetric unit** These are the atoms that are listed in a CIF file and are loaded by default by Jmol. Once-through application of a space group’s symmetry operations generates all the atoms of the unit cell. In Jmol, the asymmetric unit can be identified using:

SELECT !symmetry and {thismodel}

or

x = {!symmetry && thismodel}; SELECT x

**basis atom** I use this term, possibly incorrectly, to refer to the asymmetric unit atom that was used to generate a set of symmetry-equivalent atoms in the model. For the space group P1, the set of these atoms corresponds to the general crystallographic term “basis”. The basis atom is the first atom returned in a set of atoms with the same site for a given model, either using. It can be determined also by intersecting site with symmetry:

x = {site=2 & thismodel}[1]

x = @@245.find(“equivalent”)[1]

SELECT !symmetry and site=2 and thismodel

**symmetry-invariance** By this, I mean that the operation of symmetry operator on a model atom produces no change in position. An atom that is symmetry invariant with respect to a given mirror plane lies in the plane of that mirror plane; for a rotation, it would lie on the axis of a rotational vector. We can identify the operators for which an atom is symmetry invariant using @@1.symop(“invarianat”):

LOAD =ams/quartz 1 PACKED

PRINT @@1.symop("invariant").format("JSON")

*[ 6 ]*

SHOW symop 6

*C2 axis*

The importance of symmetry-invariant atoms is that they can be moved around without changing the space group, provided the position they move to has the same symmetry invariance. The function @@1.symop(“invariant”) returns a list of integers identifying the various symmetry operations that are invariant for Atom 1.

**fixed atoms** This is the set of atoms that cannot be moved without breaking the space group symmetry. This due to their being on a center of inversion or at the intersection of two incompatible space group operations -- for example, three planes, or a rotation axis perpendicular to a plane, or two non-colinear axes.

To find all atoms that are fixed by symmetry, just use the FIXED atom set :

A picture containing shape

Description automatically generatedload =ams/Almarudite

select fixed

*6 atoms selected*

select on @1

print @1.symop("invariant").format(“JSON”)

*[ 3,5,7,9,11,13,15,17,19,21,23 ]*

draw id "s3" symop 3

draw id "s7" symop 7

In this case, Atom 1 lies at the intersection of two different C2 rotations. Any movement of this atom will break one of these symmetries.

The principal commands include:

|  |  |
| --- | --- |
| MODELKIT HIDDEN | start org.jmol.modelkit.ModelKit, but do not display its menu. This is not actually necessary, as any MODELKIT command or the use of the FIXED atom set will initiate this directly. |
| MODELKIT ADD … [“PACKED”] | add or modify an atom and applying symmetry to add all symmetry-equivalent atoms. Optional “packed” keyword completes all faces, edges, and vertices of the unit cell in all cases  MODELKIT ADD {\*} will populate all (unpacked) sites.  MODELKIT ADD “P” {0.1 0.2 0.3} will add a set of symmetry-equivalent atoms that include this position.  MODELKIT ADD @2 “P” {0.1 0.2 0.3} will add a set of symmetry-equivalent atoms that include this position and bond them to the given atom and its similarly symmetry-equivalent atoms, if possible (there is a 1:1 correspondence, and reasonable bond distances).  MODELKIT ADD @2 “P” will change all atoms that are symmetry-equivalent to @2 to phosphorous. |
| MODELKIT DELETE … | delete a set of symmetry-equivalent atoms.  MODELKIT DELETE @1 |
| MODELKIT MOVETO … | move an atom and its associated symmetry-equivalent atoms to a new location  MODELKIT MOVETO @1 {1/2 1/3 1/3}  The move will only be successful if symmetry allows this change in position – that is, if both the original position and the new position have the same set of invariant symmetry operators. |
| MODELKIT ASSIGN SPACEGROUP | Find the space group and set it. You can use **calculate spacegroup** to see what this will be without actually setting it. |
| MODELKIT ASSIGN SPACEGROUP p1 | Set the space group to p1 |
| MODELKIT FIXED VECTOR point1 point2 | Constrains any atom dragging to a given vector; atom will still not move if symmetry does not allow it; replaces any fixed plane; cleared by changing the value of set MODELKITMODE or MODELKIT FIXED NONE |
| MODELKIT FIXED PLANE plane | constrains any atom dragging to a given plane; atom will still not move if symmetry does not allow it; replaces any fixed plane; cleared by changing the value of set MODELKITMODE or MODELKIT FIXED NONE |
| MODELKIT FIXED NONE | Removes FIXED VECTOR or FIXED PLANE constraint. |
|  |  |

Additional Jmol functions, using @@1 as an example of any atom “in this model”

|  |  |
| --- | --- |
| x = @@1.find("equivalent") | bitset of equivalent atoms |
| x = @@1.xyz.find("equivalent", options) | array of equivalent points within the unit cell; options are a single string of as many options as desired, including “packed” (includes all faces, edges, and vertices), “toFractional” (returns fractional coordinates), and “fromFractional” (assumes specified coordinates are . |
| x = @@1.symop("invariant") | array of integers, the symmetry operator indexes for which this atom is invariant; excludes the identity operator |
| x = {1/2 1/2 0}.symop("invariant") | array of integers, the symmetry operator indexes for which this position is invariant; excludes the identity operator |
| x = symop(2, "invariant") | an object that describes the invariant nature of this symmetry operator:  for a plane, returns {a, b, c, d} for that plane's equation ax + by + cz + d = 0;  for an axis, returns [point, vector], where point is on the line, and vector is the unit vector for the axis  for a center of inversion or an n-bar axis, returns the center point  for any operation involving a translation, returns "none"  for the identity operation, returns "identity" |
| x = @@1.symop(2, "invariant") | same as symop(2, "invariant"), except adjusts return to be through the given atom; presumes the atom is in a special position and not fixed; returns the "true" plane or axis or center of inversion, not just the one for the generating operator |
| x = {1/2 1/2 1/2}.symop(2, "invariant") | same as @@1.symop(2, "invariant"), but for a point rather than an atom |
|  |  |
|  |  |