Demonstration of *hkl*

Demonstrate the [*hkl* (https://people.debian.org/~picca/hkl/hkl.html)](https://people.debian.org/~picca/hkl/hkl.html) package for diffractometer computations.

First, consider you have a sample of "*demo*" with known unit cell parameters, mounted on a 6-circle (non kappa) diffractometer:

a	b	c	α	β	γ
4.542	16.955	7.389	90	90	90

This example was derived from the unit test code: [https://repo.or.cz/hkl.git/blob/HEAD:/tests/bindings/polarisation.py (https://repo.or.cz/hkl.git/blob/HEAD:/tests/bindings/polarisation.py)](https://repo.or.cz/hkl.git/blob/HEAD:/tests/bindings/polarisation.py) with settings from [https://repo.or.cz/hkl.git/blob/refs/heads/next:/tests/bindings/crystal.ini (https://repo.or.cz/hkl.git/blob/refs/heads/next:/tests/bindings/crystal.ini)](https://repo.or.cz/hkl.git/blob/refs/heads/next:/tests/bindings/crystal.ini)

```
# these other parameters from crystal.ini


Wavelength 1.62751693358

R0 0 0.0 8.0 0.0 0 1 0.0 22.31594 89.1377 0.0 0.0 45.15857
R1 1 0.0 12.0 1.0 0 1 0.0 34.96232 78.3139 0.0 0.0 71.8007

Engine hkl

Mode constant_phi_vertical

PsiRef not available in current engine mode
```

In the console:

```
# make sure to sue the bash shell
bash

# use python3
source /APSshare/anaconda3/x86_64/bin/activate

# setup the custom hkl environment
(base) jemian@wow ~ $ . /APSshare/linux/64/hkl-5/hkl_environment.sh

# start jupyter
jupyter-notebook
```

You'll see jupyter start reporting messages in the console:

```
[I 11:21:25.136 NotebookApp] Serving notebooks from local directory: /home/JEMIAN
[I 11:21:25.136 NotebookApp] The Jupyter Notebook is running at:
[I 11:21:25.137 NotebookApp] http://localhost:8888/?token=______very_long_unique_hexa
decimal_code_here______
[I 11:21:25.137 NotebookApp] Use Control-C to stop this server and shut down all kern
els (twice to skip confirmation).
[C 11:21:25.143 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///run/user/970/jupyter/nbserver-24247-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=______very_long_unique_hexadecimal_code_here____
__
```

... more to come, ignore until you're done

imports

In [1]:
```
import os
import math
import unittest

from collections import namedtuple
from gi.repository import GLib
import gi
gi.require_version("Hkl", "5.0")
from gi.repository import Hkl
from numpy import (array, cross, dot, empty, hstack, reshape, vstack)
from numpy.linalg import inv, norm

H_NU = 12.3984244 # voltage*wavelength product, angstrom * keV
```

declare the sample lattice parameters

In [2]:
```
sample = Hkl.Sample.new("demo")
lattice = Hkl.Lattice.new(
    4.542,               # a, angstrom
    16.955,              # b, angstrom
    7.389,               # c, angstrom
    math.radians(90.0),  # alpha, radians
    math.radians(90.0),  # beta, radians
    math.radians(90.0)   # gamma, radians
)
sample.lattice_set(lattice)
```

In [3]:
```
wavelength = 1.62751693358
energy = H_NU / wavelength
```

In [4]:
```
detector = Hkl.Detector.factory_new(0)  # TODO: what other values?
```

In [5]:
```
# list the names of the diffractometers known to this version of *hkl*
factories = Hkl.factories()
print(f"diffractometers: {sorted(factories.keys())}")

# our diffractometer is Eulerian 6-circle (not kappa or Petra P23)
# define a "factory" since we'll use it later
diffractometer_type = "E6C"
e6c_factory = factories[diffractometer_type]

# axes: MU, OMEGA, CHI, PHI, GAMMA, DELTA
print(f"{diffractometer_type} axes: {e6c_factory.create_new_geometry().axis_nam
es_get()}")
```

```
diffractometers: ['E4CH', 'E4CV', 'E6C', 'K4CV', 'K6C', 'PETRA3 P09 EH2', 'PETR
A3 P23 4C', 'PETRA3 P23 6C', 'SOLEIL MARS', 'SOLEIL SIRIUS KAPPA', 'SOLEIL SIRI
US TURRET', 'SOLEIL SIXS MED1+2', 'SOLEIL SIXS MED2+2', 'SOLEIL SIXS MED2+3', '
SOLEIL SIXS MED2+3 v2', 'TwoC', 'ZAXIS']
E6C axes: ['mu', 'omega', 'chi', 'phi', 'gamma', 'delta']
```

In [6]:
```
# sample orientation and reflection 0 (0 8 0)
angles = [0.0, 22.31594, 89.1377, 0.0, 0.0, 45.15857]

geometry = e6c_factory.create_new_geometry()
geometry.axis_values_set(angles, Hkl.UnitEnum.USER)
geometry.wavelength_set(wavelength, Hkl.UnitEnum.USER)

or0 = sample.add_reflection(geometry, detector, 0, 8, 0)
```

In [7]:
```
# add reflection or1 (0 12 1)
angles = [0.0, 34.96232, 78.3139, 0.0, 0.0, 71.8007]
geometry.axis_values_set(angles, Hkl.UnitEnum.USER)
or1 = sample.add_reflection(geometry, detector, 0, 12, 1)
```

In [8]:
```
# Helper methods

def hkl_matrix_to_numpy(m):
    M = empty((3, 3))
    for i in range(3):
        for j in range(3):
            M[i, j] = m.get(i, j)
    return M


def from_numpy_to_hkl_vector(v):
    V = Hkl.Vector()
    V.init(v[0], v[1], v[2])
    return V
```

In [9]:
```
# compute UB with or0 and or1
sample.compute_UB_busing_levy(or0, or1)
UB = hkl_matrix_to_numpy(sample.UB_get())
```

In [10]:
```
# compute angles for reciprocal lattice vector h, k, l
engine_name = "hkl"
engines = e6c_factory.create_new_engine_list()
engines.init(geometry, detector, sample)

engine = engines.engine_get_by_name(engine_name)
print(engine.modes_names_get())

# pick our mode
engine.current_mode_set("constant_phi_vertical")
```

```
['bissector_vertical', 'constant_omega_vertical', 'constant_chi_vertical', 'con
stant_phi_vertical', 'lifting_detector_phi', 'lifting_detector_omega', 'lifting
_detector_mu', 'double_diffraction_vertical', 'bissector_horizontal', 'double_d
iffraction_horizontal', 'psi_constant_vertical', 'psi_constant_horizontal', 'co
nstant_mu_horizontal']
```

Out[10]:
```
1
```

That defines our diffractometer setup.

In [11]:
```
def calc(hkl):
    # assumes these are known: sample, geometry, detector, engine
    solutions = engine.pseudo_axis_values_set(hkl, Hkl.UnitEnum.USER)
    for i, s in enumerate(solutions.items()):
        values = s.geometry_get().axis_values_get(Hkl.UnitEnum.USER)
        # print(f"solution {i+1}: {values}")
    first_solution = solutions.items()[0]
    values = first_solution.geometry_get().axis_values_get(Hkl.UnitEnum.USER)
    # print("picking first one")
    return values

hkl = (0, 1, 1)
angles = calc(hkl)
print(f"{hkl}: {angles}")

for reflection in (or0, or1):
    hkl = reflection.hkl_get()
    angles = calc(hkl)
    print(f"{hkl}: {angles}")
```

```
(0, 1, 1): [0.0, 3.4824458166048444, 22.712897698011936, 0.0, 0.0, 13.799774663
132288]
(h=0.0, k=8.0, l=0.0): [0.0, 22.31594087562736, 89.13769999977886, 0.0, 0.0, 45
.158571742842376]
(h=0.0, k=12.0, l=1.0): [0.0, 34.963469180020944, 78.33265876350477, 0.0, 0.0,
71.80070421791422]
```

step scan from `or0` to `or1`

In [12]:

```
import numpy

num_points = 5
# get the sequence of hkl values for each Miller index
# reference by name, not position
# these are named tuples
_s = or0.hkl_get()
_f = or1.hkl_get()
_h = numpy.linspace(_s.h, _f.h, num_points)
_k = numpy.linspace(_s.k, _f.k, num_points)
_l = numpy.linspace(_s.l, _f.l, num_points)

import pyRestTable
table = pyRestTable.Table()
table.labels = "h k l".split()
table.labels += geometry.axis_names_get()

for hkl in zip(_h, _k, _l):
    angles = calc(hkl)
    print(f"({hkl[0]:6g} {hkl[1]:6g} {hkl[2]:6g}): {angles}")
    table.addRow(list(hkl) + angles)

# print(table)  # FIXME: https://github.com/prjemian/pyRestTable/issues/31
```

```
(     0      8      0): [0.0, 22.31594087562736, 89.13769999977886, 0.0, 0.0, 4
5.158571742842376]
(     0      9   0.25): [0.0, 25.15499515419808, 85.49774632991891, 0.0, 0.0, 5
1.29500599244316]
(     0     10    0.5): [0.0, 28.214868205373786, 82.60530989054361, 0.0, 0.0,
57.77618836086002]
(     0     11   0.75): [0.0, 31.482805809422153, 80.26265823893127, 0.0, 0.0,
64.60230764844736]
(     0     12      1): [0.0, 34.963469180020944, 78.33265876350477, 0.0, 0.0,
71.80070421791422]
```

In [ ]: