Laue Lens Library

Guida per l'utente

Alessandro Pisa
pisa@fe.infn.it

31 Marzo 2008

1 Introduzione

Questo manuale descrive il software da me sviluppato negli ultimi anni per la ricerca sulle Lenti di Laue (**LL**). Il codice è scritto in Python secondo e segue gli standard di codifica previsti dalla **PEP 257**[1].
Tale caratteristica rende possibile utilizzare il programma `pydoc`[2] per avere una descrizione rapida e immediata delle funzioni e delle classi del programma.
Il programma è stato sviluppato in maniera da poter accettare i seguenti parametri di ingresso:

1. Lunghezza focale;
2. Banda di lavoro;
3. Materiale da utilizzare per i cristalli e sue caratteristiche fisiche;
4. Lunghezza e larghezza dei cristalli;
5. Algoritmo di distribuzione dei cristalli;
6. Eventuali limitazioni definite dall'utente;
7. Offset della sorgente;

e con il fine di fornire i seguenti risultati:

1. Posizioni e dimensioni dei cristalli nella lente;
2. Peso e volume dei cristalli;
3. Area efficacie;
4. Sensibilità;
5. Focusing factor;
6. Equivalent surface;
7. Point Spread Function per sorgenti in asse o fuori asse.

L'architettura del codice e' stata studiata con attenzione per soddisfare le seguenti richieste:

1. Facilità di utilizzo;
2. Flessibilità;
3. Modularità;
4. Riusabilità del codice;
5. Input/Output semplificato.

A tal fine il software si può immaginare costituito da strati. L'utente che volesse usare il programma nel modo standard dovrà interagire solo con lo strato più' esterno, senza curarsi di quelli più interni.
Per richieste non standard sarà richiesta l'interazione con gli strati più interni del codice.

2 Il modulo Lenses

L'utente dovrà interagire principale con il modulo della libreria chiamato `Lenses`. Tale modulo contiene la definizione della classe `Generic`, e delle seguenti classi da essa derivate:

1. `Spiral`;
2. `Ring`;
3. `Rings`;
4. `SingleXtal`;

1 http://www.python.org/dev/peps/pep-0257/
2 http://docs.python.org/lib/module-pydoc.html

Ogni classe, compresa `Generic`, vuole emulare una lente costruita secondo determinati principi. Una spiegazione dettagliata delle classi viene fornita nella sezione 4.1.
In quasi tutti i casi le classi derivate sono alternative alla classe `Generic` che non differiscono in termini di output e di sintassi, ma che permettono di diminuire drasticamente il tempo di calcolo sfruttando algoritmi ottimizzati. Per esempio una lente ad anelli può essere simulata sia utilizzando opportunamente la classe `Generic` che la classe `Rings`, ma quest'ultima fornirà i risultati in tempo inferiore poiché nel calcolo sfrutterà opportunamente la simmetria cilindrica dell'oggetto in questione.

3 Utilizzo del programma

Il programma può essere utilizzato in 3 modi:

1. Facendo eseguire i comandi dall'interprete Python;
2. Creando uno script eseguibile;
3. Usando una shell interattiva.

Tali procedure sono le stesse adottate dalla maggior parte dei linguaggi di programmazione standard. Per una panoramica dell'argomento relativa a Python si rimanda alla lettura del Python Tutorial[3].
Esempio: Ecco un esempio di come funziona il codice spiegato riga per riga.

```
[ale@gri ~]$python
Python 2.5.1 (r251:54863, Jan 14 2008, 11:52:06)
[GCC 4.2.2 20070909 (prerelease) (4.2.2-0.RC.1mdv2008.0)] on
linux2
Type "help", "copyright", "credits" or "license" for more
information.
>>> from Laue import Lenses
>>> lens=Lenses.Generic()
>>> print lens
Geometry:                spiral
Profile:                 spherical
Thickness profile:       fixed
Thickness threshold:     (0, 10)
Thickness factor:        1.0
Focal Length:            2.0 m
Inner/Outer radius:      0.0198022596511/0.118864548854 m
Z (Atomic number):       29
Energy range:            [100, 600] keV
hkl:                     (1,1,1)
d_hkl:                   2.08706348809 A
Cell volume:             47.2377130159 A^3
Structure factor:        88.2881011963
Microcrystal size:       0.0 micron
A_0 constant:            0.0 keV
Material factor constant: 5.04344594644 cm^-1
```

3 http://docs.python.org/tut/tut.html

```
Crystal tile size:        1x1x0.4 cm^3
Number of crystals:       198
Filling factor:           0.409238210167
Crystals volume:          79.2 cm^3
Crystals weight:          0.706464 kg
mosaicity (FWHM):         1.0 arcmin
Frame width:              0.2 cm
Misorientation (FWHM):    0.0 arcmin
>>>
```

```
[ale@gri ~]$python
Python 2.5.1 (r251:54863, Jan 14 2008, 11:52:06)
[GCC 4.2.2 20070909 (prerelease) (4.2.2-0.RC.1mdv2008.0)] on
linux2
Type "help", "copyright", "credits" or "license" for more
information.
```

Inizialmente apriamo una shell Python con il comando omonimo (`python`). In seguito al caricamento vengono mostrate informazioni di base sulla eseguibile e infine viene ritornato il classico prompt Python (`>>>`).

```
>>> from LLL import Lenses
```

Successivamente importiamo dalla libreria Laue le lenti e ci apprestiamo ad inizializzarne una che verrà immagazzinata in una variabile a nostro piacere (in questo caso `lens`)

```
>>> lens=Lenses.Generic()
```

Ora l'oggetto `lens` è a nostra disposizione. Per conoscere le informazioni relative a questo oggetto è sufficiente il comando:

```
>>> print lens
```

Molte di queste informazioni sono proprietà dell'oggetto.

```
Geometry:                 spiral
Profile:                  spherical
Thickness profile:        fixed
Thickness threshold:      (0, 10)
Thickness factor:         1.0
Focal Length:             2.0 m
Inner/Outer radius:       0.0198022596511/0.118864548854 m
Z (Atomic number):        29
Energy range:             [100, 600] keV
hkl:                      (1,1,1)
d_hkl:                    2.08706348809 A
Cell volume:              47.2377130159 A^3
Structure factor:         88.2881011963
Microcrystal size:        0.0 micron
A_0 constant:             0.0 keV
```

```
Material factor constant: 5.04344594644 cm^-1
Crystal tile size:        1x1x0.4 cm^3
Number of crystals:       198
Filling factor:           0.409238210167
Crystals volume:          79.2 cm^3
Crystals weight:          0.706464 kg
mosaicity (FWHM):         1.0 arcmin
Frame width:              0.2 cm
Misorientation (FWHM):    0.0 arcmin
```

L'oggetto in questione viene inizializzato con parametri di default che è possibile modificare all'atto della chiamata del costruttore, come si vedrà in seguito.

4 Descrizione delle classi principali contenute nella libreria

La peculiarità della Laue Lens Library risiede nel fatto che è possibile risposte alle domande che più frequentemente si ripropongono nello studio delle lenti di Laue con poche righe di codice. Questo è permesso dal fatto che in libreria sono disponibili oggetti corrispondenti alle lenti che più comunemente vengono simulate e che grazie a queste è possibile realizzare nuove lenti dalle caratteristiche più eterogenee mediante il meccanismo della ereditarietà.

4.1 Generic

Come visto nell'esempio precedente creare una lente di tipo `Generic` e' equivalente a chiamarne il costruttore. Il costruttore può essere invocato con argomenti opzionali. Gli argomenti opzionali sono i seguenti[4]:

`name (“lens”)`: una stringa identificativa del nome della lente. Tale stringa verrà usata come prefisso dei nomi dei file di output.

`geo (“spiral”)`: una stringa identificativa dell'algoritmo di disposizione dei cristalli. I valori che questa stringa può' assumere sono: `”spiral”`, `”one ring”`, `”rings”`, il cui significato e' autoesplicativo.

`profile (“spherical”)`: una stringa identificativa dell'algoritmo di disposizione dei cristalli. I valori che questa stringa può assumere sono: `”spherical”`, `”parabola”`, `”flat”`, il cui significato e' autoesplicativo.

`thickness_profile (“fixed”)`: una stringa identificativa dell'algoritmo con cui viene deciso lo spessore dei cristalli. I valori che questa stringa può assumere sono: `“fixed”`, `“optimized”` il cui significato e' autoesplicativo. I valori dello spessore sono comunque soggetti al vincolo del parametro opzionale `thickness_threshold` (descritto in seguito).

`thickness_threshold ((0, 10))`: un oggetto iterabile contente i limiti (in cm) inferiore e superiore per lo spessore del cristallo.

`thickness_factor (1.)`: specifica un coefficiente moltiplicativo per aumentare o diminuire lo spessore del cristallo. Utile per diminuire la massa della lente.

`datadir (“data”)`: cartella, esistente, nella quale verranno creati i file di output.

4Tra parentesi viene indicato il valore di default

`Focal (2000.)`: lunghezza focale della lente (cm).
`Emin (100.)`: estremo inferiore della banda nominale di energia (keV).
`Emin (600.)`: estremo superiore della banda nominale di energia (keV).
`dim((1., 1., 0.4))`: un oggetto iterabile che specifica le dimensioni dei cristalli (cm). Lo spessore (terza componente dell'oggetto) verrà poi modificato dagli algoritmi del programma in funzione della posizione del cristallo sulla lente.
`framewidth (0.20)`: spazio da riservare attorno al cristallo (cm).
`fwhm (1.)`: Larghezza a mezza altezza della distribuzione dei piani cristallini (arcmin).
`hkl ((1,1,1))`: indici di Miller dei piani principali del cristallo.
`Z (29)`: Numero atomico del materiale cristallino.
`microthick (0.)`: spessore in micron dei microblocchi cristallini. 0 significa condizione ottimale.
`xtalctrls ([])`: funzioni per modificare la presenza o meno dei cristalli su di una lente (deprecato).
`pixels ((50,50))`: numero di pixel (x,y) sul detector.
`pitches ((0.2,0.2))`: Dimensione dei pixel (x,y) sul detector (cm).
`error (0.)`: FWHM dell'errore (si suppone una distribuzione gaussiana) nel posizionamento dei cristalli (arcmin)
Un programma funzionante che utilizza questa classe si può trovare negli esempi (cartella **generic**, file **macro.py**).

4.2 Ring (derivata da Generic)

Eredita tutte le caratteristiche della classe genitrice con le seguenti differenze:

1. il valore di default dell'argomento `name` del costruttore è `“ring”`.
2. La geometria ovviamente non è più definibile, visto che la classe è stata specificatamente sviluppata per un singolo anello. L'argomento opzionale `geo` è per questo motivo stato rimosso.
3. Essendo un unico anello di cristalli identici la banda di lavoro è molto più stretta. Per questo motivo gli argomenti opzionali `Emin` ed `Emax` sono sostituiti da due nuovi argomenti, sempre opzionali ma mutualmente esclusivi: `energy` e `radius`. Entrambi hanno valore di default `None`, ma almeno uno dei due va settato ad un valore float. Per `energy` è l'energia per cui l'anello deve essere efficacie (keV), mentre `radius` permette di specificare il raggio dell'anello (cm). Nel caso entrambi siano specificati, `radius` ha la precedenza.

La classe sfrutta la simmetria cilindrica della lente per eseguire i calcoli in maniera più rapida.
Un programma funzionante che utilizza questa classe si può trovare negli esempi (cartella **ring**, file **macro.py**).

4.3 Rings (derivata da Generic)

Eredita tutte le caratteristiche della classe genitrice con le seguenti differenze:

1. il valore di default dell'argomento `name` del costruttore è `“rings”`.
2. La geometria ovviamente non è più definibile, visto che la classe è stata specificatamente sviluppata per una disposizione dei cristalli ad anelli concentrici. L'argomento opzionale `geo`

è per questo motivo stato rimosso.

La classe sfrutta la simmetria cilindrica della lente per eseguire i calcoli in maniera più rapida.

Un programma funzionante che utilizza questa classe si può trovare negli esempi (cartella **rings**, file **macro.py**).

4.4 Spiral (derivata da Generic)

Eredita tutte le caratteristiche della classe genitrice con le seguenti differenze:

1. il valore di default dell'argomento `name` del costruttore è `"spiral"`.
2. La geometria ovviamente non è più definibile, visto che la classe è stata specificatamente sviluppata per una disposizione lungo una spirale dei cristalli. L'argomento opzionale `geo` è per questo motivo stato rimosso.

Un programma funzionante che utilizza questa classe si può trovare negli esempi cartella **spiral**, file **macro.py**).

4.5 SingleXtal (derivata da Generic)

Eredita tutte le caratteristiche della classe genitrice con le seguenti differenze:

1. il valore di default dell'argomento `name` del costruttore è `"one"`.
2. La geometria ovviamente non e' più' definibile, visto che la classe e' stata specificatamente sviluppata per un singolo cristallo. L'argomento opzionale `geo` è per questo motivo stato rimosso.
3. Essendo un unico anello di cristalli identici la banda di lavoro è molto più stretta. Per questo motivo gli argomenti opzionali `Emin` ed `Emax` sono sostituiti da due nuovi argomenti, sempre opzionali ma mutualmente esclusivi: `energy` e `radius`. Entrambi hanno valore di default `None`, ma almeno uno dei due va settato ad un valore float. Per `energy` è l'energia per cui l'anello deve essere efficacie (keV), mentre `radius` permette di specificare il raggio dell'anello (cm). Nel caso entrambi siano specificati, `radius` ha la precedenza.

Un programma funzionante che utilizza questa classe si può trovare negli esempi cartella **one**, file **macro.py**).

4.6 Creare una nuova lente

E' possibile sfruttare le classi finora definite per derivare da esse nuove classi con caratteristiche diverse. La classe derivata permette di implementare in modo nuovo alcuni metodi della genitrice.

Un programma funzionante che utilizza una classe derivata si può trovare negli esempi (cartella **custom**, file **macro.py**). Nell'esempio viene mostrato come modificare la geometria di disposizione a spirale scartando tutti i cristalli che sono al di fuori di due rami di iperbole.

5 Descrizione del metodo macro

Dopo l'inizializzazione di un oggetto lente, l'utente può ottenere gran parte delle informazioni chiave sulle proprietà della stessa utilizzando il metodo macro.

Il metodo macro crea un file di log nella cartella specificata dall'argomento `datadir` del costruttore.

Nella stessa cartella vengono creati dei file sequenziali con prefisso dato dall'argomento `name` del costruttore, suffisso numerico ed estensione "`dat`" contenenti valori dai in formato ASCII e altri con estensione "`plot`" contenenti istruzioni per il programma gnuplot, in grado di generare grafici dei relativi dati. Seguendo questa logica, i nomi dei file riportati in seguito nel testo, vanno intesi nel seguente modo: al poste di "`NAME`" si dovrà sostituire il prefisso e al posto del carattere "`#`" il suffisso numerico.
Il metodo macro accetta i seguenti argomenti opzionali (tra parentesi il valore di default):
`INFO (False)`: se posto uguale a `True`, restituisce informazioni dettagliate sull'oggetto esaminato invocando il metodo `info()` dell'oggetto lente, altrimenti non fa nulla.
File di output: "`NAME_info_#.dat`"

`XTALINFO (False)`: se posto uguale a `True`, crea un file ASCII contenente una riga per ogni cristallo. Per ciascuna riga sono tabulati i dati relativi alla posizione e all'orientazione dei cristalli.
File di output: "`NAME_xtalinfo_#.dat`", "`NAME_xtalinfo_#.plot`".

`ERANGE (False)`: se è diverso da `False` genera una lista di valori di energia secondo la logica della funzione `range`[5]. Le proprietà dell'ottica dipendenti dall'energia verranno calcolate per i valori forniti da questa lista. Ad esempio `macro(AREA=True, ERANGE=(600, 700))` significa che verrà calcolata l'area efficacie per le energie che vanno da 600 a 700 keV ad intervalli di un keV. Il valore di default equivale a `ERANGE=(Emin, Emax [,1])`, dove i primi due argomenti sono i limiti della banda di lavoro nominale della lente, mentre il terzo argomento (opzionale), corrisponde allo step di default della funzione `range`.

`AREA (False)`: se `True`, crea un file ASCII contenente l'area efficacie della lente in funzione dei valori di energia descritti da `ERANGE`.
File di output: "`NAME_area_#.dat`", "`NAME_area_#.plot`".

`MC (False)`: se `True`, crea file ASCII contenente la distribuzione dei fotoni diffratti sul piano focale ("`NAME_mc_#.dat`"), la loro distribuzione radiale ("`NAME_mc_r_#.dat`"), l'integrale di quest'ultima ("`NAME_mc_encircled_#.dat`"), la distribuzione lungo i due assi ("`NOME_mc_x_X.dat`") ed ("`NOME_mc_y_X.dat`"). Inoltre crea i corrispettivi file per Gnuplot.

`SENS (False)`: Se `True`, crea un file ASCII contenente la sensibilità della lente in funzione dei valori di energia descritti da `ERANGE`.
Nota: Necessita `MC=True`.
File di output: "`NAME_sens_#.dat`", "`NAME_sens_#.plot`"

`S (False)`: Se `True`, crea un file ASCII contenente la superficie equivalente della lente in funzione dei valori di energia descritti da `ERANGE`.
Nota: Necessita `MC=True`.

5 Descritta all'indirizzo: http://docs.python.org/lib/built-in-funcs.html

File di output: “NAME_S_#.dat”, “NAME_S_#.plot”

G (False): Se True, crea un file ASCII contenente la superficie equivalente della lente in funzione dei valori di energia descritti da ERANGE.
Nota: Necessita MC=True.
File di output: “NAME_G_#.dat”, “NAME_G_#.plot”

PHOTONS (0): Numero di fotoni da simulati dal Monte Carlo

OFFSET (0.0): Offset della sorgente (arcmin)

FRACTION (0.5): Frazione dei fotoni considerata nel calcolo della sensibilità.

Tobs (1.e6): Tempo di osservazione (s)