Basic Information on 23-ID-1

Wenjie Chen

November 20, 2017

0.1 Introduction

This is a brief report on NSLS-II CSX beamline 23-ID-1, Brookhaven National Laboratory. I came here to do research on $ZrTe_3$ from Nov. 15th to Nov. 22th in 2017.

There was a slideshow of introduction to this beamline written by Professor Li before, but when I came on site, some commands have been changed to meet a higher standard of Python programming style. On the other hand, some explanations in the old slideshow seemed unclear to me (e.g., the layout), so I think it might be better to write a new document. Therefore, here it comes.

If you have any questions, please feel free to contact me.[1]

0.2 The Diffractometer

0.2.1 Layout

There are three angles very important for the diffractometer. Together, they formed the geometry we required for the measurement.

angles	Descriptions
δ	The angle between the detector and the horizontal plane.
θ	The angle between the sample plane and the horizontal plane.
γ	The angle between the detector and the yz plane.

The coordinate system is shown in Fig.1, along with the angles mentioned above.

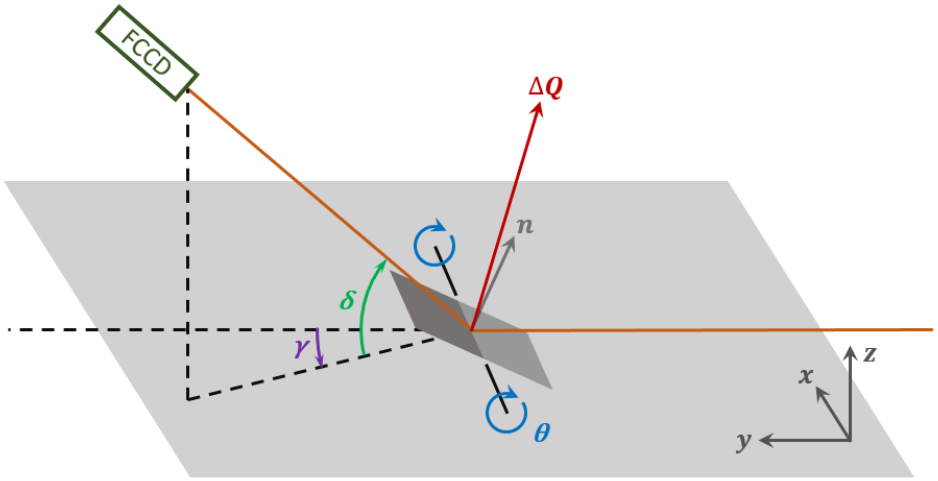

Figure 1: The diffractometer's layout.

0.2.2 Control and Movements

All movements are driven by motors. We can change both the angles and the position of the sample.

One thing you should keep in mind is that some commands make **relative** movements (relative to current angles or position), while others make **absolute** movements.

To change these angles, either type in GUI panels, or use the following commands in IPython:

[1]E-mail: wenjiechen@pku.edu.cn

```
tardis.position
    '''return the current reciprocal position of tardis'''
tardis.real_position
    '''return the current real position of tardis'''
tardis.forward(h, k, l)
    '''return the angles for observing (h, k, l) signal'''
tardis.move(h, k, l)
    '''move the tardis to receive (h, k, l) signal'''
```

To change the position of the sample, either type in GUI panels, or use the following commands in IPython:

```
RE(mv(sx, v1, sy, v2, sz, v3))
    '''move the sample to (x, y, z) = (v1, v2, v3)'''
```

0.2.3 Scan

According to **Bluesky**'s specification, all movements are done with "Run Engine". (Go to the last section for more information about **Bluesky**.) Therefore, basically every commands about movements will look like this:

```
RE(mv(variable1, value1, variable2, value2, ...))
```

Here the variables and values are not necessarily to have simple data structures. Instead, the variables are always defined to be some class, and the values are either float numbers or lists/-tuples. The assignments are automatically treated by the `mv(*args)` function.

0.3 Frequently Used Terms

In this section, several frequently used terms will be introduced in details.

Terms	Descriptions
tardis	The synchrotron diffractometer used here.

0.4 Frequently Used Functions

In this section, several frequently used Python functions will be introduced in details.

Terms	Descriptions
RE	abbr. for Run Engine. Used when operating hardware through codes.

0.5 "In-text" listing highlighting

```
class MyClass(Yourclass):
    def __init__(self, my, yours):
        bla = '5 1 2 3 4'
        print bla
```

0.6 External listing highlighting

0.7 Inline highlighting

Definition `class MyClass` means ...

0.8 More Information

NSLS-II has an organization on GitHub (`https://github.com/NSLS-II`). I recommend viewing it and saving it to your bookmarks.

The documentation on **bluesky** is also very useful (`http://nsls-ii.github.io/bluesky/`). Bluesky is a library for experiment control and collection of scientific data and metadata. There you can learn about how NSLS-II works with more details.