Diffcalc Developer Guide

Release 1.0

Diamond Light Source

December 10, 2015

CONTENTS

Author Rob Walton

Contact rob.walton (at) diamond (dot) ac (dot) uk

Website http://www.opengda.org/

Diffcalc: A diffraction condition calculator for diffractometer control

CHAPTER

ONE

INTRODUCTION

Diffcalc is a diffraction condition calculator used for controlling diffractometers within reciprocal lattice space. It performs the same task as the `fourc`, `sixc`, `twoc`, `kappa`, `psic` and `surf` macros from SPEC.

Diffcalc's standard calculation engine is an implementation of [You1999] . The first versions of Diffcalc were based on [Vlieg1993] and [Vlieg1998] and a 'Vlieg' engine is still available. The 'You' engine is more generic and the plan is to remove the old 'Vlieg' engine once beamlines have been migrated. New users should use the 'You' engine. [1]

The foundations for this type of calculation were laid by by Busing & Levi in their classic paper [Busing1967]. Diffcalc's orientation algorithm is taken from this paper. Busing & Levi also provided the original definition of the coordinate frames and of the U and B matrices used to describe a crystal's orientation and to convert between Cartesian and reciprical lattice space.

Geometry plugins are used to adapt the six circle model used internally by Diffcalc to apply to other diffractometers. These contain a dictionary of the 'missing' angles which Diffcalc uses to constrain these angles internally, and a methods to map from external angles to Diffcalc angles and visa versa.

Options to use Diffcalc:

- The *Quick-Start: Python API* section describes how to run up only the core in Python or IPython. This provides a base option for system integration.
- The *Quick-Start: Scanning* section describes how to start Diffcalc in Python in a way that provides a scan command and that also exposes user-level commands to the root namespace. This does not provide motor control, but does provide dummy software motor objects that could be easily replaced with real implementations for EPICS or TANGO for example.
- The *Quick-start: OpenGDA* section describes how to start Diffcalc within the Jython interpreter of an OpenGDA server. OpenGDA provides a scan command, a system for controlling motors and also a way to 'alias' user-level commands so that brackets and commas need not be typed, e.g typing:

  ```
  >>> addref [1 0 0]
  ```

  calls from the root namespace:

  ```
  >>> addref([1, 0, 0])
  ```

Diffcalc will work with Python 2.5 or higher with numpy, or with Jython 2.5 of higher with Jama.

[1] The very small 'Willmott' engine currently handles the case for surface diffraction where the surface normal is held vertical [Willmott2011]. The 'You' engine handles this case fine, but currently spins nu into an unhelpful quadrant. We hope to remove the need for this engine soon.

CHAPTER

TWO

PROJECT FILES & DIRECTORIES

diffcalc The main source package.

test Diffcalcs unit-test package (use Nose to run them).

numjy A *very* minimal implentation of numpy for jython. It supports only what Diffcalc needs.

doc The documentation is written in reStructuredText and can be compiled into html and pdf using Python's Sphinx. With Sphinx installed use `make clean all` from within the user and developer guide folders to build the documentation.

doc/references Includes links to relevant papers.

example Example startup scripts.

model Vrml models of diffractometers and a hokey script for animating then and controlling them from diffcalc.

CHAPTER

THREE

QUICK-START: PYTHON API

This section describes how to run up only the core in Python or IPython. Starting diffcalc without the additional functionality described in *Quick-Start: Scanning* or *Quick-start: OpenGDA* provides a good way to understand how the code is structured. It also provides an API which could be used to integrate Diffcalc into an existing data acquisition system; although the interface described in *Quick-Start: Scanning* would normally provide a better starting point.

For a full description of what Diffcalc does and how to use it please see the 'Diffcalc user manual'.

3.1 Setup environment

Change directory to the diffcalc project (python adds the current working directory to the path):

```
$ cd diffcalc
$ ls
COPYING  diffcalc  doc  example  mock.py  mock.pyc  model  numjy  test
```

If using Python make sure numpy and diffcalc can be imported:

```
$ Python 2.7.2+ (default, Oct  4 2011, 20:06:09)
[GCC 4.6.1] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> import diffcalc
```

If using Jython make sure Jama and diffcalc can be imported:

```
$ jython -Dpython.path=<diffcalc_root>:<path_to_Jama>/Jama-1.0.1.jar

Jython 2.2.1 on java1.5.0_11
Type "copyright", "credits" or "license" for more information.
>>> import Jama
>>> import diffcalc
```

3.2 Start

With Python start the sixcircle_api.py example startup script (notice the -i and -m) and type `demo_all()`:

```
$ python -i -m example/startup/sixcircle_api
>>> demo_all()
```

Or with IPython:

```
$ ipython -i example/startup/sixcircle_api.py
>>> demo_all()
```

Alternatively start Python or IPython and cut and paste lines from the rest of this tutorial:

```
$ python
$ ipython
```

3.3 Configure a diffraction calculator

To setup a Diffcalc calculator:

```
>>> from diffcalc.hkl.you.geometry import SixCircle
>>> from diffcalc.hardware import DummyHardwareAdapter
>>> from diffcalc.diffcalc_ import create_diffcalc

>>> hardware = DummyHardwareAdapter(('mu', 'delta', 'nu', 'eta', 'chi', 'phi'))
>>> dc = create_diffcalc('you', SixCircle(), hardware)
```

The hardware adapter is used by Diffcalc to read up the current angle settings, wavelength and axes limits. It is primarily used to simplify commands for end users. It could be dropped for this API use, but it is also used for the important job of checking axes limits while choosing solutions.

Geometry plugins are used to adapt the six circle model used internally by Diffcalc to apply to other diffractometers. These contain a dictionary of the 'missing' angles which Diffcalc internally uses to constrain these angles, and a methods to map from external angles to Diffcalc angles and visa versa.

3.4 Getting help

To get help for the orientation phase, the angle calculation phase, and the dummy hardware adapter commands:

```
>>> help(dc.ub)
>>> help(dc.hkl)
>>> help(hardware)
```

3.5 Orientation

To orient the crystal for example (see the user manual for a fuller tutorial) first find some reflections:

```
>>> # Create a new ub calculation and set lattice parameters
>>> dc.ub.newub('test')
>>> dc.ub.setlat('cubic', 1, 1, 1, 90, 90, 90)

>>> # Add 1st reflection
>>> dc.ub.c2th([1, 0, 0])                      # energy from hardware
60.
>>> hardware.position = 0, 60, 0, 30, 0, 0     # mu del nu eta chi phi
>>> dc.ub.addref([1, 0, 0])                    # energy and pos from hardware

>>> # Add 2nd reflection
>>> dc.ub.addref([0, 1, 0], [0, 60, 0, 30, 0, 90], en)
Calculating UB matrix.
```

To check the state of the current UB calculation:

```
>>> dc.ub.ub()

UBCALC

   name:          test

CRYSTAL

   name:         cubic

   a, b, c:    1.00000   1.00000   1.00000
              90.00000  90.00000  90.00000

   B matrix:   6.28319  -0.00000  -0.00000
               0.00000   6.28319  -0.00000 0.00000   0.00000 6.28319

UB MATRIX

   U matrix:   1.00000   0.00000   0.00000
              -0.00000   1.00000   0.00000
               0.00000   0.00000   1.00000

   UB matrix:  6.28319  -0.00000  -0.00000
              -0.00000   6.28319  -0.00000
               0.00000   0.00000   6.28319

REFLECTIONS

     ENERGY     H     K     L       MU    DELTA      NU     ETA     CHI     PHI  TAG
   1 12.398  1.00  0.00  0.00   0.0000 60.0000  0.0000 30.0000  0.0000  0.0000
   2 12.398  0.00  1.00  0.00   0.0000 60.0000  0.0000 30.0000  0.0000 90.0000
```

And finally to check the reflections were specified acurately:

```
>>> dc.checkub()
     ENERGY     H     K     L   H_COMP  K_COMP  L_COMP     TAG
 1  12.3984  1.00  0.00  0.00   1.0000  0.0000  0.0000
 2  12.3984  0.00  1.00  0.00   0.0000  1.0000  0.0000
```

3.6 Motion

Hkl positions and virtual angles can now be read up from angle settings (the easy direction!):

```
>>> dc.angles_to_hkl((0., 60., 0., 30., 0., 0.))      # energy from hardware
((1., 0.0, 0.0),
 {'alpha': 0.0,
  'beta': 0.0,
  'naz': 0.0,
  'psi': 90.0,
  'qaz': 90.0,
  'tau': 90.0,
  'theta': 29.999999999999996})
```

Before calculating the settings to reach an hkl position (the trickier direction) hardware limits must be set and combination of constraints chosen. The constraints here result in a four circle like mode with a vertical scattering plane and

incident angle 'alpha' equal to the exit angle 'beta':

```
 >>> dc.hkl.con('qaz', 90)
!   2 more constraints required
    qaz: 90.0000

 >>> dc.hkl.con('a_eq_b')
!   1 more constraint required
    qaz: 90.0000
    a_eq_b

 >>> dc.hkl.con('mu', 0)
    qaz: 90.0000
    a_eq_b
    mu: 0.0000
```

To check the constraints:

```
 >>> dc.hkl.con()
    DET         REF         SAMP
    ======      ======      ======
    delta   --> a_eq_b --> mu
    alpha       eta
--> qaz         beta        chi
    naz         psi         phi
                            mu_is_nu

    qaz: 90.0000
    a_eq_b
    mu: 0.0000

    Type 'help con' for instructions
```

Limits can be set to help Diffcalc choose a solution:

```
>>> hardware.set_lower_limit('delta', 0)       # used when choosing solution
```

Angles and virtual angles are then easily determined for a given hkl reflection:

```
>>> dc.hkl_to_angles(1, 0, 0)                          # energy from hardware
((0.0, 60.0, 0.0, 30.0, 0.0, 0.0),
 {'alpha': -0.0,
  'beta': 0.0,
  'naz': 0.0,
  'psi': 90.0,
  'qaz': 90.0,
  'tau': 90.0,
  'theta': 30.0})
```

CHAPTER

FOUR

QUICK-START: SCANNING

This section describes how to start Diffcalc in Python in a way that provides a scan command and that also exposes user-level commands to the root namespace. This does not provide motor control, but does provide dummy software motor objects that could be easily replaced with real implementations for EPICS or TANGO for example. The dummy software objects operate through a Scannable interface compatable with the OpenGDA's. gda.device.Scannable interface.

For a full description of what Diffcalc does and how to use it please see the 'Diffcalc user manual'.

4.1 Introduction to Scannables

Scannables are objects that can be operated by a scan or pos command.

4.2 Setup environment

Change directory to the diffcalc project (python adds the current working directory to the path):

```
$ cd diffcalc
$ ls
COPYING  diffcalc  doc  example  mock.py  mock.pyc  model  numjy  test
```

If using Python make sure numpy and diffcalc can be imported:

```
$ Python 2.7.2+ (default, Oct  4 2011, 20:06:09)
[GCC 4.6.1] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> import diffcalc
```

If using Jython make sure Jama and diffcalc can be imported:

```
$ jython -Dpython.path=<diffcalc_root>:<path_to_Jama>/Jama-1.0.1.jar

Jython 2.2.1 on java1.5.0_11
Type "copyright", "credits" or "license" for more information.
>>> import Jama
>>> import diffcalc
```

4.3 Start

With Python start the sixcircle_api.py example startup script (notice the -i and -m) and type `demo_all()`:

```
$ python -i -m example/startup/sixcircle
>>> demo_all()
>>> demo_scan()
```

Or with IPython:

```
$ ipython -i example/startup/sixcircle.py
>>> demo_all()
>>> demo_scan()
```

Alternatively start Python or IPython and cut and paste lines from the rest of this tutorial:

```
$ python
$ ipython
```

4.4 Configure a diffraction calculator and Scannables

Create some dummy motor Scannables and an energy Scannable:

```
>>> from diffcalc.gdasupport.minigda.scannable import SingleFieldDummyScannable
>>> mu = SingleFieldDummyScannable('mu')
>>> delta = SingleFieldDummyScannable('delta')
>>> nu = SingleFieldDummyScannable('nu')
>>> eta = SingleFieldDummyScannable('eta')
>>> chi = SingleFieldDummyScannable('chi')
>>> phi = SingleFieldDummyScannable('phi')

>>> en = SingleFieldDummyScannable('en')
```

Increase the priority of the energy scnannable so that it will be moved before `hkl` in scans (see below).

```
>>> en.level = 3
```

Build a Diffcalc calculator and associated Scannables and user-level commands:

```
>>> from diffcalc.gdasupport.factory import create_objects

>>> virtual_angles = ('theta', 'qaz', 'alpha', 'naz', 'tau', 'psi', 'beta')
>>> _objects = create_objects(
       engine_name='you',
       geometry='sixc',
       axis_scannable_list=(mu, delta, nu, eta, chi, phi),
       energy_scannable=en,
       hklverbose_virtual_angles_to_report=virtual_angles,
       simulated_crystal_counter_name='ct')
```

Add these to the root namespace for easy interactive use:

```
>>> from diffcalc.gdasupport.factory impor add_objects_to_namespace
>>> add_objects_to_namespace(_objects, globals())
>>> ================================================================
>>> Added objects/methods to namespace:
>>> addref, alpha, beta, c2th, calcub, checkub, chi_par, con,
```

```
>>> ct, dc, delref, delta_par, editref, eta_par, h, hardware, hkl,
>>> hklverbose, k, l, listub, loadub, mu_par, naz, newub, nu_par,
>>> phi_par, psi, qaz, saveubas, setcut, setlat, setmax, setmin,
>>> setu, setub, showref, sigtau, sim, sixc, swapref, trialub, ub,
>>> uncon, wl
>>> ============================================================
```

(This is equivilent to `globals().extend(_objects)` but checks for namespace collisions and does some reporting.)

4.5 Import pos and scan commands

To create a pos command for moving Scannables and a scan command for scannig them:

```
>>> from diffcalc.gdasupport.minigda import command
>>> pos = command.Pos(globals())
>>> scan = command.Scan(command.ScanDataPrinter())
```

4.6 Introduction to pos and scan

The pos command can be used to check the position of all scannables:

```
>>> pos()
sixc:     mu: 0.0 delta: 0.0 nu: 0.0 eta: 0.0 chi: 0.0 phi: 0.0
alpha:    Error: alpha
...
```

To check the position of a single scannable:

```
>>> pos(phi)
phi: 0.0

>>> phi                                # alternatively
phi: 0.0

>>> phi() + 100                        # call to get number
100
```

To move a scannable:

```
>>> pos(phi, 5)
phi:      5.0000
```

To perform a basic (and not very useful) scan for example:

```
>>> scan(phi, 0, 50, 10, chi, 5, eta)
Fri Mar 16 10:02:37 2012
================
     phi    delta
 -------  -------
  0.0000  0.0000
  0.0000  0.0000
 20.0000  0.0000
 30.0000  0.0000
 40.0000  0.0000
 ================
```

4.7 Getting help

To get help for the orientation phase, the angle calculation phase, and the dummy hardware adapter commands:

```
>>> help(ub)
>>> help(hkl)
```

4.8 Orientation

To orient the crystal for example (see the user manual for a fuller tutorial) first find some reflections:

```
>>> # Create a new ub calculation and set lattice parameters
>>> newub('test')
>>> setlat('cubic', 1, 1, 1, 90, 90, 90)

>>> # Add 1st reflection
>>> pos(wl, 1)
>>> c2th([1, 0, 0])
60.
>>> pos(sixc, [0, 60, 0, 30, 0, 0])
sixc:     mu: 0.0 delta: 60.0 nu: 0.0 eta: 30.0 chi: 0.0 phi: 0.0
>>> addref([1, 0, 0])

>>> # Add 2nd reflection
>>> pos(phi, 90)
>>> addref([0, 1, 0])
Calculating UB matrix.
```

To check the state of the current UB calculation:

```
>>> ub()

UBCALC

   name:          test

CRYSTAL

   name:         cubic

   a, b, c:    1.00000   1.00000   1.00000
              90.00000  90.00000  90.00000

   B matrix:   6.28319   0.00000   0.00000
               0.00000   6.28319   0.00000
               0.00000   0.00000   6.28319

UB MATRIX

   U matrix:   1.00000   0.00000   0.00000
               0.00000   1.00000   0.00000
               0.00000   0.00000   1.00000

   UB matrix:  6.28319   0.00000   0.00000
               0.00000   6.28319   0.00000
               0.00000   0.00000   6.28319
```

```
REFLECTIONS

     ENERGY     H     K     L       MU   DELTA      NU     ETA     CHI     PHI  TAG
   1 12.398  1.00  0.00  0.00   0.0000 60.0000  0.0000 30.0000  0.0000  0.0000
   2 12.398  0.00  1.00  0.00   0.0000 60.0000  0.0000 30.0000  0.0000 90.0000
```

And finally to check the reflections were specified acurately:

```
>>> checkub()
     ENERGY     H     K     L   H_COMP  K_COMP  L_COMP     TAG
 1  12.3984  1.00  0.00  0.00   1.0000  0.0000  0.0000
 2  12.3984  0.00  1.00  0.00   0.0000  1.0000  0.0000
```

4.9 Motion

Hkl positions and virtual angles can now be read up from angle settings (the easy direction!):

```
>>> pos(hkl)
hkl:      h: 0.00000 k: 1.00000 l: 0.00000

>>> pos(hklverbose)
hklverbose: h: 0.00000 k: 1.00000 l: 0.00000
            theta: 30.00000 qaz: 90.00000 alpha: -0.00000 naz: 0.00000
            tau: 90.00000 psi: 90.00000 beta: 0.00000
```

Before calculating the settings to reach an hkl position (the trickier direction) hardware limits must be set and combination of constraints chosen. The constraints here result in a four circle like mode with a vertical scattering plane and incident angle 'alpha' equal to the exit angle 'beta':

```
>>> con(qaz, 90)
!   2 more constraints required
    qaz: 90.0000

>>> con(a_eq_b)
!   1 more constraint required
    qaz: 90.0000
    a_eq_b

>>> con(mu, 0)
    qaz: 90.0000
    a_eq_b
    mu: 0.0000
```

To check the constraints:

```
>>> con()
    DET         REF         SAMP
    ======      ======      ======
    delta  --> a_eq_b --> mu
    alpha       eta
--> qaz         beta        chi
    naz         psi         phi
                           mu_is_nu

    qaz: 90.0000
    a_eq_b
    mu: 0.0000
```

```
    Type 'help con' for instructions
```

Limits can be set to help (or in somfe cases alow) Diffcalc choose a solution:

```
>>> setmin(delta, 0)
>>> setmin(chi, 0)
```

The hkl scannable can now be moved which will in turn move sixc and the underlying motors mu, delta, nu, eta, chi and phi:

```
>>> pos(hkl,dd [1, 0, 0])
hkl:      h: 1.00000 k: 0.00000 l: 0.00000

>>> sixc

sixc:
     mu :    0.0
  delta :   60.0
     nu :    0.0
    eta :   30.0
    chi :    0.0
    phi :    0.0

>>> hkl
hkl:
    hkl :    1.0000  0.0000  0.0000
  alpha :    0.0
   beta :    0.0
    naz :    0.0
    psi :   90.0
    qaz :   90.0
    tau :   90.0
  theta :   30.0
```

Alternatively h, k and l can moved:

```
>>> pos(k, 1)
k:        1.00000

>>> hkl

hkl:
    hkl :    1.0000  1.0000  0.0000
  alpha :   -0.0
   beta :    0.0
    naz :    0.0
    psi :   90.0
    qaz :   90.0
    tau :   90.0
  theta :   45.0
```

4.10 Scanning

Scannables can be moved in very generic ways to construct many types of scan.

Scan delta through hkl=100 while counting a counter timer, ct, for 1s and reporting the resulting hkl position:

```
>>> pos(hkl, [1, 0, 0])
hkl:      h: 1.00000 k: 0.00000 l: 0.00000

>>> scan(delta, 40, 80, 10, hkl, ct, 1)

-------  -------  -------  -------  -------
  delta        h        k        l       ct
-------  -------  -------  -------  -------
40.0000  0.67365  0.11878  0.00000  0.00959
50.0000  0.84202  0.07367  0.00000  0.87321
60.0000  1.00000  0.00000  0.00000  3.98942
70.0000  1.14279 -0.09998  0.00000  0.87321
80.0000  1.26604 -0.22324  0.00000  0.00959
```

Scan h from hkl=010 to hkl=110 while counting a counter timer for 1s and reporting h and l and all the axes postions:

```
>>> pos(hkl, [0, 1, 0])
hkl:      h: -0.00000 k: 1.00000 l: 0.00000

>>> scan(h, 0, 1, .2, k, l, sixc, ct, 1)

--------  -------  -------  ------  -------  ------  -------  ------  -------  -------
       h        k        l      mu    delta      nu      eta     chi      phi       ct
--------  -------  -------  ------  -------  ------  -------  ------  -------  -------
 0.00000  1.00000  0.00000  0.0000  60.0000  0.0000  30.0000  0.0000  90.0000  3.98942
 0.20000  1.00000  0.00000  0.0000  61.3146  0.0000  30.6573  0.0000  78.6901  0.53991
 0.40000  1.00000  0.00000  0.0000  65.1654  0.0000  32.5827  0.0000  68.1986  0.00134
 0.60000  1.00000  0.00000  0.0000  71.3371  0.0000  35.6685  0.0000  59.0362  0.00134
 0.80000  1.00000  0.00000  0.0000  79.6302  0.0000  39.8151  0.0000  51.3402  0.53991
 1.00000  1.00000  0.00000  0.0000  90.0000  0.0000  45.0000  0.0000  45.0000  3.98942
```

Rotate about the hkl=101 reflection while counting a counter timer for .1s and reporting the three unconstraied sample angles:

```
>>> con(psi)
    qaz: 90.0000
!   psi: ---
    mu: 0.0000

>>> scan(psi, 0, 90, 10, hkl, [1, 0, 1], eta, chi, phi, ct, .1)

     psi      eta      chi      phi        h        k        l       ct
 -------   ------  -------  -------  -------  -------  -------  -------
 0.00000   0.0000  90.0000  90.0000  1.00000  0.00000  1.00000  0.39894
10.00000   0.4385  82.9470  82.8929  1.00000  0.00000  1.00000  0.39894
20.00000   1.7808  76.0046  75.5672  1.00000  0.00000  1.00000  0.39894
30.00000   4.1066  69.2952  67.7923  1.00000  0.00000  1.00000  0.39894
40.00000   7.5463  62.9660  59.3179  1.00000  0.00000  1.00000  0.39894
50.00000  12.2676  57.2022  49.8793  1.00000  0.00000  1.00000  0.39894
60.00000  18.4349  52.2388  39.2315  1.00000  0.00000  1.00000  0.39894
70.00000  26.1183  48.3589  27.2363  1.00000  0.00000  1.00000  0.39894
80.00000  35.1489  45.8640  14.0019  1.00000  0.00000  1.00000  0.39894
90.00000  45.0000  45.0000   0.0000  1.00000  0.00000  1.00000  0.39894
```

CHAPTER

FIVE

QUICK-START: OPENGDA

5.1 Install

Copy the Diffcalc folder into the gda root folder (i.e. so it sits alongside plugins, thirdparty etc.)

5.2 Start diffcalc

Diffcalc is started from the command line or it can be started in localStation automatically. To start a dummy sixcircle installation:

```
>>> diffcalc_path = gda.data.PathConstructor.createFromProperty("gda.root").split('/plugins')[0]
    + '/diffcalc'
>>> execfile(diffcalc_path + '/example/startup/sixcircle_dummy.py')

/scratch/ws/8_4/diffcalc added to GDA Jython path.
================================================================================
Created DummyPD: alpha
Created DummyPD: delta
Created DummyPD: gamma
Created DummyPD: omega
Created DummyPD: chi
Created DummyPD: phi
Created diffractometer scannable: sixc
Created dummy energy scannable:  en
Set dummy energy to 1 Angstrom
Created wavelength scannable: wl
Created hkl scannable: hkl
Created hkl component scannables: h, k & l
Created verbose hkl scannable: hklverbose. Reports virtual angles: 2theta, Bin, Bout, azimuth
Created parameter scannable: phi_par
Created parameter scannable: alpha_par
Created parameter scannable: oopgamma
Created parameter scannable: betaout
Created parameter scannable: azimuth
Created parameter scannable: betain
Created parameter scannable: blw
Aliased command: addref
Aliased command: autosector
Aliased command: calcub
Aliased command: checkub
Aliased command: dcversion
```

```
Aliased command: delref
Aliased command: editref
Aliased command: handleInputError
Aliased command: helphkl
Aliased command: helpub
Aliased command: hklmode
Aliased command: listub
Aliased command: loadub
Aliased command: mapper
Aliased command: newub
Aliased command: raiseExceptionsForAllErrors
Aliased command: saveubas
Aliased command: sector
Aliased command: setcut
Aliased command: setlat
Aliased command: setmax
Aliased command: setmin
Aliased command: setpar
Aliased command: setu
Aliased command: setub
Aliased command: showref
Aliased command: sigtau
Aliased command: sim
Aliased command: swapref
Aliased command: trackalpha
Aliased command: trackgamma
Aliased command: trackphi
Aliased command: transforma
Aliased command: transformb
Aliased command: transformc
Aliased command: ub
Aliased command: diffcalcdemo
================================================================================
Try the following:
newub 'cubic'
setlat 'cubic' 1 1 1 90 90 90
pos wl 1
pos sixc [0 90 0 45 45 0]
addref 1 0 1
pos phi 90
addref 0 1 1
checkub
ub
hklmode

Or type 'diffcalcdemo' to run this script (Caution, will move the diffractometer!)
================================================================================
Added objects/methods to namespace: gamma, trackalpha, phi, diffcalc_object, editref,
transformc, newub, setub, setpar, transforma, setu, off, autosector, dcversion, betaout,
sector, swapref, showref, setmin, trackgamma, ub, oopgamma, transformb, handleInputError,
l, listub, chi, manual, helpub, helphkl, azimuth, wl, setlat, sim, trackphi, alpha,
sigtau, omega, raiseExceptionsForAllErrors, saveubas, delref, hklmode, calcub, blw,
k, setcut, en, diffcalcdemo, sixc, hklverbose, addref, h, delta, betain, setmax, auto,
checkub, hkl, mapper, on, loadub, phi_par, alpha_par
```

Notice that this example script creates dummy scannables for the six axes and energy.

To use preexisting scannables modify:

```
  diffcalcObjects = createDiffcalcObjects(
      dummyAxisNames = ('alpha', 'delta', 'gamma', 'omega', 'chi', 'phi'),
      dummyEnergyName = 'en',
      geometryPlugin = 'sixc',
      hklverboseVirtualAnglesToReport=('2theta','Bin','Bout','azimuth'),
      demoCommands = demoCommands
  )
```

to::

```
  diffcalcObjects = createDiffcalcObjects(
      axisScannableList = (alpha, delta, gamma, omega, chi, phi),
      energyScannable = en,
      geometryPlugin = 'sixc',
      hklverboseVirtualAnglesToReport=('2theta','Bin','Bout','azimuth'),
      demoCommands = demoCommands
  )
```

Check out the user manual doc/user/manual.html . Also type diffcalcdemo to run the example session displayed above.

CHAPTER

SIX

DEVELOPMENT

The files are kept here on github. See bootcamp for an introduction to using github. To contribute please fork the project. Otherwise you can make a read-only clone or export.

Code format should follow pep8 guidelines. PyDev has a good pep8 checker.

To run the tests install _nose, change directory into the test folder and run:

```
$ nosetests
.......... ...
----------------------------------------------------------------------
Ran 3914 tests in 9.584s

OK (SKIP=15)
```

CHAPTER

SEVEN

THANKS

I would like to acknowledge the people who have made a direct impact on the Diffcalc project, knowingly or not, in terms of encouragement, suggestions, criticism, bug reports, code contributions, and related projects.

Names are ordered alphabetically by surname.

Allesandro Bombardi, Mark Booth, Busing, Steve Collins, Levy, Martin Lohmier, Chris Nicklin, Elias Vlieg — writer of DIF software used as a model for Diffcalc, Robert Walton, You.

Thank you!

Rob Walton

CHAPTER

EIGHT

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

BIBLIOGRAPHY

[You1999] H. You. *Angle calculations for a '4S+2D' six-circle diffractometer.* J. Appl. Cryst. (1999). **32**, 614-623. (pdf link).

[Busing1967] W. R. Busing and H. A. Levy. *Angle calculations for 3- and 4-circle X-ray and neutron diffractometers.* Acta Cryst. (1967). **22**, 457-464. (pdf link).

[Vlieg1993] Martin Lohmeier and Elias Vlieg. *Angle calculations for a six-circle surface x-ray diffractometer.* J. Appl. Cryst. (1993). **26**, 706-716. (pdf link).

[Vlieg1998] Elias Vlieg. *A (2+3)-type surface diffractometer: mergence of the z-axis and (2+2)-type geometries.* J. Appl. Cryst. (1998). **31**, 198-203. (pdf link).

[Willmott2011] C. M. Schlepütz, S. O. Mariager, S. A. Pauli, R. Feidenhans'l and P. R. Willmott. *Angle calculations for a (2+3)-type diffractometer: focus on area detectors.* J. Appl. Cryst. (2011). **44**, 73-83. (pdf link).