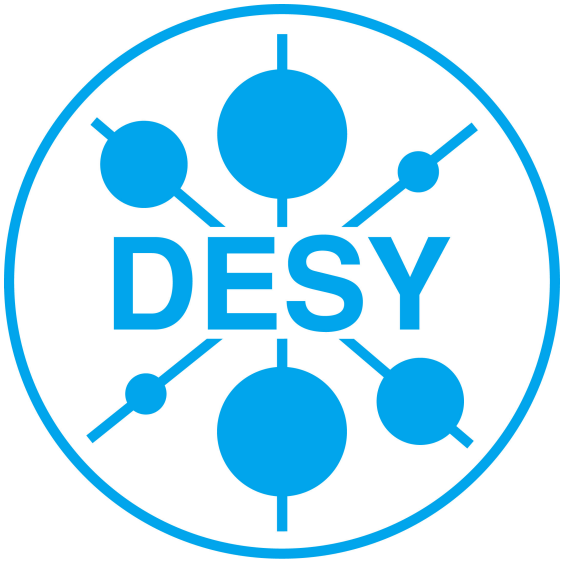

3D merging: getting more information from crystallography data

Marina Galchenkova, Moscow Institute of Physics and Technology (State University), Moscow

September 12, 2017

Abstract

Nowadays the interest to the three-dimensional structure of biological macromolecules is increased. Understanding of their functioning mechanisms is one of the most important fields of modern biology. Atomic resolution structures provide a deep and unique understanding of its function, and help to unravel the inner mechanisms of the living cell. To date, most of the proteins in the Protein Data Bank are macro-molecular structures determined by X-ray crystallography. One of the recent ways to collect information on individual crystals is to use X-ray free-electron lasers. This report describes main steps to obtain experimental datasets on some proteins, such as lysozyme, thaumatin and photosystem II; a brief description of codes that were written to automate and optimise the process of manipulation with datasets is given.

Contents

1 Introduction

Historical outline

1. The discovery of X-rays by Conrad Rntgen, *the first Nobel Prize in Physics in 1901*;

2. Max von Laue was first who observed diffraction of X-rays and revealed the wave nature of X-rays, *the Nobel Prize in Physics in 1914*;

3. The Bragg's (father and son) made the experiments and showed that X-ray diffraction could be used in the determination of the atomic structure of matter, *1913*;

4. Max Perutz and John Kendrew were the first who determined the first protein structure - the structure of myoglobin, *the Chemistry Nobel Prize in 1962*;

5. The Nobel Prize: Dorothy Hodgkin for the structures of vitamin B12 and insulin *(Chemistry Prize of 1964)*; Johann Deisenhofer, Robert Huber and Hartmut Michel for the determination of the structure of the first membrane protein, the photosynthetic reaction center *(Chemistry Prize of 1988)*; John E Walker for his role in the determination of the structure of ATP synthase *(Chemistry Prize of 1997)*;Peter Agre Roderick MacKinnon *(Chemistry Prize of 2003)*, Roger Kornberg *(Chemistry Prize of 2006)*, Venki Ramakrishnan, Thomas A. Steitz, Ada Yonath for the elucidation of the ternary structure of the ribosome *(Chemistry Prize of 2009)*, and recently Brian Kobilka and Robert Lefkowitz for functional and structural studies of GPCR proteins *(Chemistry Prize, 2012)*.

X-ray crystallography

X-ray crystallography can be thought to be one form of very high resolution microscopy. It enables us to visualise protein structures at the atomic level that enhances our understanding of protein function. Specifically we can study how proteins interact with other molecules, how they undergo conformation changes, and how they perform catalysis in the case of enzymes. The use of protein structure information is currently widely spread within many areas of science and industry, among which are biotechnology and pharmaceutical industry.

The reason of using X-rays

In comparison with X-ray crystallography, in all forms of microscopy, the resolution and amount of detail is limited by the wavelength of the electro-magnetic radiation used. To see proteins with atomic resolution one needs to work with electro-magnetic radiation with a wavelength of around 0.1 nm or 1 , in other words the best way is to use X-rays for this goal.

The common part of microscopy and X-ray crystallography is that the principle of experiment is the same on first steps: the subject is irradiated with light and causes the incident radiation to be diffracted in all directions. Bur after that both of this process

are quite different: in light microscopy the diffracted beams are collected, focused and magnified by the lenses in the microscope to give an enlarged image of the object; in electron microscopy - the diffracted beams are focused using magnets. The situation is different in the case of X-ray crystallography: it is not possible to physically focus an X-ray diffraction pattern, so it has to be done mathematically.The diffraction pattern is recorded using some sort of detector which used to be X-ray sensitive film, but nowadays is usually an image plate or a charge-coupled device (CCD).

Why the crystals are needed?

The diffraction from a single molecule is too weak to be measurable. To magnify the signal an ordered three-dimensional array of molecules (crystal) is used. If crystal is imperfect, the X-rays will not be directed to high angles and the data will not consist a detailed structure. On the other hand, in situation with well ordered crystal - diffraction will be measurable at high angles or high resolution and a detailed structure should result. The X-rays are diffracted by the electrons in the structure and the result of an X-ray experiment is a 3-dimensional map of electrons distribution in the structure.

A crystal is a three-dimensional diffraction grating, which gives rise to both constructive and destructive interference effects in the diffraction pattern, such that it appears on the detector as a series of discrete spots which are known as reflections. Each reflection contains information of all atoms in the structure and conversely each atom contributes to the intensity of each reflection.

X-rays have wave properties, as with all forms of electro-magnetic radiation, -they have both an amplitude and a phase. But only the amplitudes can be recorded experimentally - all phase information is lost.This problem is known as "the phase problem". When crystallographers say they have determined a structure, it means that they have solved "the phase problem": they have obtained phase information sufficient to enable an interpretable electron density map to be calculated.

2 Theory

X-ray free-electron lasers advance new avenues of structure determination: it uses intense X-ray pulses of sufficiently short duration to outrun most conventional radiation damage processes.

Result of eliminating proteins with X-ray pulses depends on quality of its crystal:it is necessary to ensure X-ray diffraction extends to have large scattering angles enough and as consequences will give information of high resolution with which can be solved the crystal structure.The limitation of getting higher-resolution Bragg peaks is dictated by a deviation of strict positional ordering of the crystalline lattice and has no great correlations with macromolecules heterogeneity.

As a result of such displacements of molecules from the ideal lattice, continuous diffraction pattern is obtained, in other words, we have the incoherent sum of diffraction from rigid individual molecular complexes that aligned along several discrete crystallographic orientations and as a consequence contains more information than Bragg peaks alone. Such continuous diffraction pattern have not been used for structure determination. Despite the fact that it has been observed for a long time, they raised the interest only for studying of the dynamics of proteins.

Such displacements of molecules from the ideal lattice give rise to a continuous diffraction pattern that is equal to the incoherent sum of diffraction from rigid individual molecular complexes aligned along several discrete crystallographic orientations and that, consequently, contains more information than Bragg peaks alone. Although such continuous diffraction patterns have long been observed–and are of interest as a source of information about the dynamics of proteins–they have not been used for structure determination.

The new method of structure determination is based on using femtosecond pulses from a hard-X-ray free-electron laser, the Linac Coherent Light Source,- and that single-crystal X-ray diffraction 'snapshots' are collected from a fully hydrated stream of nanocrystals. To avoid the problem of radiation damage in crystallography is preferred to use pulses briefer than the timescale of most damage processes. It produces a new strategy in the way of macromolecule's structure determination of crystals either with no sufficient size for studies using conventional radiation sources or particularly sensitive to radiation damage.
In the article (Yefanov et al., 2014) one can find a description of alternative way, called 'three-dimensional merging',- of processing data recorded by serial femtosecond crystallography: authors offer to map the diffracted intensities into three-dimensional reciprocal space instead of integrating each image in two dimensions as in the classical approach. The paper notes the following features of this procedure: it retains information about diffracted intensities between Bragg spots and asymmetry in Bragg peaks. Observed intensity between Bragg peaks and peak asymmetry are potential candidates for direct phasing strategies, while for post-refinement and structure determination can be used

the extracted reflection intensities from the intensity distribution.

3 Measurement

Table 1: Experimental parameters

Sample	Distance	Energy
Lysozyme	200 mm	12 kEv
Lysozyme	300 mm	12 kEv
Lysozyme	500 mm	12 kEv
Thaumatin	200 mm	12 kEv
Thaumatin	300 mm	12 kEv
Thaumatin	500 mm	12 kEv
Photosystem II	200 mm	12 kEv
Photosystem II	300 mm	12 kEv
Photosystem II	500 mm	12 kEv

In these series of experiment were studied these samples: lysozyme, thaumatin and photosystem II. Both of them was observed in different condition – distinctive detector distance, in Figure **??**.

To remind:

Small distance: more reflections; spots might overlap because they are closely spaced

Large distance: less reflections; neighbouring spots are better separated

X-ray Beam Wavelength

The energy of the beam is kept at 12.4 keV, since beam energy E is given by the equation

$$E = \frac{12,4}{\lambda} \tag{1}$$

with E in keV and λ in Å. An energy of 12.4 keV corresponds to a wavelength of approximately 1 Å, which is on the order of inter-atomic bond lengths in most proteins.

Few words about samples:

Lysozyme, in Figure 1, are small globular protein enzymes composed of 129 amino acid residues. Alexander Fleming had shown them to be produced by phagocytes and epithelial cells (Neufeld).They are part of the glycoside hydrolase family for damaging the cell walls of bacterial cells by catalyzing hydrolysis of 1,4-beta-linkages. Lysozymes can be found in tears, human milk, saliva, and mucus, so it acts as part of the body's defence system against bacteria.

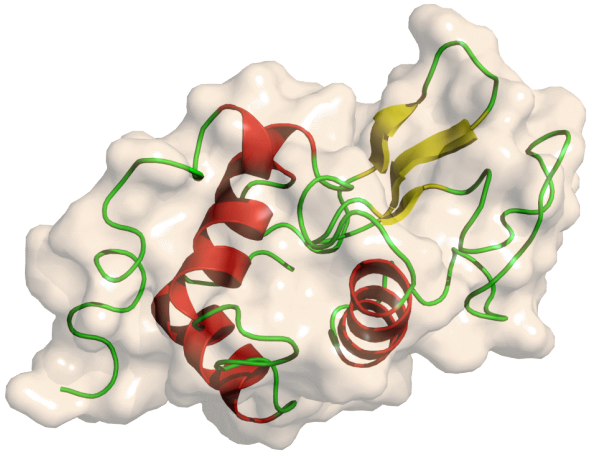

Figure 1: Lysozyme

Thaumatin, in Figure 2, – an intensely sweet-tasting protein, elicits a sweet taste sensation at 50 nM. It is isolated from the katemfe fruit, Thaumatococcus daniellii, also called the miraculous fruit of the Sudan. This is used as a sweetener in cooking, in flavouring palm wine, and in healthy alternatives to sugary treats.

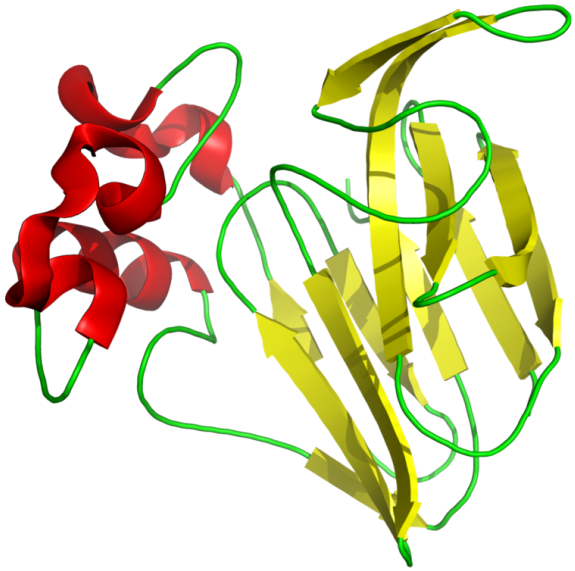

Figure 2: Thaumatin

Photosystem II (or water-plastoquinone oxidoreductase), in Figure 3, is the first multi-sub-unit, pigment-protein complex in the light-dependent reactions of oxygenic photo-

synthesis. It consists of around 30 subunits and several cofactors. It is located in the thylakoid membrane of plants, algae, and cyanobacteria.

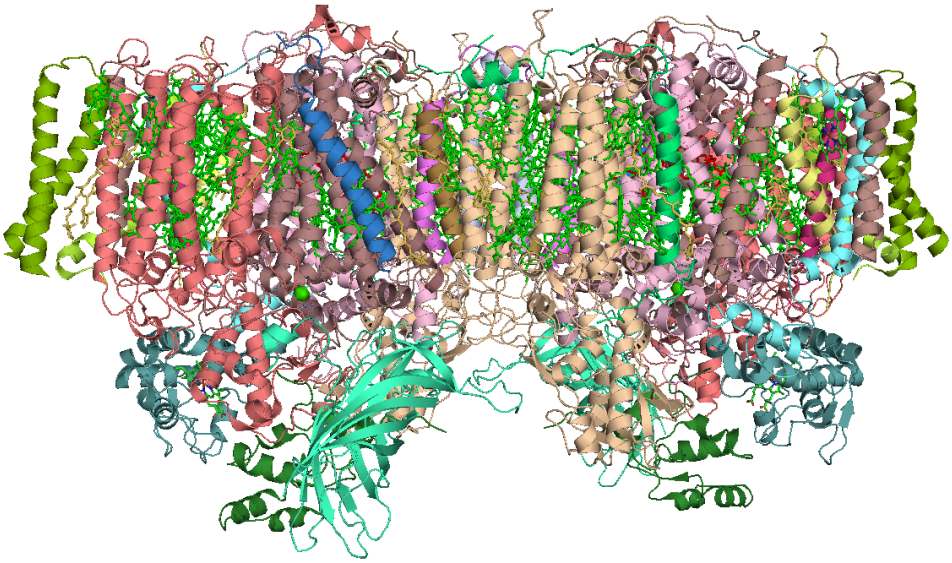

Figure 3: Photosystem II

4 Software

In this part of the report I'll give information about programs that I used and briefly introduce how to use scripts prepared for automation the process of obtaining datasets.

4.1 XDS

X-ray Detector Software (XDS) processes single-crystal diffraction data. Originally written for biological crystallography that most of the times deals with bad data due to ice rings, radiation damages, metastable phases, coexisting phases, poorly crystalline materials, etc. Bad data is exactly what a highpressure crystallographer has to work with. The principle of its work lays on processing a sequence of adjacent, non overlapping rotation images collected from a single-crystal at a fixed X-ray wavelength. Result is recorded by a variety of imaging plate, CCD, pixel and multiwire area detectors. It only requires that incident beam and rotation axis intercept in one point in the center of the crystal and allows arbitrary but fixed orientations of the detector and rotation axis. Output data consist of reflecting range, spot width, crystal orientation, symmetry, and cell parameters from the data images. In correction part of XDS work can be delivered a list of corrected integrated intensities of the reflections occurring in the data images.

The XDS includes three process:

XDS: data indexing, integration, and correction from area-sensitive detectors

XSCALE: combining the data from different runs and correcting them for absorption and radiation damage

XDSCONV: hkl data conversion to different formats.

Advantages:

1. Free software to process the data from several detectors on laboratory diffractometers and at synchrotron facilities.

2. Learned reflection profiles.

3. Gasket rings (or any Debye-Scherrer rings) could be treated like ice rings in the standard use of XDS.

Disadvantages:

1. No reciprocal space reconstruction.

2. It does not handle the disparate phases.

3. Strange "black box" - too automated.

4. Not informative graphics and poor visualisation of the data and results.
For more information, see this article (Kabsch, 2010).

Table 2: Subroutines that was used

Functionality	Comments
XYCORR	performs spatial corrections at each pixel of a detector
INIT	classifies pixels as background or strong (diffraction) spots
COLSPOT	locates strong diffraction spots and finds their centroids
IDXREF	finds and refines the orientation matrix
DEFPIX	defines the obscured regions of the detector by intruding hardware, e.g., a cryostat (not useful for masking the shadowed areas of the detector by a diamond anvil cell)
INTEGRATE	determines the intensities
CORRECT	applies various corrections to the intensities, determines the space group if unknown, and refines the unit-cell parameters

Table 3: Important output files

Name of file	Comments
IDXREF.LP	the results and data diagnostics of IDXREF
XPARM.XDS	the initial orientation matrix and parameters determined by IDXREF subsequently used in INTEGRATE
INTEGRATE.LP	the results and data diagnostics of INTEGRATE
GXPARM.XDS	the final orientation matrix and parameters determined by CORRECT
CORRECT.LP	the results and data diagnostics of CORRECT
XDS_ASCII.HKL	the final hkl data processed by CORRECT

4.2 CrystFEL

CrystFEL is a suitable software that allow to solve the specific needs of the emerging technique of serial femtosecond crystallography, where structural information is got from small crystals that was illuminated by an X-ray free-electron laser. It deals with indexing, integrating, merging, viewing and evaluating the quality of the data, and also simulating patterns. In accordance with the need to accelerate the process of indexing and integration of a large number of diffraction patterns and to translate these processes into automatic mode the software uses multi-core hardware. Other features of this software package is the merging and scaling of a large number of intensities from randomly oriented snapshot diffraction patterns. A great hallmark of software is the use of a generalised representation of the detector to facilitate the use of more complex geometries available in conventional crystallography. More details can be found in this paper (White et al., 2012).

4.3 xds_autostart

xds_autostart – is a bash script that allow in real-time without any interruption to execute xds_par for the folder contained all patterns for the one type of sample on different outer conditions (for instance, different detector distance). After running code it calls the Python script, intended to fill in the pilatus template of XDS.INP that is necessary for running XDS.

The main algorithm includes the following steps, in Figure 4:

1. make a new directory associated with folders from datasets place
2. copy the appropriate template and rename it XDS.INP
3. put the correct parameters in XDS.INP and uncomment 'JOB =' with full parameters
4. execute xds
5. execute script reading_GXPARM and/or reading_file_LP for checking output parameters and I/sigma for the sample.
6. uncomment another 'JOB =' and execute xds_autostart again

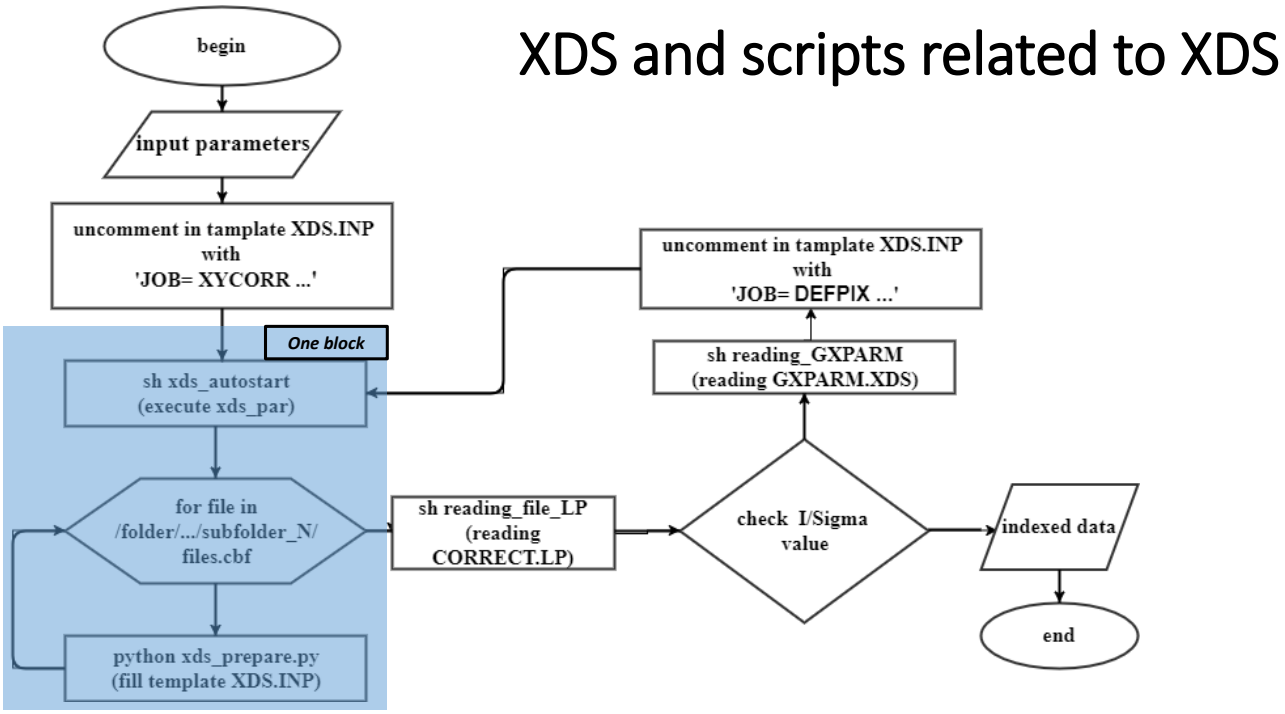

Figure 4: XDS and scripts related to XDS

On next page you can find the description of all scripts and their input arguments with brief explanation the main idea of its executing.

Table 4: Brief description of input and output parameters of scripts

Name of execute file	**Input parameters**	**Output result**
xds_autostart	*input_dir*=the full path for dataset *output_dir*=the full path to the folder that will contain the result of xds work *XDS_TPL*=template which has a substitution of values keyword occurs in the execution of Python script xds_prepare.py	the folders and sub-folders that contains the final files of XDS work
xds_prepare.py	*XDS_TPL*=template that it fill with using 'from pilatus_parser import PilatusHeader' *file*=pattern of full path for all files that have this format *.cbf	Full template XDS.INP that used in xds_par
reading_GXPARM	*input_dir*=the full path for the folder contains GXPARM file	file includes space group number and unit cell parameters and the origin of the detector coordinate system with respect to the laboratory system
reading_file_LP	*input_dir*=the full path for the folder contains file with format *LP	file contains the full path for this *.LP file and the value of I/sigma.

Listing 1: How to execute xds_autostart

```
sh xds_autostart /gpfs/cfel/cxi/scratch/data/2017/PETRA-2017
-Ayyer-Jun-P11/11003107/raw/cryo /gpfs/cfel/cxi/scratch/user/
yefanov/students/galchenm/xds/lys ~/XDS_lys.INP
```

Listing 2: How to execute reading_GXPARM

```
sh reading_GXPARM /gpfs/cfel/cxi/scratch/user/
yefanov/students/galchenm/xds/lys
```

Listing 3: How to execute reading_file_LP

```
sh reading_file_LP.sh /gpfs/cfel/cxi/scratch/user/yefanov/
students/galchenm/xds/lys
```

4.4 last_prepare_for_turbo

last_prepare_for_turbo – is bash script that allows to run *turbo-index* that call *indexamajig* in automatic mode.

Input arguments:

input_file – a list of full-path to *.cbf for every sample that is prepared before with get_list_for_turbo_index;
output_dir – output directory, where will be saved the results of executing turb-index;
PDB – file of cell-parameters;

Comments: this bash script automate the process of executing turbo-index and the result we have the folders of indexed data.

Listing 4: How to execute last_prepare_for_turbo

```
sh last_prepare_for_turbo /home/galchenm/bin/
folder_of_list_for_turbo_index/lys_index.lst
/gpfs/cfel/cxi/scratch/user/yefanov/students
/galchenm/crystfel170617 /gpfs/cfel/cxi/scratch/
user/yefanov/students/galchenm/crystfel/lys.cell
```

4.5 Merging

Merging process can be described in this way, in Figure 5:
merge_sample → *merge_symm_sample* → *merge_sub_3d_sample* → *merge_correlation_sample* → *merge_sum3d_sample*

Listing 5: How to execute merge_*

```
sh merge_lys /gpfs/cfel/cxi/scratch/data/2017/
PETRA-2017-Ayyer-Jun-P11/11003107/raw/cryo
/gpfs/cfel/cxi/scratch/user/yefanov/students
/galchenm/xds180817/lys
```

Listing 6: How to execute merge_sym_*,merge_symm_*,merge_sub_3d_*, merge_sum3d_*

```
# For instance, how to execute
# merge_symm_lys
sh merge_symm_lys 501
```

Main part for different merge-scripts

Some of merge-scripts have common parts of algorithm. In the following boxes these

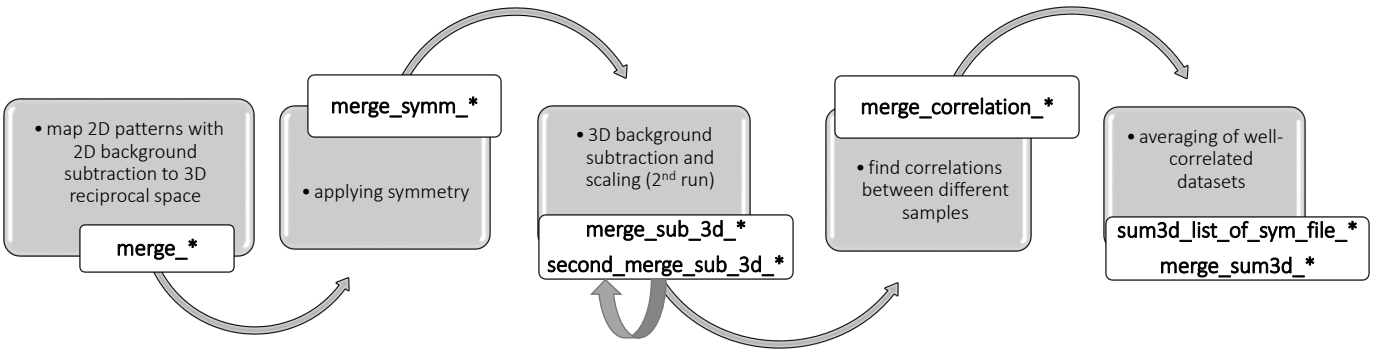

Figure 5: Merging

parts are shown and a brief explanation of each stage of their implementation is given.

Listing 7: Splitting full path into "level-name"

```
# Find all files satisfying the pattern
for file in 'ls level1/level2/level3/filename.ext'; do
  cbf=$(readlink -e $file)
  cbf_fn=$(basename $cbf)     # filename.ext
  cbf_path=$(dirname $cbf)    # /level1/level2/level3
  cbf_path_p2=$(basename $cbf_path) # level3
  cbf_path_p1=$(basename $(dirname $cbf_path)) # level2
```

In this part you can find the idea of splitting the full–path into its different parts. So if you run $cbf\_path\_p3 = \$(dirname\ \$(dirname\ \$cbf\_path))$,where $cbf = /level1/level2/level3/filename.txt$, the result will be $/level1$.

Listing 8: The part of finding satisfying geometry file

```
# Extract detector distance and find GEOM for the following
```

Table 5: Brief description of bash scripts

Name of execute file	**Input config–file**	**The result of work**
merge_*sample*	conf_rawor	mapping the continuous reciprocal space intensity distribution
merge_symm_*sample*	conf_symm	applying symmetry
merge_sub_3d_*sample*	conf_sub_3d	background subtraction
second_merge_sub_3d_*sample*	conf_sub_3d	scaling
merge_correlation_*sample*	conf_orient	find correlations
merge_sum3d_*sample*	conf_sum3d	averaging
merge_convert_*sample*	conf_convert	check the satisfaction of the application mask

```
# applying

det_dist=`awk -F "␣" '$0~"#␣Detector_dist" {print $3;}' f.cbf`
#get detector distance from file *.cbf

if [[ $det_dist == *"0.50000"* ]]; then #[[ det_dist -eq 500 ]]
    d=`echo 500` #it will use in determination MASK
    GEOM="pilatus6M_500mm_cen.geom" #get necessary mask
elif [...]
```

In this listening *awk* allow us to find in file *f.cbf* the string that starts with words *Detector_dist* and extract the third column that contains detector distance.

Listing 9: Associate, for instance, necessary mask with pattern from one block and having fixed detector distance

```
# Check the existance of block-file and after that
if [ -e block.txt ] && (grep -Fxq "pattern" block.txt); then
        MASK="mask_${d}mm1.h5"
elif [ ... ]
```

On the stage of preparation for the next coding, first af all, the full analysis was made according experimental datasets: samples were divided into blocks corresponding to different external conditions (stop beam-line and so on). Some of samples can be found in every blocks, some – not. To write code suitable for all processing situations was decided

to check does exist the file txt-format of necessary block in folders and if so, then verify whether the block contains the file of interest.

Listing 10: Avoiding problems after impossible indexing with XDS

```
# sigma - is a criterion according to which only those patterns
# whose sigma value exceeds some limit will be selected.
# If the pattern wasn't obtained with XDS, the folder didn't
# contain file CORRECT.LP and this file was written in list
# no_LP_file.txt.
sigma=`awk 'BEGIN {a = -1;} $0 ~ "RESOLUTION RANGE I/Sigma "
{ a = NR;} (NF>3) && (NR>a) && (a>=0)
{printf "%s ", $3; exit 0;}' /full_path/CORRECT.LP
echo -e file >>/output_dir/no_LP_file.txt `
if [[ "$sigma" > "10.00" ]]; then
...
```

The same idea as was shown in above listening with using *awk*: extract I/sigma and after that process only that patterns which this parameter is grater that limit value chosen for every patterns itself.

The table below shows all executing files, their input parameters and comments of their implementation.

Table 6: Brief description of input and output parameters of script merge*

Config file	Input parameters	Comments
Brief description of input and output parameters of script merge_*		
conf_rawor	*input_dir*= the full path of dataset *output_dir*=output directory *xdsfolder*=the path to CORRECT.LP	It creates associated folders like in *input_dir*. Get distance for applying necessary geometry file. Obtain only datasets where oversigma is greater than limited value (it is used for each sample). Full config with File, Mask and GEOM. Run C-code intor with conf_rawor.
Brief description of input and output parameters of script merge_symm_*		
conf_symm	*input_dir*= the full path of dataset after executing C-code intor with conf_rawor *output_dir*=output directory *num*=counts of dots	Find files suitable for one of patterns: files_???pat_num or files_????pat_num. Full config with File pattern. Run C-code intor with conf_symm.
Brief description of input and output parameters of script merge_sub_3d_*		
conf_sub_3d	*input_dir*= the full path of dataset after executing C-code intor with conf_symm *output_dir*=output directory *num*=counts of dots	Find files suitable for one of patterns: files_???pat_num_sym or files_????pat_num_sym. Full config with File pattern. Run C-code intor with conf_sub_3d.
Brief description of input and output parameters of script second_merge_sub_3d_*		
conf_sub_3d	Inside the script, the path to the stream file is registered, containing the result of the C code execution in the previous stage, the pattern itself and the path to the folder	Run C-code intor with conf_sub_3d, where *MaxScaleCoef* was added.
Brief description of input and output parameters of merge_correlation_*		
config_orient	*input_dir*= the full path of dataset after executing C-code intor with conf_sub_3d *output_dir*=output directory *num*=counts of dots	Find files satisfies the patterns: files_???pat_num_sym or files_????pat_num_sym. Find another files:files_???pat_num_sym_cleaned3D or files_????pat_num_sym_cleaned3D.Fill config_orient with File and SomeotherFile (associate with patterns). Run C-code intor with config_orient
Brief description of input and output parameters of merge_sum3d_*		
config_sum3d	—-	Extract *list* from *sum3d_list_of_sym_file_lys* and run ../../intor with config_sum3d
Brief description of input and output parameters of merge_convert_*		
conf_convert	*input_dir*= the full path of necessary dataset *output_dir*=output directory *num*=counts of dots	Fill *list* of files with . convert_list_of_files_*Full config file with MASK and GEOM.Run C-code intor with conf_convert.

Table 7: Brief description of bash script second_merge_sub_3d_*

Config file	Input arguments	The result of
conf_sub_3d	*input_file*= the result of merge_sub_3d_* work *limit*=the limit value of MaxScaleCoef: if it is greater than limit value, make scaling	scaling patterns

4.6 find_tif

find_tif – is a bash script that allow to collect necessary slices for each samples in one folder and generating for every images the name correlates with the last folder where it contains.

Listing 11: How to execute find_tif

```
sh find_tif /gpfs/cfel/cxi/scratch/user/yefanov/students
/galchenm/merge  501 xy
```

Table 8: Brief description of bash script find_tif

Input arguments	The result of work
input_dir= the full path of folder contains slices *.tif *num*=counts of dots *slice*=shear projection	Go through all the folders and find the files that satisfy the template *_slice_$slice.tif and copy them into a folder that is two levels higher.

4.7 Python script. C–extension

First of all, we should understand the differences between Python and C-language:

Table 9: Python vs C-language

Python	**C-language**
Python type discipline is duct, dynamic and strong	C language type discipline is static and weak
Python programming is slower compared to C, but coding is easier and short.	C programming is faster, but coding is complex and too long
Python does not require type declarations.	C requires type declarations.
Python programming requires indentation.	Indentation is not a necessity in C language.
Python does not use pointers instead uses associative arrays and sequences.	C makes use of the pointers.
Python supports automatic memory management with reference counting.	C supports manual memory management with library tools malloc and free.

The table had been marked only the differences in the languages which have played a significant role in the writing of the code,or the differences can be presented as in Figure 6.

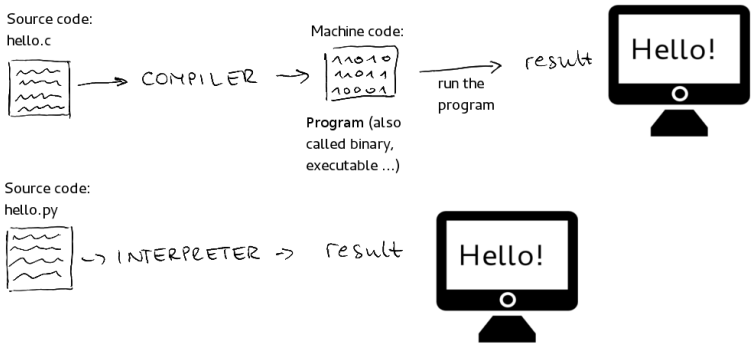

Figure 6: Python vs C

At this stage a script was written in Python 3.*. Common understanding of its work is

represent in the following algorithm:

Listing 12: The general idea of the program

```
# Main steps
1: get coordinates from geometry file: x_array,y_array,z_array
2: read stream and collect it to matrix:metadata=[[name_of_file,
                                        photon_energy_eV,
                                        average_camera_length,
                                        cell,[astar,bstar,cstar]]]
3: get patterns from HDF5 format file: Int
4: subtraction of the background: Int=Int-background
5: determine && following applying mask: mask=MASK_GOOD/MASK_BAD
6: build Ewald sphere: binary file of intensity
7: build Ewald sphere with rotation: binary file of intensity
```

Input parameter:
- geometry file: geom_file
- stream file: stream
- number of points: N_3D
- resolution: resolution
- detector distance: dist

Most of them for simplifying testing code was prescribed in the code.

There were some functions that were implemented in C-language. And to speed up executing Python code was made a decision to expand Python code with C-functions. So, during building code the problem of extension with C-code was faced. For this purpose *numpy as np* and *ctypes as ct* libraries were chosen. And the recipe how to solve this issue can be found on the following code-boxes:

Listing 13: Function that return pointer to the structure

```
def _np_ptr(np_array):
    return ct.c_void_p(np_array.ctypes.data)
```

In the next listening the main idea of calling C-function is shown:

Listing 14: Common part of C-extension in Python code

```
def C_func_extension_py(first_arg, second_arg,
                        third_arg,len_third_arg):
    # accessing functions
    # from loaded dlls
    lib = ct.CDLL('C_library.so')

    # memory allocation
```

```
    third_arg = np.array([0]*len_third_arg, dtype=np.float32)

    # calling function
    pfun = lib.FunctionOfInterest

    # return types (here integer)
    pfun.restype = ct.c_int

    # specifying the
    # required argument types
    pfun.argtypes = (ct.c_int, ct.c_float, ct.c_void_p)

    '''
    The previous lines mean in C-code that:
        int FunctionOfInterest(int first_arg,
                               float second_arg,
                               int* third_arg)

    And inside this function, third_arg is populated.
    '''
    flag = pfun(first_arg, second_arg,_np_ptr(third_arg))
    return flag, third_arg
```

One another problem that raised was when in C-function declared the structure.

Listing 15: Declaring C-structure in Python

```
class PythonVersionStructure(ct.Structure):
    '''
    typedef struct {
    public:
        long        first_param;
        float       second_param;
        int         third_param;
        float       *fourth_param;

    } tSomeList;
    '''


    _fields_=[('first_param',ct.c_long),
              ('second_param',ct.c_float),
              ('third_param',ct.c_int),
              ('fourth_param',ct.POINTER(ct.c_float)))]
...
def SomeOtherFunction(...):
```

```
...
req = PythonVersionStructure() # use created structure
...
```

In previous scripts-listing I tried to show how in general way to cope with challenge that can be faced during making extension Python–code with C–functions using Python library ctypes in Figure 7. .

Figure 7: Python extension with C

The following table provides a list of functions that have been implemented in the main Python program:

Table 10: Function in latest_version_main.py

Name function	**Comments**
PeakFinder8py	return mask
background	subtracts the radially symmetric background
pSubtractBgLoop	local background subtraction
pBuildRadiulArray	calculates to which radial neck each pixel belongs for background
hdf5_work	apply geometry file
NEW_sphere_Evald	create Ewald sphere
NEW_sphere_Evald_with_rotation	create Ewald sphere with rotation
cell_crystallographic_to_cartesian	are built on cell parameters
create_mat_cur	constructed on the parameters of the cell in the reciprocal space
get_opt_patterns	create dataset *metadata*
processing	the processing function is called in the process-vendors for each occurrence of data
init_worker	the initialisation function for each process-handler

Combining different methods (includes thread paralleling) in Python-coding the latest version (but not also final) achieved the result of obtaining data that it took 2000 seconds for processing stream that contains near 700 patterns with involving all functions described above.

5 Result

The results of this work are:

1. Became familiar with the concept of protein crystallography (in particular Serial Crystallography);

2. Was acquainted with contemporary work in the field of protein crystallography and data processing methods;

3. Discussed the differences between classical approach of determining molecules structure with new method of avoiding integrating intensities between Bragg's peaks;

Looked through the way of processing data with using XDS;

4. Dived into program CrystFEL;

5. During the development of programs that allow to automate and optimise the launch of external programs such as XDS and CrystFEL, general data processing techniques have been developed and implemented in languages such as bash and Python;

6. Written my own code that can in trivial way figuring out experimental data: subtracting different "type" of background;generate and apply mask;apply geometry file of detector; build Ewald sphere and rotate it;

7. Different challenges were faced and coped with different programming languages as ctypes, multiprocessing and numpy.

The work was carried out in CFEL, in Figure 8. .

Figure 8: CFEL

References

[1] Mapping the continuous reciprocal space intensity distribution of X-ray serial crystallography. *Yefanov, O., Gati, C., Bourenkov, G., Kirian, R. A., White, T. A., Spence, J. C. H., Barty, A. (2014).*

[2] XDS. Acta Crystallographica Section D: Biological Crystallography. *Kabsch, W. (2010).*

[3] CrystFEL: A software suite for snapshot serial crystallography. *White, T. A., Kirian, R. A., Martin, A. V., Aquila, A., Nass, K., Barty, A., Chapman, H. N. (2012)*

[4] Fundamentals of Crystallography. *C. Giacovazzo, H.L. Monaco, D. Viterbo, G. Gilli, G. Zanotti, M. C. (n.d.), (1992)*