CYXTAL

Generated by Doxygen 1.8.11

Contents

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Namespace Documentation

4.1 cyxtal.ext_aps.parsers Namespace Reference

Classes

- class VoxelStep

Functions

- def parse_xml (xmlfile, namespace={'step':'http://sector34.xor.aps.anl.gov/34ide:indexResult'}, disp=True)
- def get_reciprocal_base (lc, degrees=True)
- def get_base (lc, reciprocal=False, degrees=True)

Variables

- theta_1 = -np.pi

  MODULE LEVEL CONSTANTS RELATING TO COORDINATE TRANSFORMATION <NOTE> These are defined in terms of rotation matrices since it is more intuitive to see how each system is connected through simple rotation around x-axis (see cyxtal/documentation)
- **R_XHF2TSL**
- **R_TSL2XHF** = R_XHF2TSL.T
- float **theta_2** = -0.25
- **R_XHF2APS**
- **R_APS2XHF** = R_XHF2APS.T
- float **theta_3** = -0.75
- **R_APS2TSL**
- **R_TSL2APS** = R_APS2TSL.T
- **R_TSL2TSL** = np.eye(3)

4.1.1 Detailed Description

```
   _________  ___  ____________    __
  / _____/\ \/ / |/ /_  __/   |  / /
 / /       \  /|   / / / / /| | / /
/ /___     / //   | / / / ___ |/ /___
\_____/   /_//_/|_|/_/ /_/  |_/_____/

Copyright (c) 2016, C. Zhang.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

1) Redistributions of source code must retain the above copyright notice, this
list of conditions and the following disclaimer.
2) Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation and/or
other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

DESCRIPTION
-----------
VoxelStep: class
Container class to store voxel information and perform strain refinement.
parser_xml: function
Parsing xml output from APS (with/without strain refinement).
get_reciprocal_base:
Return reciprocal basis according to given lattice constants.
get_base: function
Return lattice basis according to given lattice constants.
NOTE
----
More information regarding the coordinate transformation can be found at:
http://www.aps.anl.gov/Sectors/33_34/microdiff/Instrument/coordinates-PE-system.pdf
```

4.1.2 Function Documentation

4.1.2.1 def cyxtal.ext_aps.parsers.get_base (*lc*, *reciprocal* = `False`, *degrees* = `True`)

```
DESCRIPTION
-----------
basis = get_base(lc)
    return the basis constructed based given lattice constant.
PARAMETERS
----------
lc: numpy.array/list/tuple [a,b,c,alpha,beta,gamma]
    Should contain necessary lattice constants that defines
    crystal structure.
reciprocal: boolean
    Whether the returned basis vectors in real reciprocal space
    or real space.
degree: boolean
    The angular lattice parameter are in degrees or radians.
RETURNS
-------
rst: numpy.array
    A 3x3 numpy array formed by the base vectors of given
    lattice constant. The base vectors are stack by column.
```

4.1.2.2 def cyxtal.ext_aps.parsers.get_reciprocal_base (*lc*, *degrees* = True)

```
DESCRIPTION
-----------
reciprocal_basis = get_reciprocal_base(lc)
    wrapper function to return the reciprocal basis rather
    than standard basis
PARAMETERS
----------
lc: numpy.array/list/tuple [a,b,c,alpha,beta,gamma]
    Should contain necessary lattice constants that defines
    crystal structure.
degree: boolean
    The angular lattice parameter are in degrees or radians.
RETURNS
-------
rst: numpy.array
    A 3x3 numpy array formed by the reciprocal base vectors of
    given lattice constant. The base vectors are stack by column.
```

4.1.2.3 def cyxtal.ext_aps.parsers.parse_xml (*xmlfile*, *namespace* = {'step':'http↩ ://sector34.xor.aps.anl.gov/34ide:indexResult'}, *disp* = True)

```
DESCRIPTION
-----------
[VoxelStep(),...]= parse_xml(DAXM_DATA.xml,
                             namespace={$XML_NAMESPACE_DICT},
                             disp=True)
    Parse the DAXM data from Beamline 34-I-DE to memory.
PARAMETERS
----------
xmlfile: str
    Path to the xml file requires data processing
namespace: dictionary
    Containing dictionary of the namespace used in the xml file.
    For data from beamline 34-ID-E, use the default setting should
    work.
    NOTE:
        If the beamline changes there namespace, it is necessary to
        extract those namespace and update them with this argument.
disp: boolean
    Toggle output of parsing progress (terminal only)
RETURNS
-------
voxels: list of VoxelStep
    List of instances of VoxelStep, each one representing indexed voxel
    in the xml data.
    NOTE:
        Not indexed file is screened out by checking the presence of a*
        for each voxel.
NOTE
----
```

4.1.3 Variable Documentation

4.1.3.1 cyxtal.ext_aps.parsers.R_APS2TSL

Initial value:

```
1 = np.array([[1.0,             0.0,             0.0],
2                     [0.0,  np.cos(theta_3), -np.sin(theta_3)],
3                     [0.0,  np.sin(theta_3),  np.cos(theta_3)]])
```

4.1.3.2 cyxtal.ext_aps.parsers.R_XHF2APS

Initial value:

```
1 = np.array([[1.0,              0.0,              0.0],
2                       [0.0,  np.cos(theta_2), -np.sin(theta_2)],
3                       [0.0,  np.sin(theta_2),  np.cos(theta_2)]])
```

4.1.3.3 cyxtal.ext_aps.parsers.R_XHF2TSL

Initial value:

```
1 = np.array([[1.0,              0.0,              0.0],
2                       [0.0,  np.cos(theta_1), -np.sin(theta_1)],
3                       [0.0,  np.sin(theta_1),  np.cos(theta_1)]])
```

4.1.3.4 cyxtal.ext_aps.parsers.theta_1 = -np.pi

MODULE LEVEL CONSTANTS RELATING TO COORDINATE TRANSFORMATION <NOTE> These are defined in terms of rotation matrices since it is more intuitive to see how each system is connected through simple rotation around x-axis (see cyxtal/documentation)

** XHF <-> TSL

Chapter 5

Class Documentation

5.1 cyxtal.cxtallite.Aggregate Class Reference

5.1.1 Detailed Description

```
DESCRIPTION
-----------
grainX = Aggregate(ListOfXtallites)
    A container class that holds several
```

The documentation for this class was generated from the following file:

- cxtallite.pyx

5.2 cyxtal.cxtallite.Eulers Class Reference

5.2.1 Detailed Description

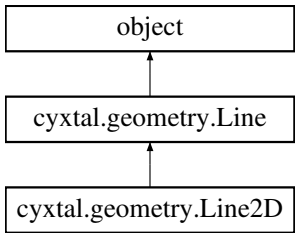
```
DESCRIPTION
-----------
Euler angle representation of orientation.
Calculation is carries out by converting to quaternions.

PARAMETERS
----------
phi1: double
    first of Euler angle
PHI:  double
    second of Euler angle
phi2: double
    third of Euler angle

METHODS
-------
```

The documentation for this class was generated from the following file:

- cxtallite.pyx

5.3 cyxtal.geometry.Line Class Reference

Inheritance diagram for cyxtal.geometry.Line:



Public Member Functions

- def **__init__** (self, pt_start, pt_end)
- def **start_pt** (self)
- def **start_pt** (self, new_start)
- def **end_pt** (self)
- def **end_pt** (self, new_end)
- def **length** (self)
- def **direction** (self)
- def **__str__** (self)
- def **__neg__** (self)
- def **__eq__** (self, other)
- def **__ne__** (self, other)
- def contain_point (self, point)
- def parallel_to (self, other)
- def skewed_from (self, other)
- def intercepted_by (self, other)
- def get_intercept (self, other)
- def dist2point (self, point)
- def dist2line (self, other)
- def angle2line (self, other, inDegree=True)

Public Attributes

- **start_pt**
- **end_pt**

5.3.1 Detailed Description

```
DESCRIPTION
-----------
Line(Point pt_0, Point pt_1)
    A line(segment) in 3D space defined with 2 Point instances.
PARAMETERS
----------
start_pt: Point
    Start point of the line instance.
end_pt: Point
    End point of the line instance.
length: float
    Return the length of the line segment.
```

```
direction: numpy.array
    Return the direction vector of the line segment.
    [start_pt->end_pt]
METHODS
-------
contain_point(Point pt)
    Test if self contains pt.
parallel_to(Line other)
    Test if self is parallel to other.
skewed_from(Line other)
    Test if self is skewed from other.
intercepted_by(Line other)
    Test if self is intercepted by other.
get_intercept(Line other)
    Return the intercept point.
dist2point(Point pt)
    Return the distance between self and pt (shortest).
dist2line(Line other)
    Return the distance between self and other (shortest).
angle2line(Line other, inDegree=True)
    Return the angle between self and other.
CLASSMETHOD
-----------
```

5.3.2 Member Function Documentation

5.3.2.1 def cyxtal.geometry.Line.angle2line (*self, other, inDegree =* `True`)

```
Return angle between self and other
```

5.3.2.2 def cyxtal.geometry.Line.contain_point (*self, point*)

```
Test if self contains point
```

5.3.2.3 def cyxtal.geometry.Line.dist2line (*self, other*)

```
Return the distance between two skewed or parallel lines
```

5.3.2.4 def cyxtal.geometry.Line.dist2point (*self, point*)

```
Return the distance to a given point
```

5.3.2.5 def cyxtal.geometry.Line.get_intercept (*self, other*)

```
Return the intercept point is exist, or return None
```

5.3.2.6 def cyxtal.geometry.Line.intercepted_by (*self, other*)

```
Quick test if one line is intercepted by another
```

5.3.2.7 def cyxtal.geometry.Line.parallel_to (*self, other*)

```
Test if two Line are parallel to each other
```

5.3.2.8 def cyxtal.geometry.Line.skewed_from (*self, other*)

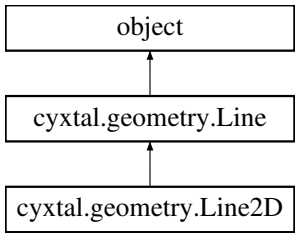
```
Quick test if one line is skewed from the other
```

The documentation for this class was generated from the following file:

- geometry.py

5.4 cyxtal.geometry.Line2D Class Reference

Inheritance diagram for cyxtal.geometry.Line2D:



Public Member Functions

- def __init__ (self, pt_start, pt_end)
- def **__str__** (self)
- def **direction** (self)
- def **parallel_to** (self, other)
- def get_intercept (self, other)
- def get_discrete_pts (self, step=5)
- def get_segments (self, step=5)

Static Public Member Functions

- def skewed_from (self, other)

Additional Inherited Members

5.4.1 Detailed Description

```
DESCRIPTION
-----------
Line2D(Point2D pt_start, Point2D pt_end)
    A 2D line (derived from the 3D Line class).
PARAMETERS
----------
METHODS
-------
get_discrete_pts(step=5)
    Return a numpy.array of coordinates discretize the 2D line.
get_segments(step=5)
    Return a numpy.array of segments.
CLASSMETHOD
-----------
```

5.4.2 Constructor & Destructor Documentation

5.4.2.1 def cyxtal.geometry.Line2D.__init__ (*self, pt_start, pt_end*)

```
Using two 2D point to define a 2D line
```

5.4.3 Member Function Documentation

5.4.3.1 def cyxtal.geometry.Line2D.get_discrete_pts (*self, step =* 5)

```
return a list of coordinates discretize the line
```

5.4.3.2 def cyxtal.geometry.Line2D.get_intercept (*self, other*)

```
Return the intercept of two lines
```

5.4.3.3 def cyxtal.geometry.Line2D.get_segments (*self, step =* 5)

```
return a list of segments
```

5.4.3.4 def cyxtal.geometry.Line2D.skewed_from (*self, other*) [static]

```
2D lines do not skew from each other
```

The documentation for this class was generated from the following file:

- geometry.py

5.5 cyxtal.cxtallite.OrientationMatrix Class Reference

5.5.1 Detailed Description

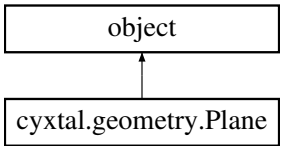
```
Matrix representation of orientation, this is defined as
the transpose of the rotation matrix.

PARAMETERS
----------
METHODS
```

The documentation for this class was generated from the following file:

- cxtallite.pyx

5.6 cyxtal.geometry.Plane Class Reference

Inheritance diagram for cyxtal.geometry.Plane:



Public Member Functions

- def **__init__** (self, point1, point2, point3)
- def normal (self)
- def **__str__** (self)
- def **__eq__** (self, other)
- def contain_point (self, point)
- def contain_line (self, line)
- def parallel_to (self, other)

5.6.1 Detailed Description

```
DESCRIPTION
-----------
Plane(Point pt_1, Point pt_2, Point pt_3)
    A plane in 3D space defined with 3 points.
PARAMETERS
----------
pt_0, pt_1, pt_2: Point
    Three non collinear points defines the flat plane (self).
normal: numpy.array
    Plane normal
METHODS
-------
contain_point(Point point)
    Test if self contains pt.
contain_line(Line l)
    Test if self contains l.
parallel_to(Plane other)
    Test if self and other are parallel to each other.
CLASSMETHOD
-----------
```

5.6.2 Member Function Documentation

5.6.2.1 def cyxtal.geometry.Plane.contain_line (*self, line*)

```
Quick test to see if a line lies in a plane
```

5.6.2.2 def cyxtal.geometry.Plane.contain_point (*self, point*)

```
Quick test to see if a point is in plane
```

5.6.2.3 def cyxtal.geometry.Plane.normal (*self*)

```
Plane normal
```

5.6.2.4 def cyxtal.geometry.Plane.parallel_to (*self, other*)

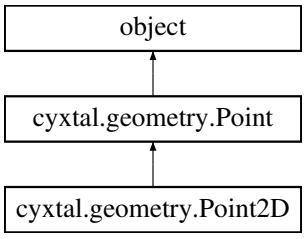
```
Quick test if two planes are parallel to each other
```

The documentation for this class was generated from the following file:

- geometry.py

5.7 cyxtal.geometry.Point Class Reference

Inheritance diagram for cyxtal.geometry.Point:



Public Member Functions

- def **__init__** (self, x, y, z)
- def **x** (self)
- def **x** (self, val)
- def **y** (self)
- def **y** (self, val)
- def **z** (self)
- def **z** (self, val)
- def **coord** (self)
- def **coord** (self, val)
- def __str__ (self)
- def **__eq__** (self, other)
- def **__ne__** (self, other)
- def **__len__** (self)
- def dist2point (self, other)
- def dist2line (self, line)
- def on_line (self, line)
- def in_plane (self, plane)

5.7.1 Detailed Description

```
DESCRIPTION
-----------
Point(x,y,z)
    Point in 3D space, base class provide bare bone abstraction
    for point related calculation.
PARAMETERS
----------
x,y,z: float
    Standard Cartesian coordinates for location description.
coord: array
    Vector of the Cartesian coordinates.
METHODS
-------
dist2point(Point other)
    Return the distance to another instance of Point.
dist2line(Line other)
    Return the distance to given instance of Line.
on_line(Line other)
    Whether the current instance lies on a given instance of Line.
in_plane(Plane other)
    Whether the current point lies in a given instance of Plane.
CLASSMETHOD
-----------
```

5.7.2 Member Function Documentation

5.7.2.1 def cyxtal.geometry.Point.__str__ (*self*)

```
String representation of Point
```

5.7.2.2 def cyxtal.geometry.Point.dist2line (*self, line*)

```
Return the distance to another line
```

5.7.2.3 def cyxtal.geometry.Point.dist2point (*self, other*)

```
Return the distance to another point
```

5.7.2.4 def cyxtal.geometry.Point.in_plane (*self, plane*)

```
Quick test if a point is in a given plane
```

5.7.2.5 def cyxtal.geometry.Point.on_line (*self, line*)

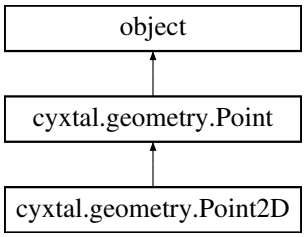
```
Quick test is the point is on the given line
```

The documentation for this class was generated from the following file:

- geometry.py

5.8 cyxtal.geometry.Point2D Class Reference

Inheritance diagram for cyxtal.geometry.Point2D:



Public Member Functions

- def **__init__** (self, x, y)
- def **__len__** (self)
- def **__str__** (self)

5.8.1 Detailed Description

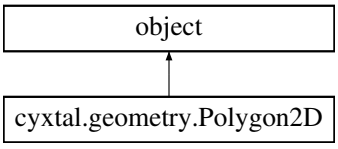
```
DESCRIPTION
-----------
Point2D(x,y)
    A 2D point (derived from the 3D Point class).
PARAMETERS
----------
METHODS
-------
CLASSMETHOD
-----------
```

The documentation for this class was generated from the following file:

- geometry.py

5.9 cyxtal.geometry.Polygon2D Class Reference

Inheritance diagram for cyxtal.geometry.Polygon2D:



Public Member Functions

- def __init__ (self)
- def __str__ (self)
- def **edges** (self)
- def **vertices** (self)
- def center (self)
- def add_vertex (self, point)
- def get_shortest (self)
- def contains_point (self, point, ray_origin=None)

5.9.1 Detailed Description

```
DESCRIPTION
-----------
Polygon2D()
    A 2D polygon class.
PARAMETERS
----------
edges: list
    List of segments/edges of the 2D polygon.
vertices: list
    List of 2D points serve as the vertices of the 2D polygon.
center: Point2D
    Gravity center of the polygon.
METHODS
-------
add_vertex(Point new_vtx)
    Add new vertex to self.
get_shortest()
    Return the shortest distance between the center and vertices.
contains_point(Point point, ray_origin=None)
    Test if given point lies inside self.
CLASSMETHOD
-----------
```

5.9.2 Constructor & Destructor Documentation

5.9.2.1 def cyxtal.geometry.Polygon2D.__init__ (*self*)

```
Initialize a 2D polygon with empty vertices list
```

5.9.3 Member Function Documentation

5.9.3.1 def cyxtal.geometry.Polygon2D.__str__ (*self*)

```
Formatted output for 2D polygon
```

5.9.3.2 def cyxtal.geometry.Polygon2D.add_vertex (*self, point*)

```
Add one more vertex to the current Polygon
```

5.9.3.3 def cyxtal.geometry.Polygon2D.center (*self*)

```
return the gravity center
```

5.9.3.4 def cyxtal.geometry.Polygon2D.contains_point (*self*, *point*, *ray_origin* = None)

```
quick test if a Point2D instance is inside the polygon.
```

5.9.3.5 def cyxtal.geometry.Polygon2D.get_shortest (*self*)

```
return the shortest distance between the center and vertices
```

The documentation for this class was generated from the following file:

- geometry.py

5.10 cyxtal.cxtallite.Quaternion Class Reference

5.10.1 Detailed Description

```
DESCRIPTION
-----------
Quaternion(np.array([w,x,y,z]))
    Quaternion is a set of numerics that extends from complex number,
    where a imaginary space (x,y,z) is constructed to facilitate a close
    set.
    Particularly, the unitary quaternions correspond to the rotation
    operation in 3D space, which is why many computer graphics used it
    to perform fast rotation calculations.
PARAMETERS
----------
q: DTYPE[:]
    Simple vector with length 4
METHODS
-------
unitary(self)
    Return a unitary quaternion, useful for using quaternion to represent
    rotation/orientation.
conj(self)
    Return the conjugate of the quaternion
tolist(self)
    Return the quaternion as a simple python list
tondarray(self)
    Return the quaternion as a numpy array (preferred)
toEulers(self)
    Convert a unitary quaternion into Euler Angles (np.ndarray)
toRodrigues(self)
    Convert a unitary quaternion into Rodrigue vector (np.ndarray)
toOrientationMatrix(self)
    Convert a unitary quaternion into Orientation Matrix (np.ndarray)
CLASSMETHOD
-----------
scale(Quaternion q, DTYPE_t scalar)
    Scale a quaternion vector with given scalar.
rotate(Quaternion q, DTYPE_t[:] pt)
    Rotate pt around origin by q.
average(list qs)
    Return an approximation of the average quaternion (forced to unitary)
    for qs (list of quaternions).
```

The documentation for this class was generated from the following file:

- cxtallite.pyx

5.11 cyxtal.cxtallite.Rodrigues Class Reference

5.11.1 Detailed Description

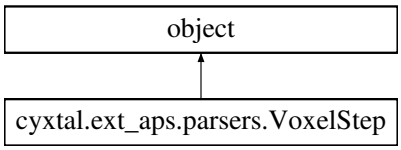
```
DESCRIPTION
-----------
Rodrigues representation of orientation, a wrapper class that use
Quaternion class as engine.
```

The documentation for this class was generated from the following file:

- cxtallite.pyx

5.12 cyxtal.ext_aps.parsers.VoxelStep Class Reference

Inheritance diagram for cyxtal.ext_aps.parsers.VoxelStep:



Public Member Functions

- def **__init__** (self)
- def **Xsample** (self)
- def **Xsample** (self, data)
- def **Ysample** (self)
- def **Ysample** (self, data)
- def **Zsample** (self)
- def **Zsample** (self, data)
- def **depth** (self)
- def **depth** (self, data)
- def **goodness** (self)
- def **goodness** (self, data)
- def **qs** (self)
- def qs (self, data)
- def **hkls** (self)
- def **hkls** (self, data)
- def **astar** (self)
- def **astar** (self, data)
- def **bstar** (self)
- def **bstar** (self, data)
- def **cstar** (self)
- def **cstar** (self, data)
- def **lc** (self)
- def **lc** (self, data)
- def **lattice** (self)
- def **lattice** (self, data)

- def **reciprocal_basis** (self)
- def validate (self, skip=False, tor=1e-2)
- def **__str__** (self)
- def get_coord (self, ref='TSL', translate=(0, 0, 0))
- def get_eulers (self, ref='TSL')
- def get_strain (self, ref='TSL', xtor=1e-8, disp=False, deviatoric=True, maxiter=1e4, opt_method='nelder-mead')

  step 4: transform strain tensor to requested configuration some preparation before hard computing
- def strain_refine (self, v_features)

Public Attributes

- **qs**

5.12.1 Detailed Description

```
DESCRIPTION
-----------
Container class for parsing through data, all the data is stored
as it is in the xml file. Additional methods are provided for
various other purposes.
PARAMETERS
----------
X|Y|Zsample: sample motor position during scan (X|Y|Z)
depth:       wire position
qs:          identified diffraction vectors
hkls:        hkl indices identified
a|b|cstar:   strain free reciprocal lattice identified
lc:          lattice constants used in indexation
lattice:     lattice structure
goodness:    the indexation goodness of first pattern (highest confidence)
_valid:      validation state of the voxel
```

5.12.2 Member Function Documentation

5.12.2.1 def cyxtal.ext_aps.parsers.VoxelStep.get_coord (*self, ref* = 'TSL', *translate* = (0, 0, 0))

```
DESCRIPTION
-----------
coord = self.get_coord(ref='TSL')
    Return the coordinates of the voxel in given reference
    system
PARAMETERS
----------
ref: string(case insensitive)
    Name for reference configuration ['TSL'|'APS'|'XHF']
translate: array
    Translate voxel with given translation vector after
    rotating to the desired reference system.
NOTE
----
The rotation matrix and orientation matrix are a very confusing
couple, especially when it comes to crystallography. This is
most due to the fact both the crystal and the reference are constantly
transform during crystallography calculation. The general rule of thumb
in determine which matrix should be used should be as follows:
    if crystal.rotated is True & reference.rotated is False:
use Rotation_Matrix
    elif reference.rotated is True & crystal.rotated if False:
use Orientation_Matrix
    else:
call divide_and_couqure()
    endif
```

5.12.2.2 def cyxtal.ext_aps.parsers.VoxelStep.get_eulers (*self, ref =* 'TSL')

```
DESCRIPTION
-----------
phi1, PhH, phi2 = self.get_eulers(ref='TSL')
PARAMETERS
----------
ref: string
    The configuration in which the Euler Angles is computed.
    The default output (a*,b*,c*) in the xml file is in the
    APS coordinate system according to
    http://www.aps.anl.gov/Sectors/33_34/microdiff/Instrument/coordinates-PE-system.pdf
RETURNS
-------
phi1,PHI,phi2: tuple
    Computed Euler angles in degrees
NOTE
----
The change of reference configuration will affect the output
of the Euler angle calculation, as a result, it is necessary
define what configuration/reference the calculation is in and
make sure all calculation is done under the same reference
configuration.
```

5.12.2.3 def cyxtal.ext_aps.parsers.VoxelStep.get_strain (*self, ref =* 'TSL', *xtor =* 1e-8, *disp =* False, *deviatoric =* True, *maxiter =* 1e4, *opt_method =* 'nelder-mead')

step 4: transform strain tensor to requested configuration some preparation before hard computing

```
DESCRIPTION
-----------
epsilon = self.get_strain(ref='TSL')
    Return strain tensor extracted/inferred through strain
    refinement process for current voxel. The returned strain
    tensor is transformed into designated coordinate system.
PARAMETERS
----------
ref:  str ['APS', 'TSL', XHF]
    The coordinate system in which the refined strain tensor
    will be returned.
xtor: float
    Tolerance used in the optimization of finding strained unit
    cell
disp: boolean
    Toggle the display of optimization process results
deviatoric: boolean
    Whether only returning the deviatoric strain components or
    full strain tensor
    !!!NOTE:
Full strain tensor requires energy beam scan data,
this particular feature has not implement yet.
maxiter: float
    Maximum iterations/calls allowed during the optimization
RETURNS
-------
epsilon: np.array (3,3)
    Green--Lagrange strain tensor in given reference configuration
NOTE
----
    The strain is approximated using the (a*,b*,c*), which are
    in the APS coordinate system.
```

5.12.2.4 def cyxtal.ext_aps.parsers.VoxelStep.qs (*self, data*)

```
DESCRIPTION
-----------
Q vectors much be stack in rows, the xml file from aps
are storing Q vectors by column.
```

5.12.2.5 def cyxtal.ext_aps.parsers.VoxelStep.strain_refine (*self, v_features*)

```
DESCRIPTION
-----------
rst = self.strain_refine(v_features)
    This is the objective function for the strain refinement.
PARAMETERS
----------
v_features: np.array
    feature vectors
    (a*_1, a*_2, a*_3, b*_1, b*_2, b*_3, c*_1, c*_2, c*_3)
RETURNS
-------
rst: float
    1-cos(q_calc, q_meas).
NOTE
----
    This approach is still under construction. Further change of
    the objective function is possible
```

5.12.2.6 def cyxtal.ext_aps.parsers.VoxelStep.validate (*self, skip* = `False`, *tor* = `1e-2`)

```
DESCRIPTION
-----------
self.validate()
    Validate all parameters are parsed;
    Prune q vectors, ensure correct mapping between
    self.hkls and self.qs;
    Instance of VoxelStep can only be used when validated.
    If strain refinement is not required, set skip=True for
    quick data process.
PARAMETERS
----------
skip: boolean
    This flag allow a simple bypass of the type check that
    ensures all attributes are properly assigned.
tor: float
    Tolerance for q vectors pruning.
RETURNS
-------
self._valid: boolean
    Return the state of the voxel (valid/invalid)
```

The documentation for this class was generated from the following file:

- ext_aps/parsers.py

5.13 cyxtal.cxtallite.Xtallite Class Reference

5.13.1 Detailed Description

```
DESCRIPTION
-----------
Composite class to represent material point in general crystal plasticity
simulation.

PARAMETERS
----------

METHODS
-------
```

The documentation for this class was generated from the following file:

- cxtallite.pyx

Index