DIALS FRAMEWORK

Graeme Winter[1], David Waterman[2], Gwyndaf Evans[1]

[1]Diamond Light Source

[2]CCP4

14th March 2013

DIALS

A framework for developing diffraction data analysis software.

CONTENTS

INTRODUCTION

This document provides a description of the DIALS (Diffraction Integration fro Advanced Light Sources) programming framework, developed as part of the BioStruct-X (www.biostruct-x.org) EU funded project. The framework structure is described with details and some examples of programming interfaces are given. The API definitions for the framework are not published here, as they will evolve towards their final state over the four-year duration of the BioStruct-X project. However module boundaries are well defined within DIALS and the development and coding of modules such as refinement and integration is already underway using the framework.

BACKGROUND AND AIMS

In DIALS we aim to deliver of a collaborative framework that will provide a future proof basis for the development of diffraction data analysis (principally integration in the first instance) aimed mainly at, but not strictly limited to, macromolecular crystallography (MX) at Synchrotron Radiation (SR) and Free Electron Laser (FEL) sources.

The modular philosophy of DIALS will enable developers to be responsive in adapting DIALS to the trends and demands of MX. We aim to provide users or expert systems with the choice of algorithm best suited to the type of data being analyzed and also empower the developers of beamline hardware, such as detectors, to write custom code modules particularly adapted to their own systems.

FRAMEWORK DESCRIPTION

We have developed a generalized programming framework that will allow us to develop a diffraction integration program capable of handling both synchrotron and FEL MX data. Our new framework, DIALS (Diffraction Integration for Advanced Light Sources) is broadly based on the very natural distinction made between the physically different components of a diffraction experiment: the Crystal, Source, Goniometer and Detector. We take as a basis the scheme for generalised, abstract description of these components first described in the reports on the EEC Cooperative Workshop on Position-Sensitive Detector Software coordinated by Gérard Bricogne between 1986 and 1987 [1-3].

To avoid “reinventing the wheel” with more general crystallographic calculations (e.g. relating to crystal unit cell and symmetry, reflection reindexing etc.) we have adopted CCTBX [4] as a basis for the developments, as this has a compatible architecture and provides many of the more fundamental tools for the analysis out-of-the-box. Where necessary we have made contributions to CCTBX to address deficiencies (such as the contribution of dxtbx) as well as bug fixes. Finally, the adoption of CCTBX as a basis provides a build mechanism and coding protocols, allowing the developer team to focus on the scientific goals of the project.

ARCHITECTURE

Figure 1 illustrates how these distinct components are brought together via a reciprocal lattice profile model, a wedge model describing a rotation method experiment and the interaction of the reciprocal lattice and Ewald sphere to generate a prediction in a

virtual detector plane. A particular method for integration is shown, based on the projection of the passage of a modeled reciprocal space profile through the Ewald sphere onto the detector plane. However, the DIALS framework ensures that we are not limited to one method and the scheme shown could equally well be replaced by conventional summation integration, or 2D [5] or 3D [6] profile-fitting methods.

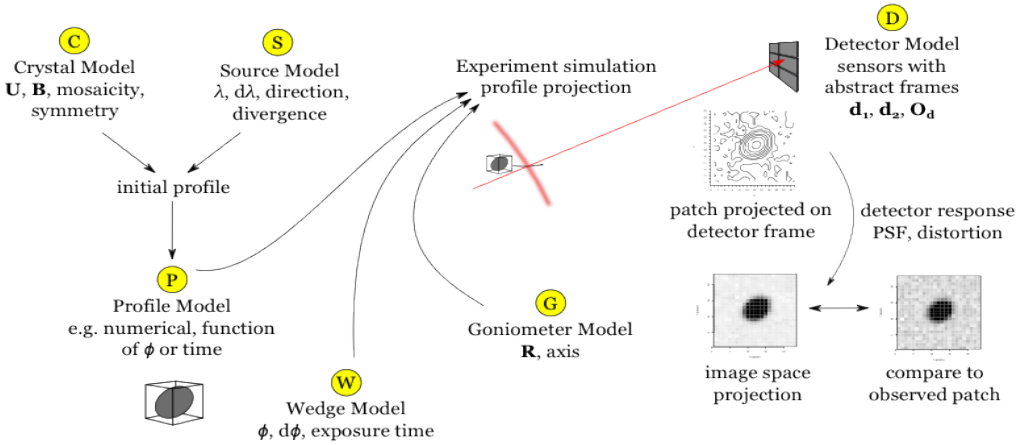

Figure 1. Schematic diagram illustrating the breakdown of the diffraction experiment into separable components that can be treated independently from a parameterization perspective. This would ultimately permit a physics-based approach to the modeling of diffraction, whereby the complete experiment is described and the observed and predicted diffraction spots are compared.

The program itself is separated into distinct modules: centroid prediction, refinement of the experimental geometry, profile formation and spot integration. In addition, the integration is explicitly separated from the global model refinement (see Figure 2) allowing for flexible control at run-time, to balance speed and quality of the results. The key modules communicate *via* the experimental models and an underlying observation database structure, into which can also be put prediction information such as reflection *shoeboxes* (either in image space or in reciprocal space) that describe that region of diffraction image space sufficient for integrating any given reflection. The details of this observation database structure are still under development but will form a key part of the communication with downstream analysis, for example detailed investigation of outlying observations.

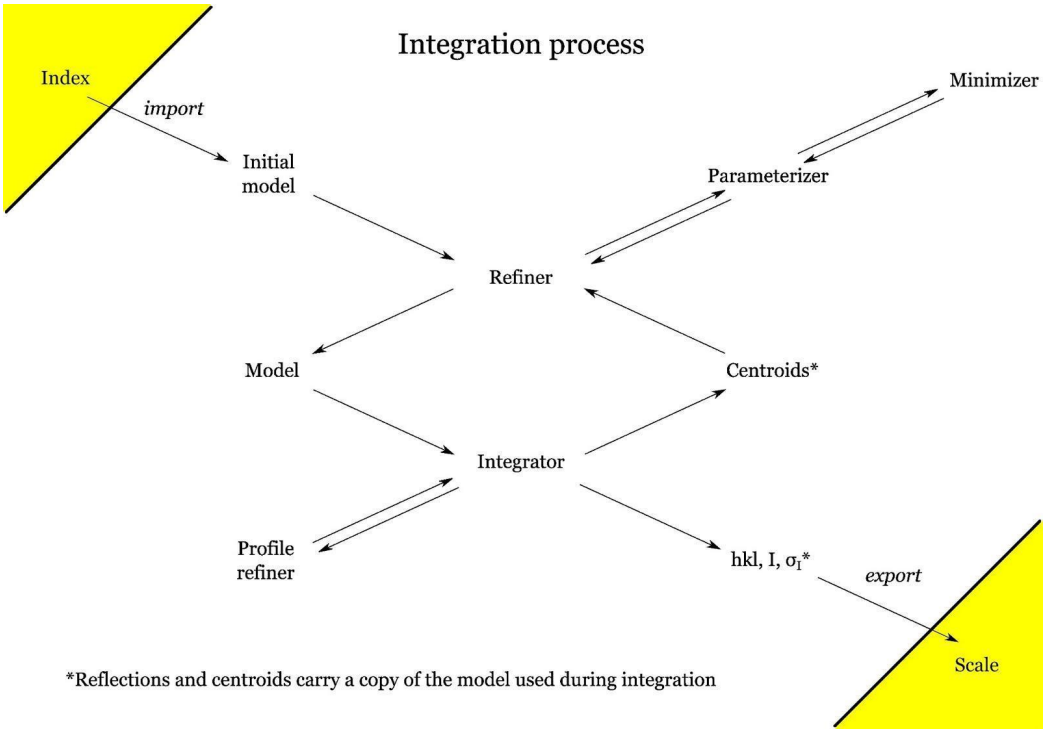

Figure 2. Distinct modules in DIALS linked by the underlying experimental models and observation data. Arrows illustrate the general direction of workflow through the program in traversing from input indexed images to output integrated intensities and their error estimates.

The experimental models and the observation data structure essentially define and inform the interfaces by which the distinct modules interact. Our design enables implementation of different types of existing integration approaches, 2D and 3D integration or even ray-tracing style integration as implemented in EVAL15 [7]. Furthermore we exchange refinement modules suitable for rotation data collection for those more matched to the refinement of XFEL diffraction data. In addition, knowledge of spot profiles can be gathered and utilized in a variety of ways: either by learning profiles from observed data or by building analytical descriptions of profiles in reciprocal space.

This extensibility is achievable because the main product of our framework development is a *toolbox*. This *toolbox* defines the language of data reduction (which includes the abstract models, the internal, flexible observation data structure, and the interfaces between modules). Integration functionality is then provided to the user through a relatively lightweight program which joins *toolbox* components together or an expert system making calls to the *toolbox* to achieve the necessary integration tasks.

Development using Use Cases

Clearly the correct design of these interfaces will be critical to the success of the project. To design these *ab initio* presents a substantial challenge and a risk to the project, so we have taken an alternative approach by designing the interfaces through the implementation of the algorithms in as simple a way as possible. Analysis of the resulting implementations provides insight into the necessary interface specification as well as identification of common elements between algorithms and the areas where

differences in the interface design are critical. This work has started with a small number of use cases: prediction of reflections, integration *via* 2-dimensional summation integration and reimplementation of the reciprocal space transformations used in XDS [6]. In each case the results are compared with the output of well trusted existing data processing packages, namely Mosflm and XDS, to ensure that the results are correct.

EXPERIMENTAL MODELS

Underpinning the distinct functional modules within DIALS is a shared vector description of diffraction geometry, including models for the beam, crystal, goniometer and detector, and their combination into the reflection prediction equation. These models have been carefully designed to maximise their generality, by avoiding hardware-specific features and restrictive choices of conventions where possible. We do not limit DIALS models to any particular idealised experimental geometry, such as the orthogonal beam and rotation axis of the canonical experiment using the rotation method. In fact, the experimental models do not rely on any special choice of coordinate frame and are equally well expressed in any. That said, the interface to dxtbx (see below) explicitly uses the internationally agreed imgCIF [8] conventions for the presentation of the experimental geometry, to simplify integration into other packages.

These ideas are not new to the DIALS project and we have borrowed heavily from the idea of the "instrument definition language" described in a report on the EEC Cooperative Programming Workshop on Position-Sensitive Detector Software [3]. In that report and associated documents the notion of device "virtualisation" was presented, in which diverse hardware options can be unified by their accommodation within a general framework. Although that idea was expressed in the context of development of software by procedural programming in a language without inbuilt support for object orientation, we find that it maps very closely to modern OOP practices. In particular, the "virtualisation" of devices may be achieved by careful design of an abstract base class, from which specialisations derive by means of interface inheritance. Despite the prescience of the workshop authors, in the quarter of a century since, the full potential of their ideas has not been met by any open source software for crystallographic data reduction. Within DIALS we intend to exploit the ideas fully. Specifically within the data modelling, we ensure we are not limited to a single detector surface. By contrast, the detector model provides a hardware-independent basis for any detector that can reasonably be described by a set of one or more flat panels, arranged with arbitrary position and orientation offsets from one another. Thus, the DIALS model will directly accommodate a complex composite PAD architecture such as an array of panels arranged in a barrel, or unique designs such as those in active development at specific FEL beamlines.

THE DIFFRACTION EXPERIMENT TOOLBOX: DXTBX

Early in the project the difficulty of interpreting diffraction data in a unified manner was identified. This arises from two sources: the range of instruments available for MX and the different usage of common elements in the image header information from those instruments. The former represents an administrative problem of collating the image header and data descriptions from a variety of manufacturers. The latter is a greater

challenge: to interpret the different usages of header elements across individual instruments in a universal way. The dxtbx addresses both of these challenges by defining a hierarchical system of interface classes, starting with a base class and "standard" instances for commonly used instruments. More specific code for an individual instrument can be added, without modification of any code in dxtbx, to allow customized interpretation of header elements ensuring consistent meaning of the resulting description. These classes are managed by a registry that provides the *best* code for interpreting images from a given instrument, providing the initial definitions of the models, namely the experimental geometry, source information and the sweep description in the DIALS standard model.

The result of this is that all instrument specific code is isolated from the rest of the framework, allowing straightforward extension to new instruments in the future. We intend to use a similar mechanism to support actual detector technology (e.g. the photon detection process) to properly support the differences between CCD, pixel array detectors and those developed in the future. Finally, the choice to develop dxtbx as a stand-alone toolbox within CCTBX rather than explicitly as a component of the DIALS framework was made to simplify the use of the software in other applications and encourage adoption (and hence testing) of the toolbox as soon as possible.

Reflection Prediction

The experimental models provide the means to express the reflection prediction equation (in X, Y, phi) in a fully vectorial, modular way. Each model's contribution to the reflection prediction equation is neatly separated from the others and it is this lends itself to future extension, e.g. the introduction of different numbers of detector panels or more than one crystal lattice. The first derivatives of the prediction equation with respect to an arbitrary parameter are also factorized into separate parts relating to each of the component models. This fact means that it is not just the description of the geometry that can be broken into component parts, but that encapsulation is extended to code that deals with the refinement of the geometry as well.

Parameter Refinement

Refinement Module

An optimization problem may be broken into three parts:

1. Objective function (in this case, the reflection prediction equation)
2. Target function (for example, a least squares sum of squared residuals, plus penalty/constraint terms)
3. Minimiser (e.g. LBFGS, BFGS or full normal matrix least squares routines).

These three parts should be kept separate, where possible, so that a modular system is formed in which any one part may be simply substituted without affecting the other two.

Abstract classes named PredictionParameterisation, Target, and Refinery define the interaction between these objects. They communicate *via* a parameter set, the objective function evaluated at a position in parameter space, and derivatives of the objective function. In practice, the target function is dependent on the objective, since they both depend upon the range of the prediction formula (the vector space with basis X, Y, phi). This implies that changes to the parameterisation of the prediction equation, such as its expression in reciprocal space rather than 'detector space', will require some modification to the target object too. However, multiple targets could be defined for any one choice of prediction equation parameterisation, and the minimization engine is completely independent of the choices of each of the other components.

Of the three basic components of the refinement module, only the objective function needs any knowledge of what the chosen parameters mean. In fact, because the vectorial prediction equation can be factorized into independent, abstract models, the objective function delegates all responsibility for parameterisation of the equation into individual model parameterisation classes. Thus any hardware- or experiment-specific code is pushed down to the lowest level, concrete instances of an abstract model parameterisation class. This keeps the interfaces, which are defined between the higher level, abstract classes, free from 'pollution' by specifics.

There is naturally a distinction between the *models* and the *model parameterisations* (as shown in Figure 3). The models are shared as part of the overall framework. The parameterisations have a scope limited to the refinement module only. Individual concrete parameterisations could be specific to hardware. In particular, it is expected that the more exotic paneled detector systems would be best parameterised in a way that directly exploits knowledge of spatial relationships between and groupings of the individual detector panels.

The distinction between model and model parameterisation enforces encapsulation of the code required only by the refinement module and ensures that particular hardware- or experiment-specific descriptions do not infiltrate the purely abstract description shared by other elements of the framework. It also frees the model parameterisations from the rigours required in maintaining an abstract description. Model parameterisations become temporary objects that can be exchanged for more appropriate versions at greater detail as more information about the experiment becomes available. Or a new version of the same object could be instantiated effectively resetting the values of its parameters if large shifts are detected during refinement of the models. These features enable a great degree of flexibility within the refinement scheme, without detrimental effect to any other aspect of the program.

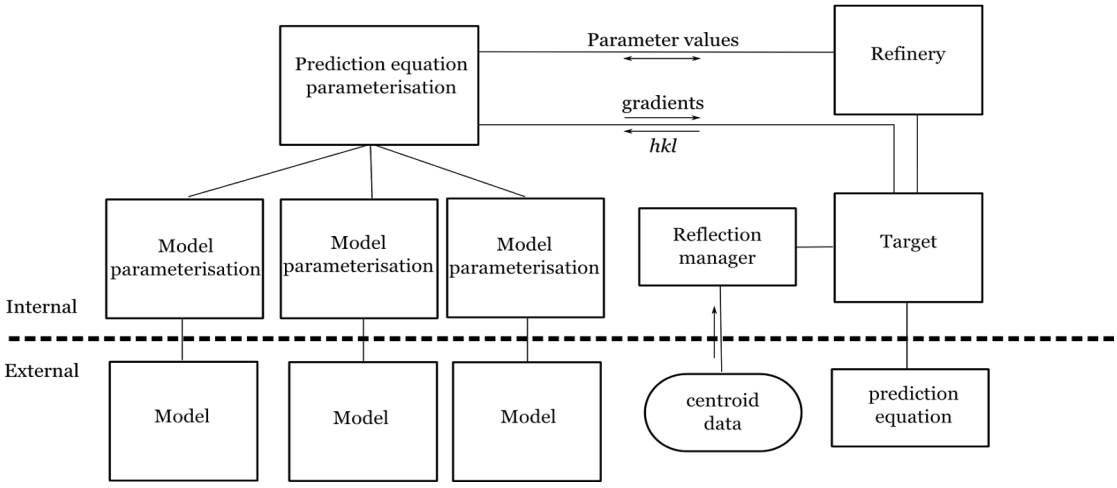

Figure 3. Overview of the refinement module illustrating communication between abstract classes.

An auxiliary object called the 'reflection manager' supports the refinement module. This entity is responsible for filtering the input centroid data to detect and reject outliers, and observations with limited value for model refinement, such as those determined to be too close to the spindle axis in a rotation experiment. The reflection manager also controls access to observation data, performing matching of predictions at each step to their observed positions and returning residuals to the target function.

Near-future extensions to the refinement module will introduce a 'refinement manager' to work alongside the reflection manager. The purpose of this object is to control decision-making at a higher level, taking responsibility for setting macrocycles. The need for this is highlighted by the observation that it is much more efficient to start refinement with a limited set of reflections to bring an initially bad model closer to truth quickly, and then finish with macrocycle(s) using more data. As well as requesting different size data sets from the reflection manager, the refinement manager will allow different choices of minimiser or minimiser parameters for different macrocycles, in order to optimise the usual trade-off between speed and radius of convergence in any optimisation problem.

REFINEMENT PARAMETERISATIONS

The refinement module currently under development for DIALS is designed on the hypothesis that joint, global refinement using all (useful) data, with physically meaningful parameters, will give superior results to separate positional/angular, local refinement. In the latter case, correlated parameters are allowed to "mop up" effects whose source is indeterminate, leading to individual deviation from physical values. DIALS refinement uses all available data and a detailed physical model of the experiment to tease apart the effects of different parameters and refine to the true values that are in principle measurable quantities. This means also that parameters that truly are expected to be fixed during an experiment, such as detector position/orientation parameters and the beam direction (at least for standard rotation experiments), will be refined as such to point values. By contrast, parameters that are known or expected to change, such as the crystal unit cell and possibly crystal

orientation, are allowed to do so within a suitably chosen smoothing level for the class of experiment and knowledge of the sample properties.

Apart from the crystal unit cell parameterization, the chosen parameter set is directly related to geometry of the diffraction experiment, and all parameters are either distances of translation, or angles of rotation about vector directions. We restrict the refinement to those aspects of the models that actually affect the centroid positions of predicted reflections. That is, we avoid consideration of parameters relating to the reflection profile, such as beam divergence, mosaicity and other profile shape parameters. That separates the refinement of experiment geometry from that of more intricate modelling of reflection profiles, which is more naturally considered an aspect of profile-fitting integration. Nevertheless, we recognise that any scheme for profile construction, refinement and fitting may result in a model that can improve on the initially determined centroid positions. Therefore for maximum accuracy we envisage a loop between centroid refinement, profile construction, and integration with iterative improvements to convergence (as illustrated by Figure 2).

The parameter sets are chosen carefully so that each model parameterisation relates only to the underlying model that it parameterizes. For example, with a complete parameterisation of the detector position and orientation, and another for the beam direction, it is not correct to treat the beam centre on the detector as additional parameters. These are simply quantities that are derived as needed from the refined geometry exported from the refinement module to the experimental models.

As an example, a basic implementation of parameterisation for a multi-panel detector is shown in Figure 4. Here the component panels of the detector are mutually positioned and orientated with known translational and angular offsets. The group of panels is treated as a rigid block, with a full six degrees of freedom expressed by its parameter set. Although simple, this basic parameterisation is expected to satisfy most use cases within the field of area-detector crystallography at synchrotrons, including complex arrangements of composite panels, such as V-shaped or barrel detectors. Where additional degrees of freedom are possible, such as movement of subsets of panels with respect to the others, or indeed reduced degrees of freedom, such as translations and rotations restricted to known axes, then a parameterisation tailored to that particular system may be necessary. Conveniently, the modular nature of our design means that this bespoke version of a detector parameterisation is simply another subclass of the general model parameterisation abstract base class, and can be dropped in to replace the basic implementation described above.

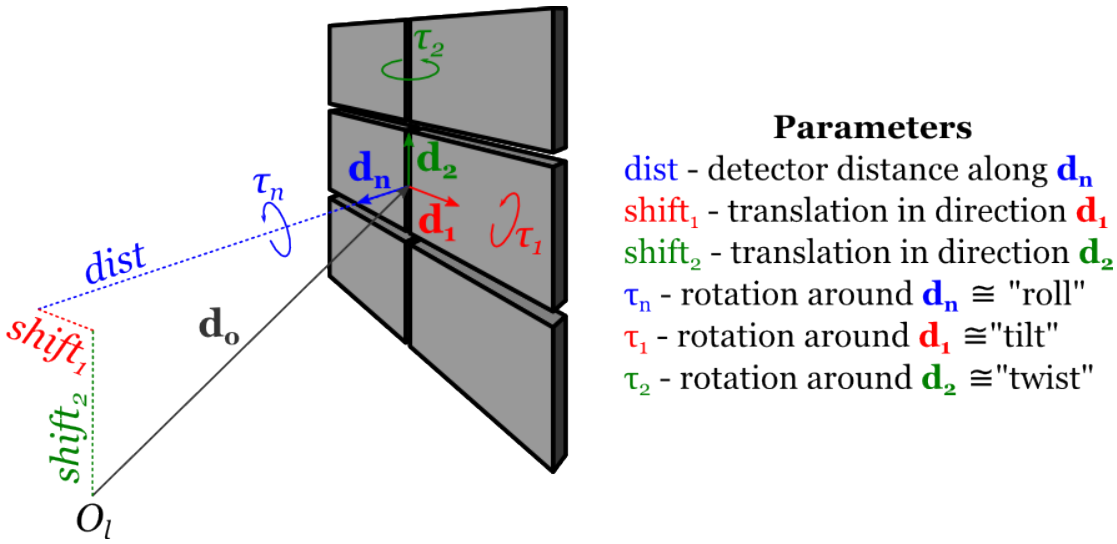

Figure 4. A scheme for generalised parameterization of a detector consisting of one or more panels that are to be treated as a rigid block.

REFERENCES

[1] Bricogne, G. (1986). Proceedings of the EEC Cooperative Workshop on Position-Sensitive Detector Software (Phases I & II), Paris: LURE.

[2] Bricogne, G. (1986). Proceedings of the EEC Cooperative Workshop on Position-Sensitive Detector Software (Phase III), Paris: LURE.

[3] Bricogne, G. (1987). Proceedings of the Daresbury Study Weekend, edited by J. R. Helliwell, P. A. Machin & M. Z. Papiz, Warrington: Daresbury Laboratory.

[4] Grosse-Kunstleve, R. W., Sauter, N. K., Moriarty, N. W., & Adams, P. D. (2002). The Computational Crystallography Toolbox: crystallographic algorithms in a reusable software framework. *Journal of Applied Crystallography , 35* (1), 126 - 136.

[5] Leslie, A. G., & Powell, H. R. (2007). Processing Diffraction Data with Mosflm. In R. J. Read, & J. L. Sussman, *Evolving Methods for Macromolecular Crystallography* (pp. 41 - 51). Dordrecht, The Netherlands: Springer.

[5] Kabsch, W. (2010). XDS. *Acta Crystallographica Section D , 66* (2), 125 - 132.

[7] Schreurs, A. M. M., Xian, X. & Kroon-Batenburg, L. M. J. (2010). *J. Appl. Cryst.* **43**, 70-82.

[8] Hammersley, A. P., Bernstein, H. J., & Westbrook, J. D. (2006). Image dictionary (imgCIF) International Tables for Crystallography: John Wiley & Sons, Ltd.