Code to verify and visualize reorientation caused twinning in the deformation gradient.

Using the correspondence matrix method.

By Satyapriya Gupta, Achal H P

Reference: Niewczas, Acta Materialia, 2010

[1]:
```
import math
import numpy as np
import scipy.linalg
from numpy.linalg import inv
import damask
```

Test Case: Extension Twinning $(\bar{1}012)[10\bar{1}1]$

Conversion to Miller indices: $(\bar{1}02)[211]$

[2]:
```
# Twin direction:
m = np.array([2, 1, 1])
# Habit plane normal
n = np.array([-1, 0, 2])
# Mapping from Miller indes to cartesian coordinate system
matA=np.array([[1, -0.5, 0],[0, 0.5*np.sqrt(3), 0],[0, 0, 1.624]])
#norm of twin direction
norm_mcart=np.linalg.norm(np.matmul(matA,m))
# norm of twin plane
norm_ncart=np.linalg.norm(np.matmul(n,inv(matA)))
# normalized unit vectors:
unit_mcart = np.matmul(matA,m)/norm_mcart
unit_ncart = np.matmul(n, inv(matA))/norm_ncart
# Characteristic shear for Extension twin of Mg
s = 0.128917
print(norm_mcart)
print(norm_ncart)
print(unit_mcart)
print(unit_ncart)
```

```
2.3743159014756228
1.6881920256873888
[0.63176092 0.36474734 0.68398649]
[-0.59234968 -0.34199325  0.72949468]
```

Please provide Euler Angles Here:

[3]:
```
phi_1 = 17
phi_ = 92
phi_2 = 170

init = (damask.Orientation.from_Euler_angles(
          phi=[phi_1,phi_,phi_2],
          degrees=True,
          family='hexagonal',
          lattice='hP',
          a=1.0,b=1.0,c=1.6235).as_matrix())
print(init)
```

```
[[-0.9400045  -0.29372535  0.1735424 ]
 [-0.17610919 -0.01790229 -0.98420783]
 [ 0.2921936  -0.9557222  -0.0348995 ]]
```

Initial Undeformed Condition, F = Identity matrix

[4]:
```
F = np.identity(3)
```

If $F = I$, then inverse of F gives F_e.

[5]:
```
Fp=init
Fe=inv(Fp)
Rinv = inv(init)
```

Shape change by twinning in the reference configuration:

[6]:
```
S_ref= (s * np.einsum('i,j',np.matmul(Rinv,unit_mcart),np.
 ↪matmul(Rinv,unit_ncart))
        + np.identity(3))
print("deformation gradient of shape change= \n", S_ref)
```

```
deformation gradient of shape change=
 [[ 0.95095672  0.03054648 -0.01230733]
 [-0.09052208  1.05638144 -0.02271637]
 [-0.02924172  0.01821313  0.99266184]]
```

After twinning, $F_1 = Fe_1 * F_{p_1}$, where $F_{p_1} = C * F_p$

[7]:
```
F1=S_ref
C=np.array([[-0.25,0.433013,-0.923645],[0.433013,-0.75,-0.533267],[-0.
 ↪812, -0.468808, 0]])
Fe1=np.matmul(F1,inv(np.matmul(C,Fp)))
print ("with determinant of ",np.linalg.det(Fe1))
```

```
with determinant of  1.000000166095149
```

Right polar decomposition of F_e, $F_e = RU$ gives the rotation of deformed voxel.

[8]:
```
(R_,U_) = scipy.linalg.polar(Fe1,'right')
check = (damask.Orientation.from_matrix(R=R_,
                                        family='hexagonal',lattice='hP',
                                        a=1.0,b=1.0,c=1.6235)
         .as_Euler_angles(degrees=True))
print(check)
```

```
[206.53799212  70.1926324   77.64540521]
```

Get the orientation details

[9]:
```
new_ori = (damask.Orientation.from_matrix(R=np.linalg.inv(R_),
                                          family='hexagonal',
                                          lattice='hP',a=1.0,b=1.0,c=1.
 ↪6235))
(print(damask.Orientation.from_matrix(R=np.linalg.inv(R_),
                                      family='hexagonal',lattice='hP',
                                      a=1.0,b=1.0,c=1.6235).
 ↪as_Euler_angles(degrees=True)))
```

```
[102.35459479  70.1926324  333.46200788]
```

[10]:
```
old_ori = (damask.Orientation.from_Euler_angles(phi=[phi_1,phi_,phi_2],
                                                ␣
 ↪degrees=True,family='hexagonal',
                                                 lattice='hP',a=1.0,b=1.
 ↪0,c=1.6235))
print(old_ori)
```

```
Crystal family: hexagonal
Bravais lattice: hP
a=1 m, b=1 m, c=1.6235 m
α=90°, β=90°, γ=120°
Quaternion [ 0.04240788 -0.16792654  0.69946438 -0.69336269]
```

[11]:
```
dis = old_ori.disorientation(new_ori)
print(dis)
print(dis.as_Euler_angles(degrees=True))
```

```
Crystal family: hexagonal
Bravais lattice: hP
a=1 m, b=1 m, c=1.6235 m
α=90°, β=90°, γ=120°
Quaternion [7.29494643e-01 6.83986524e-01 3.14974755e-08 1.07984812e-07]
[ 0.         86.31191012  0.        ]
```

Result: Print the disorientation between undeformed and deformed voxel.

[12]:
```
print('disorientation =',dis.
 ↪as_axis_angle(degrees=True,pair=True)[1],'degrees')
```

```
disorientation = 86.31191011698508 degrees
```

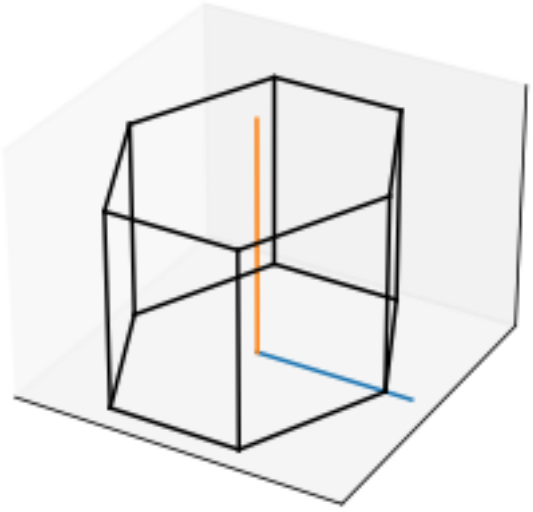
Code to visualization of lattice reorientation caused by twinning.

First cell below gives HCP unit cell with euler angles 0,0,0.

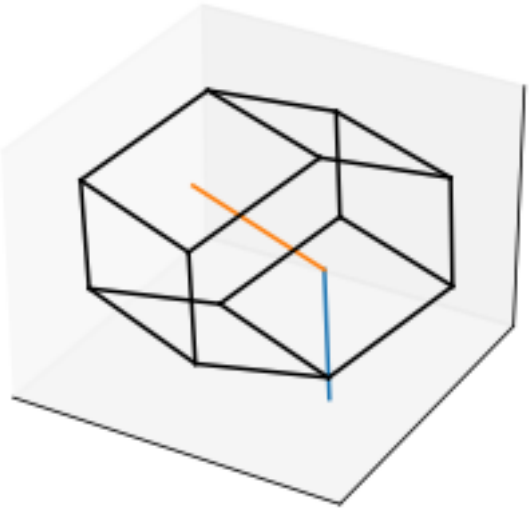
Second cell below rotates unit cell by provided euler angles.

Third cell below shows reorientation done after twinning.

[13]:
TL;DR : Code to generate hexagon with no rotation.
Full code available in : $https://github.com/achalhp/Validation_codes/tree/main$



[14]:
TL;DR : Code to generate hexagon with given rotation.
Full code available in : $https://github.com/achalhp/Validation_codes/tree/main$



[15]: TL;DR : Code to generate hexagon with twinned rotation.
Full code available in : $https://github.com/achalhp/Validation_codes/tree/main$