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.cxtallite.OrientationMatrix Class Reference

5.3.1 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

5.4 cyxtal.cxtallite.Quaternion Class Reference

5.4.1 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

5.5 cyxtal.cxtallite.Rodrigues Class Reference

5.5.1 Detailed Description

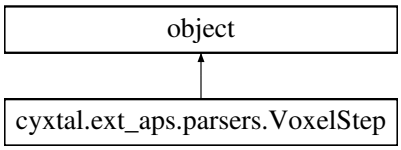
```
DESCRIPTION
-----------
Rodrigues representation of orientation, a wrapper class that use
Quaternion class as engine.
```

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

- cxtallite.pyx

5.6 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 **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=True, deviatoric=True, maxiter=1e4, weight=8e2, approximate=False, keep_volume=False)

  step 1: extract rotation (transformation).
- def strain_refine (self, lc, r, weight)
- def strain_refine_tischler (self, lc, r, lattice_c)

Public Attributes

- **qs**

5.6.1 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
_valid:         validation state of the voxel
```

5.6.2 Member Function Documentation

5.6.2.1 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
```

5.6.2.2 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.
```

5.6.2.3 def cyxtal.ext_aps.parsers.VoxelStep.get_strain (*self*, *ref* = 'TSL', *xtor* = 1e-8, *disp* = True, *deviatoric* = True, *maxiter* = 1e4, *weight* = 8e2, *approximate* = False, *keep_volume* = False)

step 1: extract rotation (transformation).

```
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
weight: float
    Fudge factor used to control the penalty term in the objective
    function of the optimization step (constrains on unit cell).
approximate: boolean
    Perform full calculation of the residual strain tensor or using
    simple approximation E = U - I
keep_volume: boolean
    Keep the volume to be constant throughout the strain refinement,
    which is suggested by Dr. Tischler at APS.
RETURNS
-------
epsilon: np.array (3,3)
    Strain tensor in given reference configuration
NOTE
----
    Since the strain is approximated using the (a*,b*,c*), which are
    in the APS coordinate system.
```

step 2: call scipy.optmize.minimize on the objective function self.get_qmismatch to find the ideal set of lattice constants that provide best match to measured Q vectors. Since Dr.Tischler suggested a different approach, additional implementation is provided for this case. step 3: calculate the stretch tensor using the deformation gradient Dr. Tischler is doing all the calculation in the reciprocal space, however the deformation gradient is in real space. Based on the derivation in the reference, the ref: cyxtal/documentation step 4: transform strain tensor to requested configuration some preparation before hard computing

5.6.2.4 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.
```

5.6.2.5 def cyxtal.ext_aps.parsers.VoxelStep.strain_refine (*self, lc, r, weight*)

```
DESCRIPTION
-----------
rst = self.strain_refine(lc, r, weight)
    This is the objective function for the strain refinement.
PARAMETERS
----------
lc: np.array
    lattice constant
r: np.array (3,3)
    transformation matrix (orientation matrix) that converts standard
    unit cell system to APS coordinate system
weight: float
    fudge factor in the penalty term that scales the effect of
    unit cell volume change
RETURNS
-------
rst:  float
    angular difference between calculated qs using new_lc and
    measurements (self.qs). A penalty term (delta_V) is added
    to ensure no large strain happens to the unit cell.
NOTE
----
    This approach is still under construction. Further change of
    the objective function is possible
```

5.6.2.6 def cyxtal.ext_aps.parsers.VoxelStep.strain_refine_tischler (*self, lc, r, lattice_c*)

```
Dr. Tischler implementation of strain refinement
NOTE:
    This method currently leads to unstable results
    (singular matrix)
```

5.6.2.7 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

5.7 cyxtal.cxtallite.Xtallite Class Reference

5.7.1 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