# A New Metric for Assessing Anomalous Data Quality in Sulphur SAD experiments

Dissertation
zur Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultät der
Christian-Albrechts-Universität zu Kiel

vorgelegt von
Selina Lea Sophie Storm
Kiel, 2016

| | |
|---|---|
| Erster Gutachter: | Prof. Dr. Martin Müller |
| Zweite/r Gutachter/in: | Dr. Gwyndaf Evans |
| | |
| Tag der mündlichen Prüfung: | 1.11.2016 |
| Zur Druck genehmigt: | 12.12.2016 |

gez. Prof. Dr. Natascha Oppelt, Dekanin

# Abstract

In this thesis, a metric for assessing the anomalous data quality based on the sulphur signal in single wavelength anomalous diffraction (SAD) experiments as applied to protein crystallography is presented. For doing so, SAD experiments were performed on five thaumatin crystals with a top-hat beam profile at an energy of 8 keV on the beamline P14 of PETRA III. To investigate the influence of radiation damage on the data quality, crystals smaller than the beam were chosen to enable even illumination. Data were collected with high multiplicity in set-ups with and without compound refractive lenses (CRL).

The data were processed with XDS and the data quality was analysed with an emphasis on the influence of radiation damage quantified with the program RADDOSE-3D, but also regarding other factors imparing the data quality, such as instrumental errors. It could be shown that the data are nearly identical within experimental error. The substructure of both 360° wedges and accumulated data were processed with SHELXD. The best substructure dependent on the amount of data as well as the substructure quality were analysed with SITCOM based on the known solution of the structure. Specific damage on the sulphur sites was investigated with the help of the program ANODE.

As this method for finding the balance between multiplicity and radiation damage implies the knowledge of the protein structure, a program was developed in Python to calculate the average signed anomalous differences $\langle \Delta F \rangle$ for four different space groups based on the intensities of the reflections alone. Based on the mostly normally distributed $\langle \Delta F \rangle$ values, a metric predicated on the widths of the distributions of the data collected in wedges was developed. The ideal amount of data determined with this metric is not only in good agreement with the results based on the known structure. The metric appears also to be sensitive for processes and changes within different resolution shells. A possible further development of this metric and its potential use in the future is discussed.

# Zusammenfassung

In dieser Thesis wurde eine Metrik zur Charakterisierung der Datenqualität entwickelt, die auf dem anormalen Signal von Schwefel in Proteinkristallographie-Experimenten mit einer Wellenlänge (SAD) basiert. Hierfür wurden SAD-Experimente mit fünf Thaumatinkristallen unter der Verwendung eines kastenförmigen Strahlprofils und einer Energie von 8 keV an der Beamline P14 (PETRA III) durchgeführt. Um den Einfluss von Strahlenschaden auf die Datenqualität zu untersuchen wurden Kristalle ausgewählt, die kleiner waren als der Strahl und somit einer gleichmäßigen Bestrahlung ausgesetzt waren. Die Daten wurden mit hoher Multiplizität in Aufbauten mit und ohne Röntgenlinsen verwendet.
Die Daten wurden mit XDS prozessiert und die Datenqualität analysiert, wobei der Einfluss von Strahlenschaden, der mit dem Programm RADDOSE-3D quantifiziert wurde, im Mittelpunkt stand. Ferner wurden auch andere Faktoren wie der instrumentelle Fehler betrachtet, die ebenfalls die Datenqualität beeinträchtigen. Es konnte gezeigt werden, dass die Daten unter Berücksichtigung des experimentellen Fehlers annähernd identisch sind. Die Substruktur der in 360° Umdrehungen gemessenen als auch die der akkumulierten Daten wurden mit SHELXD prozessiert. Die beste Substruktur in Abhängigkeit von der eingehenden Datenmenge als auch die Substrukturqualität wurden mit dem Programm SITCOM auf Grundlage der bekannten Strukturlösung untersucht.
Um das Gleichgewicht zwischen Strahlenschaden und Multiplizität zu finden setzt die soeben beschriebene Methode also die Lösung der Struktur voraus. Deshalb wurde ein Programm in Python entwickelt, um die mittleren anormalend Differenzen $\langle \Delta F \rangle$ für vier verschiedene Raumgruppen ausschließlich basierend auf den Reflexintensitäten zu berechnen. Ausgehend von den zumeist normalverteilten $\langle \Delta F \rangle$ Werten wurde eine Metrik entwickelt, die auf den Verteilungsbreiten der in 360° Umdrehungen gemessenen Daten beruht. Die so bestimmte, ideale Datenmenge weist nicht nur eine gute Übereinstimmung mit den Ergebnissen auf, die zuvor mithilfe der Lösung der Struktur bestimmt wurden. Ferner scheint die Metrik auch empfindlich gegenüber Prozessen und Veränderungen in unterschiedlichen Auflösungsbereichen. Eine mögliche Weiterentwicklung und potentielle Nutzungsmöglichkeiten werden diskutiert.

# Contents

# Chapter 1

# Introduction

At the beginning of the last century the discovery of X-rays by Conrad Röntgen opened up a fundamentally new possibility to investigate matter. Shortly after the Rutherford model of the atom was formulated, the work of Max von Laue and father and son Bragg revealed that the wave nature of X-rays can be used for the determination of the atomic structure of matter [14]. While the atomic structure of small inorganic compounds (such as NaCl) and organic molecules (such as benzene) were resolved soon, it was more than 40 years for the first protein structure myoglobin to be resolved by John Kendrew [63]. The reasons for this were manifold: the purification and crystallization of proteins as a method was first developed in the late twenties, the scattering of the mainly light atoms in proteins was very weak, and due to the huge number of atoms in a protein molecule, the calculations required to reconstruct the three-dimensional structure went beyond the computational capacities at that time.

Built of 20 α-amino acids and mainly only of five atom species - hydrogen, oxygen, carbon, nitrogen and sulphur -, proteins are together with the nucleic acids the basis for life. Despite their limited number of components, proteins are amazingly versatile. They are for instance responsible for the chemical reactions that replicate our genes, they transmit signals in the body and detect and kill foreign invaders [95]. To understand the function and interaction of these molecular machines it is necessary to know their three-dimensional structure at (near)-atomic resolution. While solution nuclear magnetic resonance spectroscopy (NMR) is useful for small proteins and electron microscopy (EM) for large proteins, X-ray crystallography can provide molecular structure models of both small components, as used in structure guided drug design, and large molecular complexes, as for example evidenced in the nearly 2 MDa structure of the $50S$ ribosomal subunit [7]. Thus it has also become the most popular method for structure deter-

mination in structural biology, so that 90% of all structures in the protein data bank (PDB) have so far been determined by means of X-ray crystallography [86].
The success of macromolecular X-ray crystallography is strongly related to the use and the development of synchrotron radiation sources. Mainly on account of their high photon flux and the possibility to exploit their continuous spectral distribution, synchrotrons became interesting for the crystallographic community. While first-generation facilities were still 'parasitically used' products of particle physics, the second generation of facilities were already dedicated to the production of synchrotron radiation. The nowadays used third generation synchrotrons make use of undulators, resulting in more and more brilliant beams [48]. These brilliant beams are needed to push the limits of X-ray crystallography: to collect data of micron-sized crystals, to deal with the large unit cells of virus and multi-protein complexes, to achieve high-throughput and to use X-rays in the low-energy range for phasing [45, 70].

## The phase problem

If diffracting crystals are available and data can be collected, the crucial step is to solve the phase problem. If a structure with an adequate similarity is available and the crystals are diffracting to a sufficiently high resolution, molecular replacement can be performed. For *de novo* phasing, experimental phasing is the method with which the phases can be determined. The latter is based on the inelastic interactions modelled with damped harmonic oscillations which arise when the incident X-ray energies match the energy of an electron shell of a certain atomic species. This is referred to as anomalous scattering. While anomalous scattering was first exploited in combination with isomorphous replacement for solving protein structures, multi-wavelength anomalous diffraction (MAD) phasing was established as a successful stand-alone method when X-rays with tunable energy became available at synchrotrons [49, 96]. Nowadays, single-wavelength anomalous diffraction (SAD) is the method of choice for performing *de novo* phasing, as it is experimentally less challenging [105]. However, as the anomalous differences obtained from SAD experiments are based on only two measurements instead of four and more in MAD experiments [29], the accuracy of these measurements is crucial. An increased accuracy can be reached if the measurements are repeated (high multiplicity).

## Radiation damage

The achievement of high multiplicity is hindered by one effect: radiation damage. Even though only 2% of the primary X-ray beam interact with the sample, the majority of these X-ray photons interacts with the electrons of a protein via the photoelectric and the Auger effect, resulting in an electron cascades within the sample. Additionally, inelastic scattering takes place, contributing to the background. At an energy of 12.4 keV, only 8% of the interacting X-ray photons contribute to the diffraction pattern via elastic scattering, while the inelastic scattering accounts for another 8% [82]. The direct interaction of X-rays with the electrons of a protein via the photoelectric, Auger and Compton effects is known as primary radiation damage, the reaction of the resulting radiolytic products as secondary radiation damage [97]. The consequences are observable in both reciprocal space (global damage), for example by the fading of high resolution reflections, and in real space (specific damage) [56].
Global radiation damage leads to a steady increase of inconsistencies, as the order in the crystal is gradually destroyed [25]. This is connected to the amount of absorbed energy per mass, i.e. the dose. However, as the bonds holding a protein crystal together differ individually, it is impossible to predict an overall tolerable dose limit for all protein crystals. Thus, different metrics have been developed and used to characterize global radiation damage [37], e.g. :

- the increase of the unit cell volume and mosaicity (compare Fig. 1.1 a) [72, 84],
- the isotropic B-factor $B_{rel}$, which can be interpreted as proportional to the change in the mean-squared atomic displacements (compare Fig. 1.1 b) [64].
- the pairwise comparison of symmetry-related reflection recorded on diffraction images at different doses ($R_d$, compare Fig. 1.1 c) [25],
- the ratio of the summed mean intensity exposed to a certain dose $I_d$ to a summed mean intensity of a first data set (compare Fig. 1.1 d) [13],

Specific radiation damage often occurs long before global damage becomes visible. After the reduction of metal centres, followed by elongation and scission of disulphide bonds, aspartates and glutamates are decarboxylated [41], which can all hamper the biological interpretation of structures.

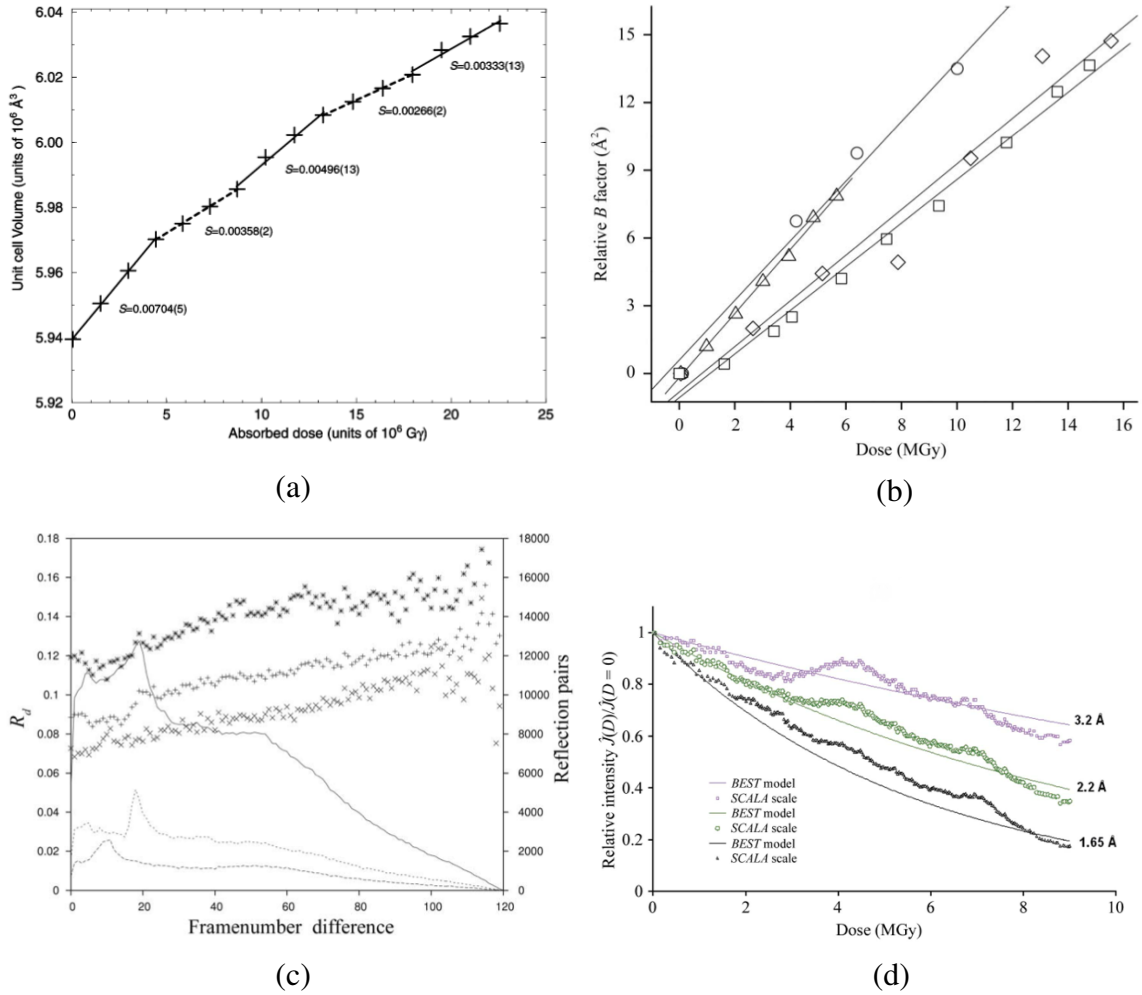



Figure 1.1: Metrics to characterize radiation damage: a) increase of the unit cell volume of a ferritin crystal with dose, taken from [84]; b) change of the relative B-factor versus dose for crystals of lysozyme (squares), catalase (diamonds), thaumatin (triangles) and apoferritin (circles), taken from [64]; c) R factor as a function of frame number difference ($R_d$) without radiation damage correction performed in XSCALE, where "+" stands for all data, "x" for data in the resolution range of 10 – 5 Å and "*" for the one in the 3.5$\hat{a}$\3 Å. The lines show the number of contributing reflection pairs, where the solid line represents all data, the long dashes the 10 – 5 Å resolution shell and the short dashes the ones in the 3.5$\hat{a}$\3 Å resolution shell. Taken from [25]. Fig. 1.1 d) shows the predicted and experimental relative ï¿diffraction intensities for cubic insulin crystals (taken from [13]).

## Sulphur SAD phasing

Light elements such as carbon, nitrogen and oxygen have negligible anomalous scattering at the energies suitable for performing X-ray diffraction experiments, and heavy atoms only occur in some proteins. The only light element which can give rise to usable anomalous scattering at X-ray energies used for protein crystallography is sulphur, which is found in the amino acids methionine and cystein. This is an advantage in comparison to the option of introducing heavier elements into the crystal, which often influences the crystal quality negatively.

Although Teeter et al. [53] proved that the anomalous signal of sulphur can be used to solve small protein structures as early as 1981, the technique only became widely accessible from 1999 on due to improvements in data collection and statistical phasing methods, which was firstly demonstrated by Dauter et al. [22]. To use the anomalous signal of sulphur, certain conditions have to be fulfilled. First of all, it is necessary that a certain ratio of the protein building amino acids are cysteins or methionines. Secondly, the anomalous signal from the sulphur atoms has to exceed a certain signal to noise strength, which can be enhanced by lowering the X-ray energy. However, as the sulphur K-edge is located at an energy of 2.47 keV which is too low to be used for crystallography, MAD phasing is not an option. Thirdly, the data collection has to be performed very carefully, as the anomalous signal from sulphur is generally very weak. To measure the anomalous signal with high precision, each reflection or its symmetry equivalent should be measured as often as possible (high multiplicity). Besides, the anomalous signal should be measured accurately by reducing the systematic error.

Recently, there has been further methodological progress in sulphur SAD phasing. One approach pursues the merging of multiple crystals to obtain high multiplicity without severe radiation damage [68, 69]. Others are trying to perform the measurements at energies closer to the sulphur absorption edge, facing the problem of stronger absorption and larger diffraction angles at longer wavelengths [17, 103]. Furthermore, the approach of collecting high multiplicity data sets with small systematic error by exposing the crystal in various orientations to a low dose turned out to be very successful [75, 105]. Despite this progress in the field of sulphur SAD phasing, it is not possible to monitor specific radiation damage during the experiment [37] and it is still hard to tell whether adding data would lead to an improvement or to worsening of the substructure on account of radiation damage, before the structure is refined.

## Outline of the thesis

Within this thesis, an attempt was made to develop a metric for finding the balance between radiation damage and multiplicity. For doing so, the low emittance of PETRA III was exploited to generate a top-hat beam. By using crystals smaller than the beam and the excellent set-up at P14 the conditions were close to the ones of a perfect experiment. As a test system, thaumatin was chosen, as it is well investigated. Containing eight disulphide bonds and an additional single sulphur atom, it provides a sufficiently high anomalous signal at the chosen energy of 8 keV. Knowing the result, a metric was developed which enables to find the balance between radiation damage and multiplicity for getting the best substructure, based on the measured reflections only. This balance is verified by subsequently adding more data and conventionally solving the (sub)structure.

The present thesis starts with a chapter explaining the physics and the used software with a focus on the quantities required in the following. The experimental set-up and the materials and methods used are shortly described in the correspondingly named chapter. The obtained results are presented and discussed within two chapters. The first part of the results deals with a detailed analysis of the measured data, including the solution of the substructure and a comparison to a refined structure. The second part introduces the developed metric and its validity is confirmed based on the results analysed in the previous chapter. A summation and an outlook can be found in the last chapter, giving ideas how the metric might be further tested and applied in the future. The corresponding scripts and programs to use this metric can be found in the appendix, as well as additional plots.

# Chapter 2

# Theory and Background

Protein structures can be solved by performing X-ray diffraction experiments. These experiments require the protein to be in a crystalline form. Assuming an ideal diffraction experiment, it is first explained how X-rays interact with protein molecules in an ordered lattice. The symmetry of the lattice introduces certain relations of reflections which can be used for structure determination. Reflections which are related by an inversion centre have the same intensity in absence of anomalous scattering. This relation is known as Friedel's law, and its break-down in presence of anomalous scattering is the basis for experimental phasing. Dependent on the number of wavelengths used for the experiment one distinguishes between single anomalous diffraction (SAD) and multiple anomalous diffraction (MAD).
In a real X-ray diffraction experiment one has to consider that the protein crystal is not perfect but rather an assembly of nearly perfectly aligned domains. A protein is held together by chemical bonds. X-rays can influence and even break these bonds, which is known as radiation damage. In this context several parameters are introduced which help to judge the data quality. Using anomalous diffraction for solving the protein structure includes the solution of the so-called substructure of heavy atoms. The basic principle of the structure solution process and the relevant quality indicators will be described in the last part of the chapter.

## 2.1 The ideal X-ray diffraction experiment

In an ideal X-ray diffraction experiment X-rays are scattered by the electrons of an atom. These atoms can be part of a molecule of which many copies are symmetrically ordered in space, building a crystal lattice. In this section the process of the elastic scattering on proteins in a

crystal is as well described as the symmetry of protein crystals in real and in reciprocal space. On the basis of the crystal's symmetry reflections can be classified. The theory behind anomalous diffraction can be found in the last part of the section.

### 2.1.1 X-ray diffraction: from atoms to crystals

X-rays can be described as electromagnetic waves with a wavelength λ in the order of an Ångström ($10^{-10}$ m). When they interact with matter physical processes such as absorption, fluorescence and scattering take place. Like any other electromagnetic wave X-rays can be scattered by objects of similar size as their wavelength. Thereby X-rays are well suited for the investigation of matter on the atomic level as the diameter of an atom also has the size of an Ångström. When X-rays are elastically scattered by an atom the main interaction occurs between X-rays and electrons. The interaction of the atomic nucleus and the X-rays can be neglected, as the atomic nucleus has a too high moment of inertion to follow the fast oscillations of the incident wave. In the classic description of the scattering process a single free electron starts vibrating when it is exposed to the electric component of the incident X-rays. Thus it becomes a source of spherical X-ray waves itself, radiating like a dipole by $\pi$ out of phase with the incident wave. This process is known as Thomson scattering [5].

#### 2.1.1.1 X-ray diffraction of an atom

In an atom Z electrons occupy a certain volume which can in a classical approach be described as an electron density $\rho(\vec{r})$. The scattered radiation is a superposition of the emitted dipole radiation which is arising from the interaction between the X-rays and the electron density ρ in i volume elements at position $\vec{r_i}$. Describing the direction of the incident X-rays by a wave vector $\vec{k}$ with $|\vec{k}| = \frac{2\pi}{\lambda}$ the phase difference of the interaction with a volume element at the origin and the one at position $\vec{r}$ is the scalar product of $\vec{k}$ and $\vec{r}$ (compare Fig. 2.1). The scattered wave behaves locally like a plane wave with wave vector $\vec{k'}$, thus resulting in a phase difference of

$$\varphi(\vec{r}) = (\vec{k} - \vec{k'}) \cdot \vec{r} = \vec{Q} \cdot \vec{r} \tag{2.1}$$

between the incident wave and the one scattered at the volume elements at the origin and at $\vec{r}$ [5]. Here,

$$\vec{Q} = (\vec{k} - \vec{k'}) = \frac{4\pi}{\lambda} \sin(\theta) \tag{2.2}$$

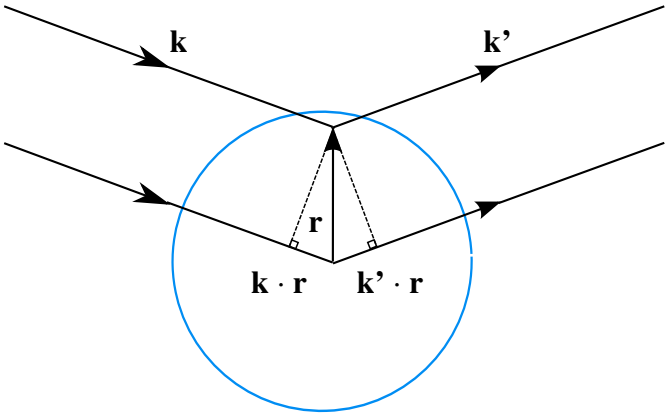



Figure 2.1: Elastic scattering of an X-ray with wave vector $\vec{k}$ from an atom to a direction $\vec{k'}$. The interaction of the X-ray with a volume element of the electron density at the origin and at a position $\vec{r}$ leads to the phase difference $(\vec{k} - \vec{k'}) \cdot \vec{r}$. Own representation based on [5].

is the scattering vector, while θ is defined as the angle between $\vec{k}$ and $\vec{k'}$. The scattering event is considered to be elastic here, i.e. $|\vec{k}| = |\vec{k'}|$. In this case, a volume element at $d\vec{r}$ will contribute $-r_0\rho(\vec{r})e^{i\vec{Q}\vec{r}}d\vec{r}$ to the emitted radiation, whereby $r_0 = \frac{e^2}{4\pi\epsilon_0 mc^2}$ is known as the Thomson scattering length, or as classical electron radius of the electron. Integration delivers the total scattering length $-r_0 f^0$ of an atom:

$$-r_0 f^0(\vec{Q}) = -r_0 \int \rho(\vec{r})\, e^{i\vec{Q}\cdot\vec{r}} d\vec{r} \tag{2.3}$$

with $f^0(\vec{Q})$ as the atomic form factor. Thus it becomes clear that the atomic form factor is a Fourier transform of the electron density $\rho(\vec{r})$. Furthermore, it is angle-dependent. If all different volume elements would scatter in phase and thereby $Q \to 0$, the atomic form factor would be the atomic number Z. Thereby it is evident that the amplitude of the scattered wave and therefore the amplitude of the emitted dipole radiation is proportional to the number of the atom's electrons [5].

#### 2.1.1.2 X-ray diffraction of molecules

In analogy to the atomic form factor the structure factor $F^{mol}$ can be calculated as the sum over all j atoms with atomic form factors $f_j$ at positions $\vec{r_j}$ in the molecule:

$$F^{mol}(\vec{Q}) = \sum_{\vec{r_j}} f_j(\vec{Q})\, e^{i\vec{Q}\cdot\vec{r_j}} = |F^{mol}(\vec{Q})|\, e^{i\varphi} \tag{2.4}$$

with φ as the phase [5]. Not long ago the scattering length of a single molecule was not sufficient to get a measurable signal, but since X-ray free electron lasers are available this is beginning to change [111]. Nevertheless it is much easier to determine the atomic structure of a molecule if many molecules of the same kind can be assembled in a crystal.

#### 2.1.1.3 X-ray diffraction of a crystal

A crystal is characterized by its unit cell. The unit cell consists of a structural unit, also called motif, which is repeated periodically in space. This periodic repetition is described by a lattice vector which reflects the translational symmetry of the crystal:

$$\vec{R_n} = n_1\vec{+}n_2\vec{b} + n_3\vec{c}, \tag{2.5}$$

with $n_1$, $n_2$ and $n_3$ as integers. Thus the position of any atom j in the crystal is given by $\vec{R_n} + \vec{r_j}$. Consequently, the structure factor of the crystal is composed of the unit cell structure factor and the lattice sum:

$$F^{crystal}(\vec{Q}) = \sum_{\vec{r_j}} f_j(\vec{Q})\, e^{i\vec{Q}\cdot\vec{r_j}} \sum_{\vec{R_n}} e^{i\vec{Q}\cdot\vec{R_n}}. \tag{2.6}$$

The lattice sum only becomes a real number if $\vec{Q}\cdot\vec{R_n} = 2\pi n$. This is only valid in case that the Laue condition comes true, i.e. $\vec{Q}$ coincides with a reciprocal lattice vector

$$\vec{G} = h\vec{a^*} + k\vec{b^*} + l\vec{c^*}, \tag{2.7}$$

whereby h, k, l are integers (also known as Miller indices) and $\vec{a^*}$, $\vec{b^*}$ and $\vec{c^*}$ are reciprocal lattice vectors. They are related to the lattice vectors by

$$\vec{a^*} = 2\pi\frac{\vec{b}\times\vec{c}}{\vec{a}\cdot(\vec{b}\times\vec{c})},\ \vec{b^*} = 2\pi\frac{\vec{a}\times\vec{c}}{\vec{a}\cdot(\vec{b}\times\vec{c})} \text{ and } \vec{c^*} = 2\pi\frac{\vec{a}\times\vec{b}}{\vec{a}\cdot(\vec{b}\times\vec{c})}. \tag{2.8}$$

An equivalent description is Bragg's law. It describes the condition of constructive interference of scattered waves by

$$n\lambda = 2d\sin\theta, n \in \mathbb{N} \tag{2.9}$$

with $d_{hkl} = \frac{2\pi}{|\vec{G}|}$ as the distance of lattice planes with the Miller indices hkl and θ the angle of incidence [5].

In crystallography, the norm of the lattice vectors and the angles between the lattice vectors in

real space are called unit cell constants. To reach any point within the unit cell dimensionless crystallographic coordinates x, y, z are introduced. These coordinates represent fractions of the unit cell constants and are therefore also known as fractional coordinates [86], i.e.

$$\vec{r_j} = x\vec{+}y\vec{b} + z\vec{c}. \tag{2.10}$$

If now constructive interference takes place ($\vec{Q} = \vec{G}$), reflections with Miller indices hkl are generated. From now on, hkl will be used synonymously for constructive interference and $\vec{h} = \vec{G}$. Evaluating equation 2.6 for a given reflection $\vec{h}$ with the Miller indices hkl, the complex structure factor becomes

$$F_{hkl} = \sum_j f_j \, e^{2\pi i \vec{r_j} \cdot \vec{h}} = |F_{hkl}| \, e^{i\varphi}. \tag{2.11}$$

In analogy to the scattering of one atom one can also consider the electron density $\rho(\vec{r})$ in the unit cell instead of the different atomic form factors:

$$F_{hkl} = V_{cell} \int_0^1 \int_0^1 \int_0^1 \rho(x, y, z) \, e^{2\pi i(hx+ky+lz)} \, dx \, dy \, dz. \tag{2.12}$$

One has to consider that the structure factor $F_{hkl}$ is complex, but the measurable quantity, i.e. the intensity of the reflections, is real:

$$I \approx |F_{hkl}|^2 = ||F_{hkl}| \, e^{i\varphi}|^2; \tag{2.13}$$

i.e. the phase information is lost during the measurement. But for solving the protein structure the electron density $\rho(\vec{r})$ needs to be calculated in real space as an inverse Fourier transformation of $F_{hkl}$ [10], taking into account that each hkl is representing a discrete reflection:

$$\rho(x, y, z) = \frac{1}{V} \sum_{hkl} |F_{hkl}| \, e^{-i\varphi} \, e^{-2\pi i(hx+ky+lz)}. \tag{2.14}$$

That being so, the phase information needs to be reconstructed somehow.

### 2.1.2 Symmetry in real and reciprocal space

A lattice can be described as a periodic sequence of points separated by the three basis vectors $\vec{a}$, $\vec{b}$ and $\vec{c}$ along the non-coplanar directions and is defined by equation 2.5 [43]. Depending on

the angle between the vectors and the lengths of the lattice vectors $\vec{a}$, $\vec{b}$ and $\vec{c}$, it is convenient to define fourteen so-called Bravais lattices and seven different crystal systems in three-dimensional space. In this context, a lattice is called primitive if it builds the minimum volume cell.
A mathematical group of symmetry operations is called a point group if it is leaving at least one point stationary and the base unchanged. In general, these symmetry operations are rotation, inversion, mirror planes and rotoreflection, resulting in 32 point groups in real space. Due to the chirality of macromolecules the only symmetry operations leaving the motif unchanged in real space are rotation, translation and a combination of both. Inversion and mirror planes change the handedness of a chiral motif and are therefore not relevant in protein crystallography.
Combining all fourteen Bravais lattices with the 32 point groups and translational elements like glide planes results 230 different space groups. As not all point groups apply for protein crystals due to their chirality the possible number of space groups is limited to 65. Considering now the symmetry of the crystal, a smallest unit which can be defined that can generate the complete crystal structure. This smallest unit is also known as the asymmetric unit (ASU) [10]. More details regarding Bravais lattices and point groups can be found in the books of Rupp [86] and Giacovazzo et al. [43].

#### 2.1.2.1 Friedel's law

Diffraction generates centrosymmetry in reciprocal space if anomalous scattering can be neglected, even if the point group of the crystal is non-centrosymmetric in real space [86]. This implies that the same set of the crystallographic planes in real space lead to both reflections hkl and $\overline{\text{hkl}}$. Calculating the structure factor $\text{F}_{\overline{\text{hkl}}}$ according to equation 2.12 results in

$$F_{\overline{hkl}} = V_{cell} \int_0^1 \int_0^1 \int_0^1 \rho(x,y,z)\, e^{2\pi i(-hx-ky-lz)}\, dx\, dy\, dz = F^*_{hkl} = |F_{hkl}|\, e^{-i\varphi}, \tag{2.15}$$

i.e. $\text{F}_{\overline{\text{hkl}}}$ is the complex conjugate of $\text{F}_{\text{hkl}}$. Therefore, $\text{F}_{\overline{\text{hkl}}}$ and $\text{F}_{\text{hkl}}$ have the same amplitude, but the signs of the phases are inverted [10]. This is known as Friedel's law:

$$|F^+| = |F_{hkl}| = |F_{\overline{hkl}}| = |F^-|, \tag{2.16}$$

so that the intensities of reflections building a Friedel pair are the same as it can be seen in the Argand diagram (compare Fig. 2.2). The reflections of a Friedel pair are also named Friedel opposites.

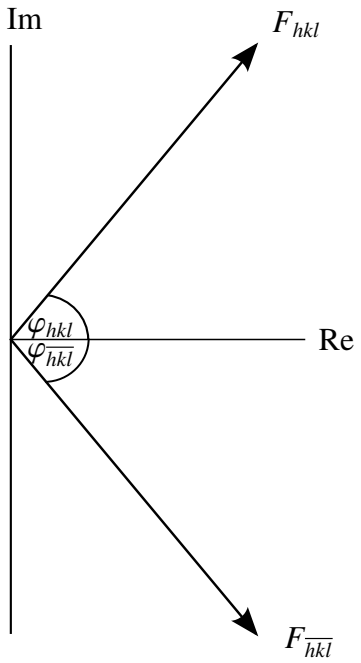



Figure 2.2: Diffraction generates centrosymmetry in absence of anomalous diffraction. Consequently, $|F_{hkl}|=|F_{\overline{hkl}}|$ and $\varphi_{hkl}=-\varphi_{\overline{hkl}}$.

#### 2.1.2.2 Reflection categories

If there is a rotational symmetry operator **R** in a space group which maps a reflection $\vec{h}$=hkl onto its Friedel opposite ($\vec{h}\mathbf{R} = -\vec{h}$), this reflection is called centric. Depending on the translational symmetry operator **T** the phase $\varphi_c$ of centric reflection is $\varphi_c = \pi\vec{h}\mathbf{T}$, i.e. $\varphi_c$ is limited to 0° or 180°.

Apart from the centrosymmetry induced by diffraction, reflections can be related to each other by the symmetry of the space group. These so-called symmetry equivalents of a reflection always have the same amplitude, even in the presence of anomalous diffraction. Each of these symmetry equivalent reflections with a structure factor $|F^+|$ has a Friedel opposite with a structure factor $|F^-|$. The symmetry equivalent reflections with a structure factor $|F^+|$ are called Bijvoet positive reflections or simply Bijvoet positives, the ones with $|F^-|$ are referred to as Bijvoet negatives. Thereby, a Bijvoet pair indicates a Bijvoet positive and a Bijvoet negative reflection, which can be, but not necessarily are a Friedel pair [86].

In case Friedel's law holds, one reflection can be representative of all symmetry equivalents, including centrosymmetry, and is referred to as unique reflection. Taking space group $P2_1$ as an

example, the reflection 2 1 3 stands as a unique reflection for the symmetry equivalent reflections $\overline{2}\ \overline{1}\ 3$, $\overline{2}\ \overline{1}\ \overline{3}$ and $2\ 1\ \overline{3}$. If this is not the case and there is anomalous diffraction, one unique reflection stands for all Bijvoet positive and another for all Bijvoet negatives. Sticking to the example of $P2_1$, one unique reflection 2 1 3 stands now only for itself and the other Bijvoet positive $\overline{2}\ \overline{1}\ 3$, and another for the Bijvoet negative reflections $\overline{2}\ \overline{1}\ \overline{3}$ and $2\ 1\ \overline{3}$, i.e. the number of unique reflection increases if Friedel's law does not hold. The completeness of a data set is calculated based on the number of unique reflections.
Another reflection category are systematically absent reflections. In an ideal crystal any symmetry operation that includes translational elements leads to systematic absences of certain reflections due to destructive interference of the scattered X-rays. The knowledge of the Laue group, i.e. the point group with the inversion centre induced by diffraction, in combination with systematic absences allows the unambiguous determination of the space group in the majority of cases [86].

### 2.1.3 Anomalous Diffraction

In analogy to the Thompson scattering of a free electron the scattering of an electron bound to an atom can be derived. Assuming that the incident X-ray wave triggers a damped harmonic oscillation of the electron with a frequency $\omega$ and a damping constant $\gamma$ the scattering factor of a bound electron is [31]:

$$f = \frac{\omega^2}{\omega^2 - \omega_0^2 - i\gamma\omega}. \tag{2.17}$$

Due to the damping term, the scattering factor of a bound electron is a complex quantity, involving a phase shift of the scattered X-rays. Assuming that the damping constant $\gamma$ is small, the atomic form factor defined in equation 2.3 becomes

$$f(\vec{Q}, \lambda) = f^0(\vec{Q}) + f'(\lambda) + if''(\lambda) \tag{2.18}$$

with the already known Thomson scattering $f^0$, f' as the real, but negative component and f'' as the imaginary component of the resonant scattering terms [5]. As mentioned before, the Thomson scattering $f^0$ is $\vec{Q}$-dependent as it is produced by all atomic electrons. However, the resonant scattering terms only play a role, if the energy of the incident wave is near the energy of an absorption edge of an atom in the molecule. Then, the X-rays interact with the electrons in the outer shells. The electrons in these shells are spatially so confined that the $\vec{Q}$-dependency

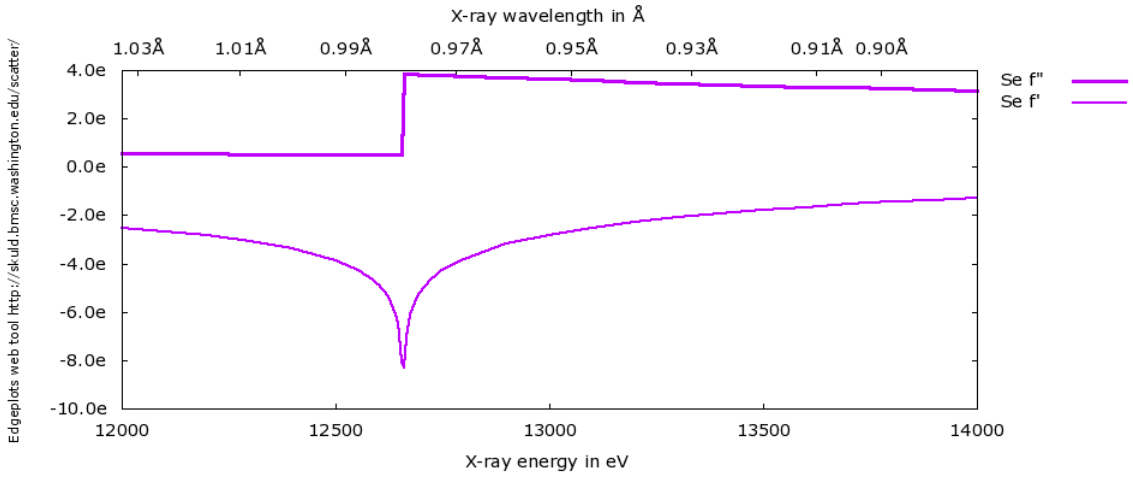



Figure 2.3: Relation of thetheoretical resonant scattering terms f' and f" of selenium in the energy range of 12 – 14 keV, depicted in electrons. Taken from http://skuld.bmsc.washington.edu/scatter/AS_form.html.

can be neglected. With increasing $\vec{Q}$, f($\vec{Q}$) decreases, whereas the resonant scattering terms stay the same. Thereby their relative contribution increases at large scattering angles [5].
The contribution of the dispersive term f' add up to $f^0$, while the contribution of f" leads to destructive interference with the incident wave and with that to absorption, as it adds a phase shift of $\frac{\pi}{2}$ in clockwise direction respective to f' [49]. Furthermore, it has to be considered that there is not only one electron but an assembly of oscillators in an atom. The corresponding resonant scattering terms can be analytically calculated [5], but in practice the energy-dependent atomic absorption coefficient $\mu$ is measured with fluorescence techniques. It is related to f" by

$$f''(E) = \frac{m_e c E}{2he^2}\mu(E) \tag{2.19}$$

with h as Planck's constant, c as the speed of light, $m_e$ as the mass and e the charge of an electron, respectively. Now, f' can be calculated with the Kramer-Kronig transformation [5]:

$$f'(\omega) = \frac{2}{\pi}\int_0^\infty \frac{\omega' f''(\omega')}{(\omega^2 - \omega'^2)} d\omega'. \tag{2.20}$$

An example for the relation between the resonant scattering terms f' and f" of selenium in the energy range of 12 – 14 keV can be found in Fig. 2.3.

#### 2.1.3.1 Breakdown of Friedel's law

If there are now A atoms in a protein having a strong anomalous scattering contribution at a certain wavelength, there are also N scatterers in this protein whose anomalous scattering contribution can be neglected. In the following, A will be referred to as anomalous scatterers and N as 'normal' scatterers, even though each of the latter also become anomalous scatterers at another wavelength. Considering this in the formulation of the structure factor equation together with the complex quantity arising from the anomalous scattering, it follows

$$F_{hkl} = \sum_{j}^{N} f_j \, e^{2\pi i \vec{h}\cdot\vec{r_j}} + \sum_{j}^{A} (f_j^0 + f_j' + i f_j'') e^{2\pi i \vec{h}\vec{r_j}}. \tag{2.21}$$

As also the anomalous scatterers provide normal scattering $f^0$, there is not only a 'normal' structure factor ${}^{o}F_N(\vec{h})$, but also a structure factor ${}^{o}F_A(\vec{h})$ arising from the 'normal' scattering of the anomalous scatterers:

$${}^{\lambda}F(\vec{h}) = |{}^{o}F_N(\vec{h})|e^{i\phi_N} + |{}^{o}F_A(\vec{h})|e^{i\phi_A} + {}^{\lambda}F_A'(\vec{h}) + i\,{}^{\lambda}F_A''(\vec{h}), \tag{2.22}$$

where ${}^{\lambda}F_A'$=f(f') and ${}^{\lambda}F_A$=f(f") correspond to the anomalous scattering. Accordingly, it follows for the Friedel mate:

$${}^{\lambda}F(-\vec{h}) = |{}^{o}F_N(-\vec{h})|e^{i\phi_N} + |{}^{o}F_A(-\vec{h})|e^{i\phi_A} + {}^{\lambda}F_A'(-\vec{h}) + i\,{}^{\lambda}F_A''(-\vec{h}). \tag{2.23}$$

With that, the structure factor equation 2.21 can be rewritten as

$$F^+ = F_{hkl} = {}^{o}F_T(\vec{h}) + F_A'(\lambda) + F_A''(\lambda), \tag{2.24}$$

or, as the anomalous structure factor can be expressed as a product of the normal structure factor of the anomalous scatterers $|{}^{o}F_A|$ and scattering factor ratios [51, 92]:

$$F_{hkl} = {}^{o}F_T(\vec{h}) + \sum_j \left(\frac{f_j'(\lambda)}{f_j^0} + i\frac{f_j''(\lambda)}{f_j^0}\right) {}^{o}F_{A_j}(\vec{h}). \tag{2.25}$$

The advantage of this formulation is the wavelength-independence of ${}^{o}F$ and ${}^{o}F_T$, which can be determined from normal scattering of the partial structure of anomalous scatterers. As already mentioned, the scattering factor ratios can be determined based on the measurement of absorption

spectra at the wavelength of interest [92].
However, on account of the phase shift introduced by f", Friedel's law no long holds, as it can be seen in Fig. 2.4. Consequently, there is a difference in the intensity of the Bijvoet positive and Bijvoet negative reflection named Bijvoet - or signed anomalous difference $\Delta F^{\pm}$:

$$\Delta F^{\pm} = |F^{+}| - |F^{-}|. \tag{2.26}$$

The phase shift due to anomalous scattering is $\alpha = \phi_T - \phi_A$.

## 2.1.4 Experimental phasing

How can anomalous scattering help to solve the phase problem? To determine the anomalous differences, an X-ray diffraction experiment can either be performed with a single or multiple wavelengths. Assuming that as in this thesis only one crystal is used per experiment, they are referred to as single and multiple anomalous diffraction experiments, or SAD and MAD, which are be explained in the following.

### 2.1.4.1 Multiple anomalous diffraction

By using two or more wavelengths, the anomalous differences can be computed analytically. To determine the unknown quantities $\alpha$, $|^{o}F_{T}|$ and $|^{o}F_{A}|$, the squared modulus of equation 2.25 needs to be calculated as it was first done by Karle [60] and then rearranged by Hendrickson for a single kind of anomalous scatterer to [52]:

$$|F^{\pm}|^2 = |^{o}F_T|^2 + a(\lambda)|^{o}F_A|^2 + b(\lambda)|^{o}F_T||^{o}F_A|\cos(\alpha) \pm c(\lambda)|^{o}F_T||^{o}F_A|\sin(\alpha), \tag{2.27}$$

where

$$a(\lambda) = \frac{f'^2(\lambda) + f''^2(\lambda)}{f^0}, \tag{2.28}$$

$$b(\lambda) = 2\frac{f'(\lambda)}{f^0}, \tag{2.29}$$

$$c(\lambda) = 2\frac{f''(\lambda)}{f^0}. \tag{2.30}$$

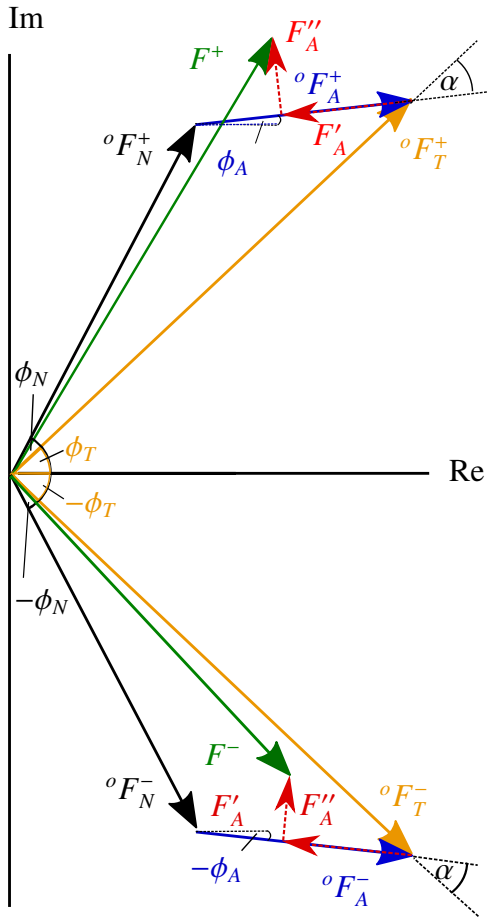



Figure 2.4: Visualisation of the breakdown of Friedel's law in the presence of an anomalous scatterer in an Argand diagram. An anomalous scatterer has a structure factor ${}^oF_A$ like any other 'normal' scatterer. Adding ${}^oF_A$ to the structure factor ${}^oF_N$ of a protein results in the total 'normal' scattering ${}^oF_T$. However, the anomalous part $F'_A$ reduces ${}^oF_A$ and $F''_A$ introduces a phase shift of $\frac{\pi}{2}$ in clockwise direction. It follows that the magnitude of the resulting structure factors $|F^+|$ and $|F^-|$ differ and that a phase shift of $\alpha = \phi_T - \phi_A$ is introduced by the anomalous scattering.

Hence a, b, and c are different for each wavelength but the same for all reflections, where the wavelength-independent quantities α, $|^{o}F_{T}|$ and $|^{o}F_{A}|$ are different for each reflection. If now a Bijvoet pair $|F^{+}|$ and $|F^{-}|$ is measured at two different wavelengths, four different equations of type 2.27 are generated. With that, the system of equations is overdetermined and the three wavelength-independent quantities α, $^{o}F_{T}$ and $^{o}F_{A}$ can be calculated. The Harker construction shown in Fig. 2.5 illustrates how the changes of the resonant scattering terms lead to an unambiguous solution of the phase problem [44].

Before the structure can be determined from the calculated quantities, the partial structure of the anomalous scatterers needs to be calculated. For this, either direct methods or the Patterson function are used. Direct methods are normally applied in small molecule crystallography, where the phases are calculated via *ab initio* methods, i.e. the phases are calculated based on the diffraction amplitudes only without any prior knowledge of the atomic positions. As a special class of *ab initio* methods, direct methods use probabilistic phase relations to derive phases from the measured intensities. Hence, normalized structure factors, also called E-values, are used:

$$E_{hkl} = |E_{hkl}|e^{i\varphi_{hkl}} = \frac{|F_{hkl}|}{\sqrt{\langle |F_{hkl}|^2 \rangle}} e^{i\varphi_{hkl}} = \frac{k\left\langle e^{-B_{iso}(\frac{\sin\theta}{\lambda})^2}\right\rangle^{-1} |F_{hkl}|_{meas}}{\sqrt{\epsilon_{hkl}\sum_{j=1}^{N} f_j^2}} e^{i\varphi_{hkl}}, \tag{2.31}$$

where the angle brackets indicate probabilistic or statistical expectation values, $|F_{hkl}|$ the already defined structure-factor magnitudes, $\varphi_{hkl}$ the corresponding phases, k is the absolute scaling factor for the measured magnitudes, $B_{iso}$ is an overall isotropic atomic mean-square displacement parameter, the $f_j$ are the atomic scattering factors for the N atoms in the unit cell, and the $\epsilon_{hkl} \geq 1$ are factors that account for the multiple enhancement of reflections due to space-group symmetry. The advantage of this formulation is that unlike the values for $\langle |F_{hkl}| \rangle$, the values of $\langle E_{hkl} \rangle$ are constant for concentric resolution shells, so that all reflections are on a common basis instead of being resolution dependent. Besides, no information about atomic positions are required [91].

In contrast, the auto-correlation of the electron density named Patterson function is not using phases, but the set of vectors $\vec{u}$. With the vectors between the heavy atoms, the Patterson function becomes

$$P(\vec{u}) = \sum_{\vec{h}} |F_{\vec{h}}|^2 e^{-2\pi i \vec{h}\vec{u}}, \tag{2.32}$$

and its solution can then generate a set of heavy atom positions and with that the so-called Patterson map. Thereafter, the phases of the partial structure $\phi_A$ can then be calculated based on

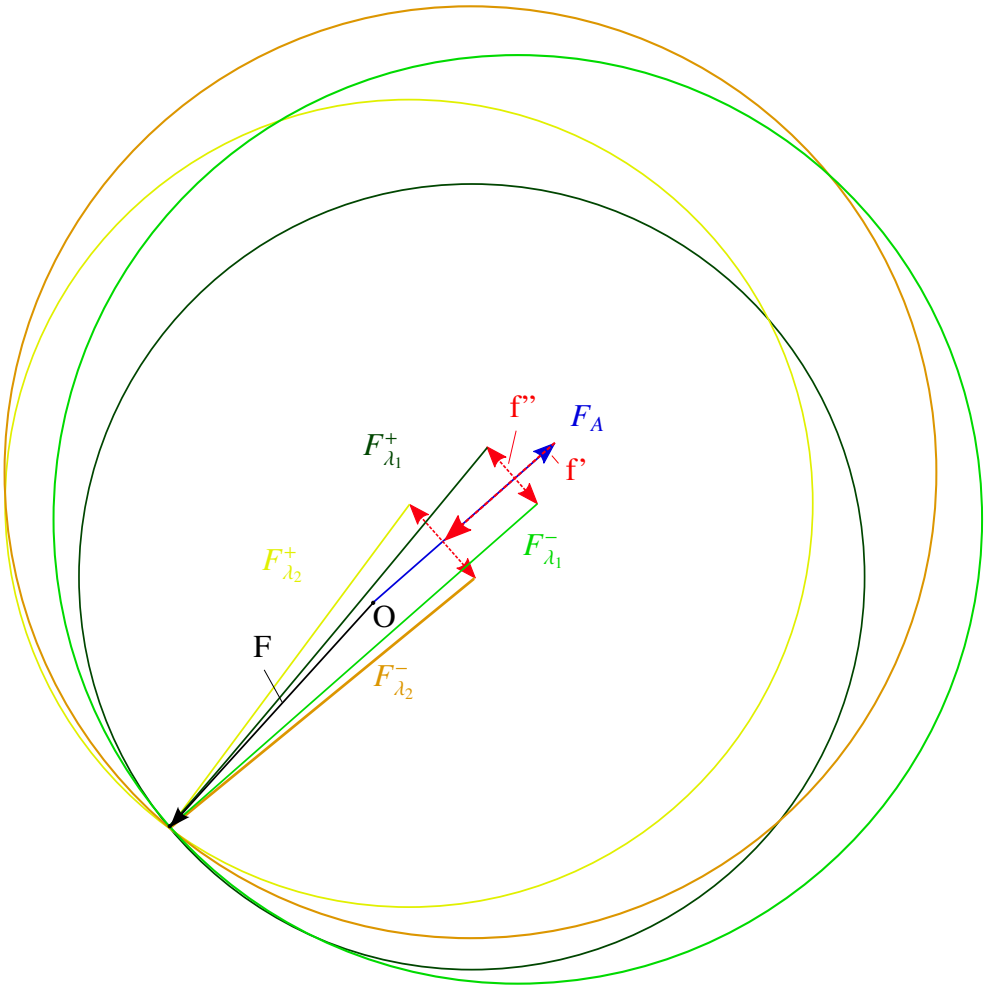



Figure 2.5: Harker construction for a two-wavelength MAD experiment. The observed structure factor amplitudes $F^+_{\lambda_1}$, $F^-_{\lambda_1}$, $F^+_{\lambda_2}$ and $F^{\lambda_1}$ arise from the measurements of the Bijvoet pairs at the two different wavelengths $\lambda_1$ and $\lambda_2$. Each of the structure factors consists of the wavelength-independent structure factors F for the protein without the anomalous scatterer and the structure factor of the anomalous scatterers $F_A$ and the wavelength-dependent contributions f' and f''. Drawing a circle for each of the structure factors with its amplitudes as radii leads to an intersection of these circles, indicating the unambiguous solution of the phase problem.

an atomic model of the anomalous scattering structure [10]. With the help of the experimentally determined phase difference, the phases for both enantiomers, i.e. the non-identical mirror- images, have to be calculated. One of the resulting electron density maps contains an image of the molecule which is defined by chemical plausibility or clear solvent boundaries [92]. Now, the electron density of the entire protein structure can be calculated by Fourier synthesis [92].

Being derived from several quantities, the treatment of errors is crucial for the MAD method. In the past, the least-squares fit of the phase equation to the experimental observation has been used to find weights for the $|^{o}F_{T}|$, $\phi_T$ Fourier synthesis. Nowadays, the weights are mainly calculated based on the Blow and Crick error model [11]. It computes phase probabilities from the lack of closure of phase triangles for each of the multiple observations of each reflection [92]. Based on that, Hendrickson and Lattman have shown that the probability distribution of a phase $\alpha$ can be expressed as

$$P(\phi_T) = Ne^{A\cos(\alpha)+B\sin(\alpha)+C\cos(\alpha)+D\sin(\alpha)}, \tag{2.33}$$

where A, B, C and D are Hendrickson-Lattman coefficients and N is a normalization constant [46]. In case of a MAD experiment, a unimodal phase distribution is present and the Hendrickson Lattman coefficients C and D are zero [76].

Given that the radiation damage is not too strong, a MAD experiment is often performed with three different wavelengths on one crystal to exploit the maximum dispersive signal. The first wavelength generally corresponds to energy at the peak of f". The second one correlates with energy at the minimum of f' which is at the same time the inflection point of f", and the third wavelength is at an energy remote from the two peaks. The Bijvoet difference is only $1-8\%$ of |F|, i.e. the measurements of these small changes need to be precise and accurate. To reduce radiation damage Friedel mates can be recorded closely in time using special measurement protocols as described in the section Materials and Methods [52, 96].

#### 2.1.4.2 Single anomalous diffraction

Before it became possible to perform MAD experiments at tunable synchrotron beamlines, it could be shown that the anomalous scattering information obtained from a single wavelength experiment can be sufficient to calculate phases and solve a protein structure [53]. With the possibility to substitute native sulphurs in proteins for selenium atoms providing a much higher anomalous signal at standard wavelengths [50], this method became more and more popular. However, the phases obtained by this method have two-fold ambiguity, as it will be shown in the following.

The anomalous difference of a protein with one type of anomalous scatterers measured at one wavelength has to be calculated from the only two observables available, i.e. $|F^+|^2$ and $|F^-|^2$.

Starting from equation 2.24, the anomalous difference can be expressed with [52]:

$$|F'_A| = \sqrt{\frac{1}{2}(|F^+|^2 + |F^-|^2) - |F''_A|} \tag{2.34}$$

in dependence of ${}^{o}\mathrm{F_T}$, $\mathrm{F''_A}$ and $\alpha$:

$$|F^+|^2 - |F^-|^2 = 4|{}^oF_T||F''_A|\sin(\alpha). \tag{2.35}$$

Most of the time, the contribution of the anomalous scattering is small in comparison to the total scattering. Then, $\frac{|F^+|-|F^-|}{2} \simeq F_T$ [52] and the Bijvoet difference becomes

$$\Delta F^{\pm} = |F^+| - |F^-| \simeq 2|F''_A|\sin(\alpha). \tag{2.36}$$

Due to the sine property $\sin(\alpha) = \sin(180^\circ - \alpha)$ there is an ambiguity [23] which is also visible in the Harker diagram (compare Fig. 2.6). Ramachandran and Raman [81] showed that

$$\phi_T = \alpha + \phi_A = \phi_A + 90^\circ \pm \theta, \tag{2.37}$$

where $\theta = \cos^{-1}(\frac{\Delta F^{\pm}}{2F''_A})$. Consequently, the ambiguity is only eliminated if $\theta = 90^\circ$. In any other case, the probability of the phase distribution resulting from anomalous scattering is bimodal and can be expressed as

$$P(\phi_T) = N\exp(-\frac{(\Delta F^{\pm} + 2F''_A\sin(\alpha))^2}{2E^2}), \tag{2.38}$$

where N is again the normalization factor and E the standard error estimation [23, 52]. To resolve the ambiguity, several approaches have been made. Nowadays, several probabilistic methods are available to determine the initial phases and their reliability. The most commonly used are maximum likelihood-based phasing [76] and density modification by applying solvent flattening, histogram matching and using non-crystallographic symmetry [19, 23].

#### 2.1.4.3 The phasing power of the anomalous signal

Before performing a SAD or MAD experiment, it is useful to estimate the phasing power of the anomalous scatterers. For doing so, the so-called Bijvoet ratio is defined, which is the mean ratio of the Bijvoet differences to the total protein structure factor amplitude. Taking into account

the number of non-hydrogen atoms $N_P$, the number of anomalous scatterers $N_A$, the anomalous scattering $f''_A$ and the effective scattering by the average atom in the structure $f_{eff}$ at a diffraction angle $\theta$, the Bijvoet ratio becomes

$$\frac{\langle \Delta F^{\pm} \rangle}{\langle F \rangle} = \sqrt{2} \sqrt{\frac{N_A}{N_p}} \frac{f''_A}{f_{eff}(\theta)}. \tag{2.39}$$

Unlike $f_{eff}(\theta)$, which decreases with increasing resolution, $f''_A$ is only dependent of the type of anomalous scatterers and the wavelength measured at. Hence, the Bijvoet ratio could be expected

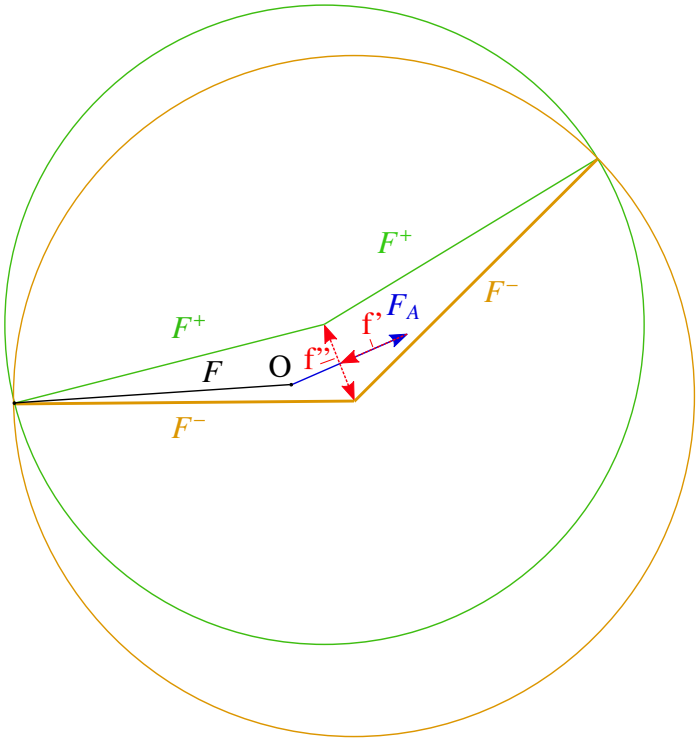



Figure 2.6: Harker construction for a SAD experiment. The two intersections of the circles with the radii $F^+$ and $F^-$ visualize the ambiguity. Unlike the MAD experiment, a single phase cannot be derived. The complex vector F represents the structure factor in absence of anomalous scattering.

to increase at high resolution [23]. It can more easily be calculated with

$$\frac{\langle \Delta F^{\pm} \rangle}{\langle F \rangle} = \sqrt{\frac{N_A}{2}} \frac{2f_A''}{\left\langle |F_p| \right\rangle}, \tag{2.40}$$

where $\left\langle |F_p| \right\rangle$ can be estimated as $\sqrt{346}$ times the square root of the number of amino acids in the protein [92]. The lower limit of the Bijvoet ratio with which successful SAD phasing can still be performed has been theoretically calculated, assuming error-free data. To be still able to solve the phase ambiguity of two weakly scattering sulphurs, the Bijvoet ratio must not fall below 0.6% [104].

However, the anomalous differences are often in the order of the measurement errors, leading to a serious overestimation of $\langle \Delta F^{\pm} \rangle$. That is why the accuracy of the anomalous differences is of high importance. The multiplicity of intensity observations for a given unique reflection is of particular importance, as it reduces the statistical error and thus, through error propagation, enhances the accuracy of $\langle \Delta F^{\pm} \rangle$ [16, 23]. It has been shown that if a Bijvoet ratio lower than 1% is to be used for the estimation of phases, the errors in the intensities should not exceed 2% [80].

### 2.1.5 Phasing with molecular replacement

In case that there is already a protein structure available with a similar amino acid sequence as the unknown one, molecular replacement can be considered to obtain initial phases. Molecular replacement refers to a method first described by Rossman and Blow [85]. They and others observed that many larger proteins have similar subunits, often with a different orientation, but nevertheless built from similar or even the same atoms. Consequently, the known structure of a homologous protein could be rotated and translated in the unit cell or the asymmetric unit in a way that its calculated diffraction data fit the measured data [86]. This is mostly done by optimizing the correlation of intra-molecular vectors using the Patterson function.

Nowadays, with more than 100, 000 available structures in the Protein Data Bank (PDB), it is the most popular method for solving protein structures [87]. Given that the structural similarity is high enough, a sequence identity of only 25% and an r.m.s. deviation of $< 2.0$ Å between the C atoms of the model and the target structure can be sufficient for successful molecular replacement [87, 96]. However, as the phases for the reconstruction of the electron density emerge entirely from the search model, the electron density maps are highly susceptible to model bias [86]. First and foremost, the initial search model has to be entirely correct. Another issue is the resolution

of the measured data. If the resolution is far from atomic, electron density maps are prone to human misinterpretation, and once a model has been fit incorrectly to a part of the map, most refinement methods reinforce the wrong features as well as correct ones [55].

## 2.2 The real X-ray diffraction experiment

When performing an X-ray diffraction experiment, one has to be aware that a real crystal has some properties not considered before. Furthermore, one has to know the experimental limitations to make some strategic decisions beforehand. This comprises the X-ray source, the temperature at which the data should be collected and the method with which the structure should be solved with. Additionally, the size and the composition of the protein crystal influences the data collection strategies.
During the real X-ray diffraction experiment, X-rays are partially absorbed by the crystal, leading to radiation damage. A further differentiation will be presented here, as well as methods for calculating the absorbed dose and strategies for dealing with radiation damage.

### 2.2.1 The real crystal

In a real crystal, one has to consider the different shape of the electron density of each atom and the disorder of the structure due to thermal vibrations and crystal disorder. Therefore, the structure factor is calculated as a sum of the scattering due to each of the N atoms in the unit cell:

$$F_{hkl} = \sum_{N} f_i e^{2\pi i(hx_i+ky_i+lz_i)} e^{-B_i(\frac{\sin\theta}{\lambda})^2}. \tag{2.41}$$

The last exponential term is known as the Debye-Waller factor. It decreases when the resolution increases and smears the electron density by a Gaussian shape and represents thermal displacement (dynamic) and crystal disorder (static). The quantity B represents the width of the smearing, i.e. if an atom has a root-mean-square displacement u, the atomic B-factor equals $8\pi^2u^2$ [10].
To account for the fact that the atomic scattering factor is reduced with increasing resolution (compare section 2.1.4.3), an overall temperature factor B, also named Wilson-B-factor, is estimated. The diffracted intensities are divided into a set of shells according to the resolution and the mean intensity $\bar{I}$ is calculated for each shell. A graph is drawn by plotting $\log_e(\frac{\bar{I}}{\overline{f^2}})$ against $(\frac{\sin\theta}{\lambda})^2$, where $\overline{f^2}$ is calculated based on an average protein f. Ideally this results in a straight line with a slope equal to -B, accounting for the effective overall Debye-Waller factor [10].
Apart from the disorder it is also important to give thought to the fact that in contrast to ionic crystals, the motifs of macromolecular crystals are normally irregularly shaped. Because of this, the packing cannot occupy all space. The resulting voids are filled with an aqueous solution consisting of chemicals specifically leading to the crystallization of a certain protein. The frac-

tion of the solvent within the crystal can vary between 0.27 − 0.65 of the total volume and is referred to as solvent content. Therefore the interaction is based on a few weak bonds between the molecules, making the protein crystals soft and fragile [10].

### 2.2.2 Experimental limitations

According to Bragg's law 2.9, there is a diffraction limit

$$d_{min} = \frac{\lambda}{2 \sin \theta_{max}}. \tag{2.42}$$

In theory, this diffraction limit is only dependent on the quality of the crystal. In practice, the diffraction limit or the best obtainable resolution is also dependent on the size of the detector y and the minimum distance x between the crystal and the detector:

$$\tan 2\theta_{max} = \frac{y/2}{x}, \tag{2.43}$$

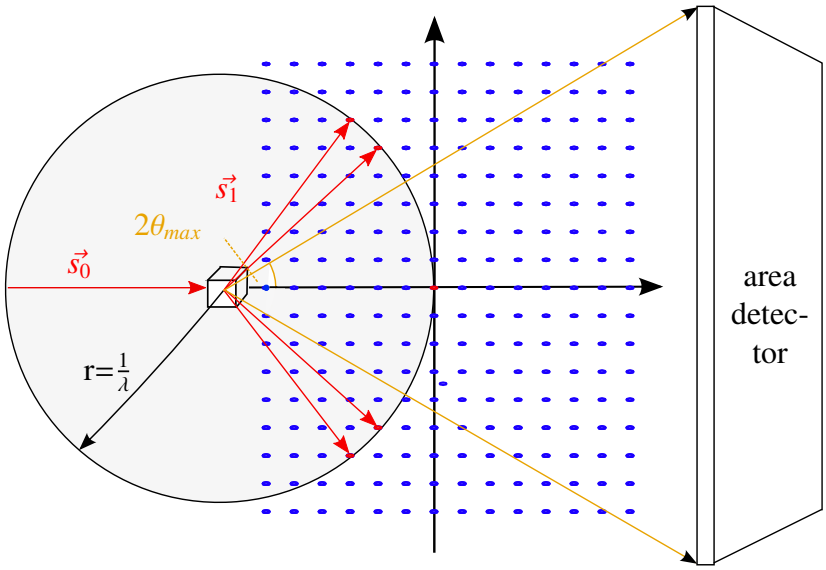



Figure 2.7: Not all reciprocal lattice points (blue) intersect with the Ewald sphere, whose radii are indicated with the red arrows $\vec{s_1}$. The geometrical restrictions of the maximum resolution are half the detector size $\frac{y}{2}$ and the distance x between detector and sample, if symmetric diffraction images are to be collected. Own representation based on [86].

if fully symmetric diffraction images are to be recorded. Consequently, the resolution of a well diffracting crystal measured with X-ray energies normally used in protein crystallography is mostly limited by the detector distance. Besides, not all reciprocal lattice points intersect the Ewald sphere, even if the crystal is rotated.

It has also to be considered that a protein crystal is rarely a true single crystal but rather a mosaic of nearly perfectly aligned domains. The parameter for the misalignment is called mosaicity and is measured in degrees [86]. A high mosaicity is leading to a broad intensity profile of a Bragg reflection. If the rotation-angle increment per diffraction image, from now on referred to as oscillation range, is less or equal than the mosaicity of the crystal, the Bragg reflection is finely sampled in three dimensions - two for the area detector and a third for each rotation angle increment of the crystal. In case that the intensity profile is incomplete, the reflection is referred to as partial, otherwise it is called a full reflection [79].

Another issue requiring consideration in a real experiment is the absorption of X-rays. The relative decrease of an intensity $I_0$ to I due to absorption is

$$\frac{I}{I_0} = e^{-\mu x} = e^{-N\sigma x}, \tag{2.44}$$

where $\mu$ is the absorption coefficient, N the number of atoms per volume unit, $\sigma$ the element-specific total absorption cross-section and x the path length of the X-rays. The integrated diffracted intensity of a crystal is dependent on the wavelength and the scattering angle $2\theta$:

$$I \propto \frac{\lambda^3 x^3}{\sin(2\theta)} e^{-\mu x}. \tag{2.45}$$

For small scattering angles, $2\theta \approx \frac{\lambda}{d}$ and it follows

$$I \propto \lambda^2 x^3 e^{-\mu x}, \tag{2.46}$$

i.e. scattering increases with $\lambda^2$, but absorption is increasing with $\lambda^3$ at the same time [28]. If it is now the aim to perform a sulphur-SAD experiment, one would aim for a long wavelength. The longer the wavelength, the higher the anomalous signal, but with the stronger absorption at longer wavelength the noise of the data is also increasing. For this reason, one has to find a compromise. Fig. 2.8 shows the transmission $\frac{I}{I_0}$ of X-rays in 153 mm air, which is approximately the minimum detector distance achievable at the EMBL beamlines.

To reduce absorption, a helium-purged beam path can be used, as helium absorbs X-rays by

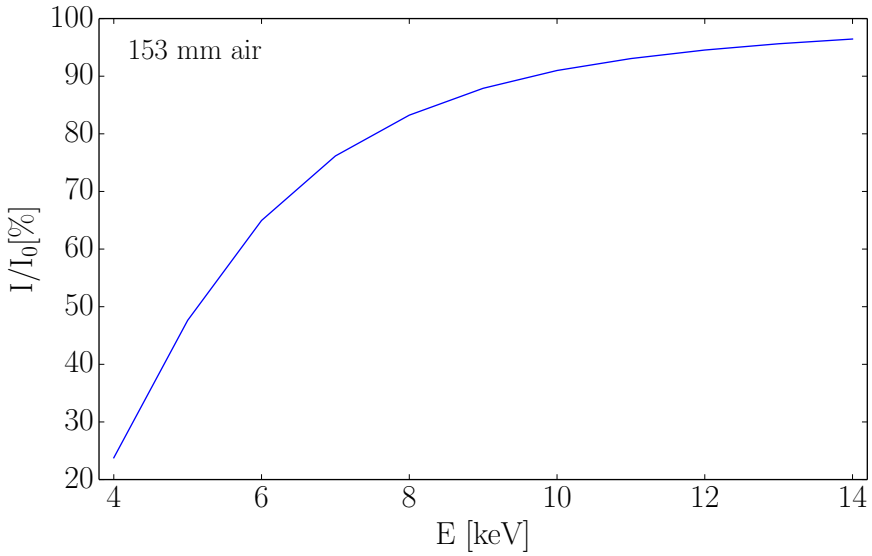



Figure 2.8: Transmission of the initial intensity $I_0$ depending on the energy for X-rays passing a distance of 153 mm in air. The values were calculated with the online X-ray attenuation and absorption calculator (http://web-docs.gsi.de/ stoe_exp/web_programs/x_ray_absorption/index.php).

two to three orders of magnitude less than air. This is a strict requirement for experiments with wavelengths longer than 3 Å. Another option is to perform the experiments in an evacuated environment [28].

### 2.2.3 Radiation damage

The small part of the primary X-ray beam actually interacting with the sample is subdivided in three processes:

- elastic scattering contributing to the diffraction pattern,
- inelastic scattering contributing to the background and
- the photo-electric absorption.

At an energy of 12.4 keV, the elastic and the inelastic scattering account for only 8 % each, while the photo-electric absorption dominates with 84 % [82]. The direct interaction of the X-rays with the electrons is called primary radiation damage. As photoelectric absorption results in the emission of energetic electrons via the photoelectric, Auger and Compton effect, a cascade

of secondary electrons is generated [93]. Thus each photo-electron can generate radicals and up to 500 secondary lower energy electrons. They migrate to sites of high electron affinity like metal centres or disulphide bonds, long before the crystalline diffraction is lost [82]. The damage caused is known as secondary radiation damage. While primary radiation damage is dose dependent only, the resulting secondary radiation damage is time and temperature dependent [38].
In macromolecular crystallography, radiation damage is observable in both reciprocal space (global damage) and real space (specific damage) [56]. The effect of radiation damage becomes visible via five symptoms [82]:

- high resolution reflections are fading with increasing exposure,
- unit cell parameters are changing and mosaicity is increasing (non-isomorphism),
- Wilson- and atomic B-factors are increasing,
- colour changes in the irradiated volume of the crystal and
- site-specific damage.

The fading of the high resolution reflections is accompanied with an increasing noise. Together with the non-isomorphism it particularly hampers the reliable measurements of small dispersive signals [73]. The B-factors do not increase all in the same way, as the presence of site-specific damage already suggests. Specific damage occurs in a clearly defined order as a function of the absorbed X-rays, starting with the reduction of metal centres, followed by elongation and scission of disulphide bonds, and then decarboxylation of aspartates and glutamates [41].

#### 2.2.3.1 The quantification of radiation damage

Blake and Philips [9] were the first to perform a radiation damage study. They found that the damage was proportional to dose, i.e. the mean energy deposited to matter per unit mass by ionizing radiation, and suspected that the damage might be structurally specific. Within the last fifteen years, several models have been developed to calculate the dose absorbed by the crystal. The model most recently developed is the one by Zeldin et al. [115]. By calculating the distribution of dose within the crystal volume across the oscillation range, a time-resolved picture of the dose state of the crystal can be drawn. The diffraction-weighted dose DWD is

calculated as follows:

$$DWD = \frac{\int_{t_{i-1}}^{t_i} \int_{crystal} D(V,t)F(V,t)dVdt}{\int_{t_{i-1}}^{t_i} \int_{crystal} F(V,t)dVdt}, \tag{2.47}$$

where D(V,t) is the total cumulative dose (MGy) at position V and t is the experimental coordinate, which is proportional to the goniometer angle for a constant rotation rate:

$$D(V,t) \propto \int_0^t F(V,t) \tag{2.48}$$

and

$$F(V,t) = F_{surface}(V_x, V_y, t) \cdot e^{-\mu_{abs} \cdot depth}. \tag{2.49}$$

$F_{surface}$ is the intensity of the beam at the surface of the crystal at the (x, y) coordinates associated with position V, and V is a function of the beam profile and the total flux. Also, $\mu_{abs}$ is the absorption coefficient of the crystal which is effective in the depth, i.e. the distance along the beam axis from the front face of the crystal to position V. Consequently, F(V, t) is the weighting function, which is complying to how much the volume element V makes for the diffraction pattern at time t. The DWD is normalized to the weighting function F(V, t) so that the unit is the one of a dose [113]. The average diffraction weighted dose is calculating the mean DWD over an oscillation range.

The model includes photoelectric absorption, the fluorescent emission probability, the probability that fluorescent photons might escape the crystal and inelastic X-ray scattering.

The above mentioned mechanisms have been implemented in the program RADDOSE-3D. It requires a number of input parameters regarding the crystal, the beam and the data collection parameters. For the crystal, the crystal and unit cell size, the number of amino acids, the number of heavy atoms in the monomeric protein, the concentration of heavy atoms in the solvent and the solvent content have to be known. Regarding the beam, the flux, the energy and the beam type and size have to be given; further collection parameters are the exposure time, the oscillation range and the angular resolution. The DWD has proven to be effective at predicting intensity loss under a variety of dose contrast conditions [113]. Therefore, the average DWD is used in this thesis.

#### 2.2.3.2 How to reduce radiation damage

As mentioned before, secondary radiation damage is time- and temperature dependent. Accordingly, first experiments with a cooled sample were performed in the 1970*s*, reporting a reduced intensity loss. At the beginning of the 1990*s*, measurements at cryogenic temperatures started to increase exponentially. At 100 K, radiation damage is dramatically reduced since the mobility of radicals is much lower compared to room temperature. Furthermore, atomic motion is reduced at cryogenic temperatures. Depending on the relative degree of dynamic and static disorder in crystals of a particular macromolecule, this can make higher-resolution data accessible. When cryo-cooling the crystal, the formation of ice within the crystal has to be avoided. For doing so, some of the water in the solvent should be replaced with a cryo-protectant such as glycerol. The mosaicity of cryo-cooled crystals is usually slightly higher than the one of crystals measured at room temperature [38, 39].
Apart from cooling the crystal, radiation damage can also be minimized by adapting the data collection strategy. First of all, it should be avoided that the crystal is exposed inhomogeneously, because this leads to an inhomogeneous distribution of dose within the crystal and thereby to inhomogeneous data. This is the case when a) the crystal is bigger than the beam so that 'fresh' material gets into the beam while rotation, or b) when the beam is non-uniform (Gaussian- like). Consequently, it would be the best to measure protein crystals with a flat, so-called top-hat beam profile bigger than the crystal. Another option especially suited for long, needle-shaped crystals is to perform a helical scan with a beam which is either narrow along the rotation axis and matches the crystal size along the perpendicular axis or smaller than the crystal in both dimensions [114].
As experimental phasing generally requires data of high multiplicity to measure the small anomalous differences as accurately as possible, radiation damage is even more problematic. Being the basis for calculating the anomalous differences, the Friedel pairs should be measured with a comparable radiation damage. To achieve this, the crystal can be aligned in a way that the Friedel pairs can be recorded on one diffraction image. Because this alignment is not very easy, it is also possible to first record a small wedge and to measure the same wedge after rotating the crystal by 180°. This set-up is called 'inverse beam geometry'. Recently, it became also popular for SAD measurements to measure and merge highly redundant data from one crystal at a very low dose of 0.5 MGy per full turn [105]. Therefore this dose is far below the Henderson limit of $D_{1/2}$ = 20 MGy. $D_{1/2}$ is the calculated dose limit for the loss of half the diffraction intensity

of a protein crystal. The Henderson limit was calculated in analogy with observed destruction rates in electron crystallography [40]. However, a theoretical dose limit only takes the physics of the energy loss in the crystal into account, and not its chemistry. For example, if a crystal has particularly radiation susceptible amino acids (e.g. aspartates) which might form the only crystal contacts, radiation damage may cause the crystal to lose its order long before predicted by the Henderson limit [38].

## 2.3 Solving and refining macromolecular structures

For macromolecular crystallography, there exists a large selection of programs for data processing, data analysis and structure determination. Therefore, automated structure determination via scripting or with graphical interfaces is possible. In the following, the main programs used in this thesis will shortly be described, followed by a section dealing with data quality.

### 2.3.1 XDS

XDS is a program to process the collected diffraction images to a list of indexed reflections, giving i.e. information about intensity I, $\sigma(I)$ and the oscillation angle the reflection was collected at. At the EMBL beamlines, the input file for XDS is automatically written during data collection, including information such as detector specifications, sample-detector distance, the energy and the number of collected frames. The user has only to decide whether the data should be processed with Friedel's law as true or false.

XDS is organized into eight major subroutines. First of all, the program calculates the spatial correction of each detector pixel, followed by the generation of a look-up table for background, detector noise and the gain, i.e. a table for the expected variation of the pixel contents in the background region of a data image. In the third step, strong diffraction spots are located which are adjacent in three dimensions. Up to 3000 of these strong diffraction spots are then used in the next subroutine together with the information from the input file to find the orientation, metric and symmetry of the crystal lattice and to refine all or a specified subset of these parameters. Based on this, a first indexing is performed. In the fifth step, the background table is modified in a way that it does not allow reflections in the regions that are obscured by hardware or which the user does not want to include, for instance by setting a certain resolution limit in the input file. Afterwards, a report is generated to support the planning of the data collection based on the estimation of the

completeness of new reflection data expected to be collected for each given starting angle and total crystal rotation [58]. However, this subroutine is practically never used. Before performing the integration of the reflections, they are identified based on the modelled reflection positions in the detector plane [57]. The integration itself is performed by first generating spot templates by superimposing the profiles of strong reflections after their mapping to the Ewald sphere. Then, the actual integration is carried out by profile fitting with respect to the before determined spot shape.

In a final step called 'CORRECT', basically all parameters are (re)-refined, most importantly the geometry of the experiment like cell parameters, crystal orientation, distance, beam direction and spindle direction. The intensities and standard deviations are corrected and written to a file, and the space group is determined if unknown. For the correction factors, it is crucial whether Friedel's law holds, as the scaling procedures scale the variances of individual observations such that they match the experimental spread of symmetry-related observations. The variance-scaling formula is

$$v(I) = a(v_0(I) + bI^2), \tag{2.50}$$

where the initial estimate $v_0(I)$ is obtained from the INTEGRATE step and a and b are chosen to minimize discrepancies between v(I) and the variance estimated from sample statistics of symmetry-related reflections [58]. The parameters a and b can also be interpreted as follows: the first component a is random error, and the other component is systematic error which is scaled by a·b. In the variance-scaling formula, the variance is dominated by the systematic error $a \cdot b \cdot I^2$ for strong and well-measured reflections, while for weak reflections, $a \cdot v_0(I)$, the variance from counting statistics, dominates. In the output file of the CORRECT step (CORRECT.LP), the parameters a and b are as well given as the $I/\sigma(I)^{asymptotic}$ = ISa= $\frac{1}{\sqrt{ab}}$, which is the I/σ of an infinite strong reflection. Without systematic errors, ISa would be infinite; however, in reality, the ISa is finite and is the upper limit of I/σ of any observation in the dataset [3, 26]. Apart from that, the CORRECT steps generates tables reporting on the completeness and the quality of the data [58].

As XDS has been the only program capable of dealing with data of the PILATUS format for a long time and because it still is the only freeware program able to perform three-dimensional integration, it is the most commonly used for data collected at a synchrotron.

### 2.3.2 XSCALE

The scaling program XSCALE puts one or more files obtained from data processing with XDS on a common scale and can optionally merge symmetry-equivalent reflections. It can produce one combined data set file with scaled intensities or separate data sets in different output files, but with data on a common scale, as for data obtained from a MAD experiment [58]. Therefore, the data sets are individually multiplied with a factor $Ke^{2B\sin\theta/\lambda^2}$ involving two parameters, K and B. The parameter values are assigned so that the resulting correction factors fit best to the observed intensity ratios of common reflections in each pair of data sets. The aim of a more detailed correction is to remove the correlation of a reflection with image number and resolution, location in the detector plane and the image number and different detector surface regions. The correction factors are calculated by minimizing iteratively a function including reciprocal factors, the weighted mean intensities and standard deviations of symmetry related reflections for different grid regions [57].
Depending on the further use, it can be chosen whether the data should be merged, with both options for Friedel's law. XSCALE also allows to compensate radiation damage by the option of a zero-dose extrapolation [58]. Like for XDS, XSCALE is suited for parallel processing.

### 2.3.3 SHELX for structure determination with experimental phasing

For performing experimental phasing, the freeware programs SHELXC, SHELXD and SHELXE are very popular, because they enable simple, robust and efficient experimental phasing of macromolecules by the SAD, MAD, SIR, SIRAS and RIP (radiation induced phasing) methods. The programs are run from the command line or via scripts.

#### 2.3.3.1 SHELXC

The program SHELXC works with one or more lists of reflections as input files generated from different programs, i.e. with the output from XDS. In preparation for the following work with direct methods, it is calculating normalized structure vectors and performs local scaling. On account of that, it is advantageous to leave the reflections unmerged. SHELXC provides a statistical analysis of the input data, estimates the heavy structure factors $F_A$ and, in case of SAD, the phase shifts $\alpha$. For MAD phasing, these factors are calculated from the overdetermined equation system (compare equation 2.27). For SAD phasing, the first estimate of $\alpha$ is either 90° or 270°,

which relates to the intensity of a reflection hkl in comparison to the intensity of $\overline{\text{hkl}}$ (see equation 2.36). The stronger the anomalous difference, the better these estimates. In this context, the <d"/σ>=<$\frac{|\Delta F|}{\sigma(|\Delta F|)}$ >values calculated dependent on the resolution are critical. In the MAD case, the Pearson correlation coefficient CC of the different data sets are calculated for different resolution shells. Apart from that, the input files for SHELXD and SHELXE are generated [90]. With HKL2MAP, there is also a graphical user interface available [77]. In the following, the main operation mode of the three programs will shortly as well be described as important parameters. Due to the variety of parameters and options especially for SHELXE, only the ones used in the scope of this work will be explained.

#### 2.3.3.2 SHELXD

The program SHELXD locates the heavy atoms by using the dual-space recycling approach. Hence, the structure solution problem is reduced from several thousands to the limited number of heavy atoms (substructure). As the name "dual-space recycling" already suggests, this approach (also known as Shake-and-Bake algorithm) alternates between real and reciprocal space. Belonging to the direct methods (see section 2.1.4.1), it is based on the strongest 15% of the normalized structure factors E in each resolution shell, where |E| is derived from $|F_A|$, i.e. the amplitudes of the heavy atom structures. In the SAD case, $|\Delta F^{\pm}|$ are taken as lower limit estimates for $|F_A|$. Since the normalized structure factors used for direct methods emphasize high-resolution data, the resolution cut-off is critical so that not too much noise is added. The data should be cut at the resolution where the d"/σ value falls below 0.8, or, in case of MAD phasing, where the CC of the different data sets is less than $25 - 30\%$ [88].

To find the initial atoms, a special form of the Patterson Minimum Function (PMF) is used. Two atoms are placed in a unit cell and all their symmetry equivalents generated [88]. The strongest peaks in the Patterson function can be considered as potential two-atom search fragments with a fixed vector distance between the two atoms. These vectors can only be translated. A large number of random positions in the unit cell are tested for the resulting two-atom fragment which is chosen pseudo-randomly from the Patterson peak list, favouring the higher peaks. The position of the two-atom fragment that gives the best Patterson super- position minimum function, based on the two atoms and all their symmetry equivalents, is used as a so-called Patterson seed. By using these two atoms and their symmetry equivalents a full-symmetry Patterson superposition minimum function is generated. The resulting peak list is then searched to obtain further heavy-atom positions, up to N atoms which is corresponding to about 120% of the value the user entered

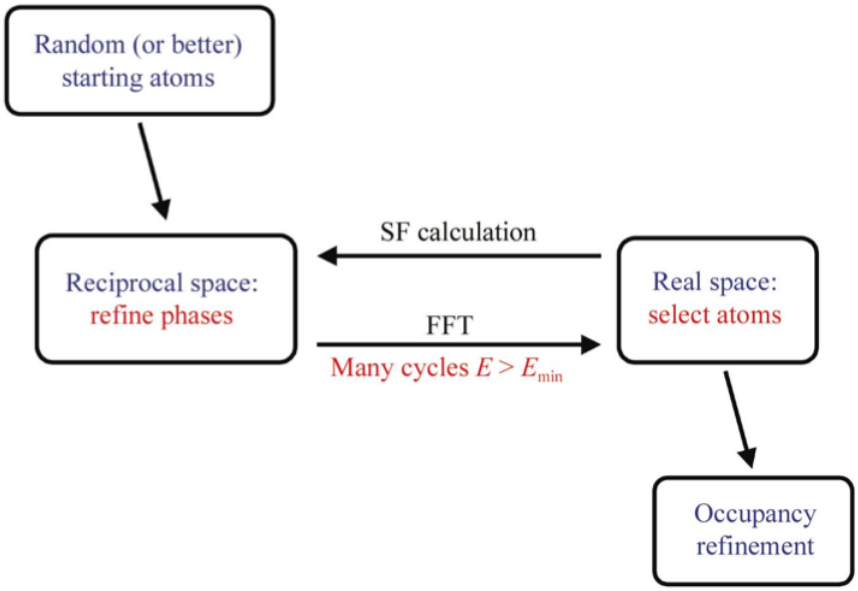



Figure 2.9: Flow chart for the dual-space recycling approach to solve the substructure. Taken from [88].

as the number of expected heavy atoms. In this manner, an unlimited number of different starting atoms consistent with the Patterson function can be generated. These atoms are now used in the dual space recycling approach [90]. In reciprocal space, the phases calculated from the N Patterson peaks are expanded or refined from the 40% most reliable using the tangent formula invented by Karle and Hauptman [59]. To avoid a phase divergence away from a chemically sensible (e.g. equal-atom) arrangement of sites, real-space cycles are necessary to force the constraint that the N sites have approximately equal scattering power. It is also possible to search for 'super sulphurs' with the option 'DSUL', if the number of disulphide bridges are known. 'DSUL' can be helpful if scattering of sulphurs is weak at the given wavelength [89] and if the used anomalous data have an effective high-resolution limit, with which disulphide bonds cannot be resolved, i.e. when the resolution is worse than 2.0 Å. Strong super-sulphur peaks can be located even at lower resolution and 'DSUL' will split them geometrically into separate sulphur positions, so that the overall substructure will ideally be complete. The dual-space recycling is normally performed for several hundred or more sets of N random starting atoms, with typically 2N cycles for each (compare Fig. 2.9) [88].

Potential solutions are identified by the correlation coefficient between the calculated ($E_c$) and

observed ($E_o$) normalized structure factors:

$$CC = 100\frac{\sum wE_oE_c \sum w - \sum wE_o \sum wE_c}{\sqrt{[w\sum E_o^2 \sum w - (\sum wE_o)^2][\sum wE_c^2 \sum w - (wE_c)^2]}} \tag{2.51}$$

with weights w=$(1 + g\sigma^2(E))^{-1}$ and g=0.1 as a default value to weigh down less reliable ΔF values. This correlation coefficient is calculated as well for the weak E values as for all E values. By adding these two correlation coefficients, the final measure for failure or success is defined: CFOM = $CC_{all}$+$CC_{weak}$. The substructure is identified, when a group of correlation coefficients is well clear off the rest [88]. Normally, the CFOM is at least 30%, when the substructure is solved, but the values also depend on resolution limits and whether SAD or MAD is used. It can even happen that the CFOM is higher than 40% and that there is no group of correlation coefficients separated from the other values. In this case, it makes sense to have a look at the number of found sites with an occupancy greater than 0.3 in comparison to the number of searched sited. The occupancy refers to the probability of a heavy atom position, and a rule of thumb is that with an occupancy greater 0.3, the heavy atoms are considered reliable.

#### 2.3.3.3 SHELXE

SHELXE is a program for experimental phasing based on the heavy atoms found by SHELXD. It provides the option to refine the heavy atom positions, and then uses the heavy atom phases $\phi_A$ to obtain the starting phases according to $\phi_T = \phi_A + \alpha$, and those are then improved via density modification. For density modification, the sphere of influence algorithm is used. It is based on the fact that the 1,3-distance in macromolecules is often close to 2.42 Å. On account of that, a sphere with a radius of 2.42 Å is constructed around each voxel of the electron density map. If the density in this spherical surface has a high variance V, i.e. probably contains atoms, the voxel at the centre of the sphere is also a possible atomic position. By sorting the variances, protein regions with high variances and solvent regions with low variances can be determined. These individual variances V of all voxels within an asymmetric unit have a variance of their own, which is named 'contrast.' After some density modification cycles, this contrast is nearly always higher for the correct enantiomorph [89]. It works best if the solvent content is high [90]. For cases where density modification alone is not successful, the improvement of phases is aided by an iterative backbone auto-tracing algorithm that is run in macrocycles with density modification cycles in between. The autotracing starts with the location of possible seven-residue α-helices and common tripeptides. After extension of these fragments in both directions, various

criteria are used to decide whether these fragments, representing a poly-alanine trace, should be accepted or rejected. In case there is noncrystallographic symmetry (NCS), it can be applied in to the traced fragments, not to the density [90].
The protein structure is indicated as solved, when there is a clear difference in the contrast of the two enantiomers and the correlation coefficient between $F_{calc}$ from the polyalanine trace and the $F_{obs}$ of the native data ($CC_{partial}$) is higher than 25% [100].

### 2.3.4 SITCOM

SITCOM is a program to compare substructures calculated by different programs and can be used for analysing the SHELXD output. In the scope of this work, it is used to compare the best substructure sites of SHELXD. If the distance between one site s in SHELXD and another site r in the refined model does not exceed a specific value, the sites are matching [20]. For matches, the

$$rmsd = \sqrt{\sum_{i}^{N} \frac{(s_i - r_i)^2}{N}} \tag{2.52}$$

is calculated as a measure of how well the two compared substructures agree. SITCOM is run from the command line.

### 2.3.5 ANODE

The program ANODE has been designed to calculate and analyse anomalous or heavy-atom density by reversing the usual procedure for experimental phase determination. The heavy-atom phase $\phi_A$ is calculated by subtracting the phase shift $\alpha$ obtained by SHELXC from the total phase $\phi_T$, which is calculated from a refined protein structure. The heavy-atom density map, also referred to as an anomalous difference map, is calculated by fast Fourier transform. From the map in the case of SAD, the square root of the variance of the electron density $\sigma(\rho)$ can be derived. ANODE is command line based and returns i.a.

- the heights and coordinates of the unique peaks in the map and their distances from the nearest atom in the refined protein structure, taking space-group symmetry and unit-cell translations into account,

- the map coefficients so that the anomalous peaks with their density can be displayed,

- an output file in SHELXD format which can be used to solve the structure with SHELXE.

By this means, it is possible to locate even weak anomalous scatterers like sulphur [99].

### 2.3.6 ARP/WARP, REFMAC5 and COOT

When the final electron density map has been calculated, it needs to be interpreted by building a macromolecular model in agreement with experimental and stereochemical knowledge. In the extent of this work, the programs ARP/WARP, REFMAC5 and COOT have been used.
ARP/WARP is a program for automated model building. Based on the amino acid sequence of the protein, atoms are added or removed until an initial hybrid model consisting of a partial molecular model and free atoms of undefined chemical identity has been built. The hybrid model is refined in REFMAC5 where the model parameters are adjusted in a way that the experimental data and a priori stereochemical expectations are matched in a better way. The cycle of adding atoms to the model and refining them is repeated several times, until it converges in the case of success to the final macromolecular model, with the remaining free atoms approximating the surrounding solvent structure. ARP/WARP is limited by the available resolution and the quality of the phases [65].
The already mentioned program REFMAC5 is used to REFine MACromolecular models by adjusting the model parameters (coordinates, B-factors, TLS etc) in order to obtain the model which best explains the experimental data. REFMAC5target function for maximization upon refinement is the log-likelihood. This is a concept of Bayesian statistics, where chemical/geometrical restraints are taken as prior knowledge and the 'likelihood' is the posterior probability according to Bayes' theorem. Quality indicators are the R- and the $R_{free}$ factor (compare section data quality). Progress is measured by R-factor and Free R-factor, as well as by the likelihood scores themselves [102, 109].
COOT is a graphical application for model building and validation of biological macromolecules. The program displays electron-density maps and atomic models and allows a variety of model manipulations [30]. As automated model building is only successful to a certain degree, missing residues have to be modelled manually and the agreement of model and electron density regarding stereochemistry for instance, have to be verified by visual inspection. After each model building and (optionally) real-space refinement step in COOT, the model is refined in REFMAC5 to correct for possible mistakes and to monitor the progress.

## 2.4 Data quality

In macromolecular crystallography, a variety of parameters have been developed to judge data quality in nearly every step from data collection to the final model. In the context of data measurement, there is often the discussion how precise and how accurate the measurements are. Precise measurements refer to data points very close together, accurate measurements refer to data points measured as close as possible to a true value. Precision is limited by random errors, while accuracy is limited by systematic errors [86].

### 2.4.1 What is the resolution of the data?

In the first step after initial processing of the diffraction images, for instance with XDS, one has to judge up to which resolution the data can reliably be used so that more signal than noise is added. To do so, several parameters have been introduced. For one, there is the I/σ value in the highest resolution shell measured. The resolution shells are determined automatically by XDS and the intensity I is calculated as the mean of intensity of the unique reflections in this resolution shell, after merging symmetry-related reflections. $\sigma(I)$ is the standard deviation of this intensity. There are opinions considering data with an I/σ value below 2.0 as too weak to be used for the further data evaluation [32].
Another value for deciding where to cut the data is the correlation coefficient between the intensities of two random halves of a data set ($CC_{1/2}$). Each data pair ($x_i$, $y_i$) used to calculate a linear Pearson correlation coefficient represents one unique reflection; where $x_i$ and $y_i$ are the averaged $\bar{I}$ after creating two subgroups of $\frac{N}{2}$ symmetry-equivalent observations by random picking. It has been observed that including data with a $CC_{1/2}$ of $0.1 - 0.2$ in the highest resolution shell still leads to an improvement of the final model. However, this resolution limit implies that data are included which would be excluded by any other cut-off parameter [27].
Apart from these factors, the completeness of the data set is of great importance. Any missing reflection leads to a deterioration of the model parameters [54]. Therefore, the 'effective' resolution can be defined based on the nominal resolution and the cube root of the completeness of the data set [106].

### 2.4.2 How good are the data?

A measure for internal consistency of unmerged data, in the past also used for the determination of the resolution cut-off, is the $R_{meas}$-value. It measures how well the different reflections agree, taking into account how many times n a unique reflection is measured:

$$R_{meas} = \frac{\sum_{hkl} \sqrt{\frac{n}{n-1}} \sum_{i=1}^{n} |I_i(hkl) - \bar{I}(hkl)|}{\sum_{hkl} \sum_{i=1}^{n} I_i(hkl)}. \tag{2.53}$$

Here, $\bar{I}$ represents the average intensity of the n equivalent reflections. The variable n is also known as redundancy or multiplicity.
In the course of structure determination and refinement, symmetry equivalent reflections are merged. The precision-indicating merging value $R_{p.i.m.}$ is a measure of the quality of the data after averaging the multiple measurements:

$$R_{p.i.m} = \frac{\sum_{hkl} \sqrt{\frac{1}{N-1}} \sum_i |I_i(hkl) - \bar{I}(hkl)|}{\sum_{hkl} \sum_i I_i(hkl)}. \tag{2.54}$$

As it can be seen from the formula, $R_{meas}$ is multiplicity-independent, while $R_{p.i.m.}$ is not [106].
For assessing the reliability of a refined model compared to the experimental data, the reliability index, better known as R-factor, is used:

$$R = \frac{\sum_{hkl} |F_{obs(hkl)} - F_{calc(hkl)}|}{\sum_{hkl} F_{obs}(hkl)}. \tag{2.55}$$

An R-factor of 0 would indicate a perfect agreement of the structure factors calculated from the refined model and the measured structure factor, which would be 0.59 for a random model [107]. However, it has been shown that the R-value can become quite good due to overfitting. Therefore it became common to omit a small percentage of the data (usually 5 %) in the modelling process. These data are then used to calculate an R-value, which is named $R_{free}$ [15]. The R-value, which is calculated based on the rest of the data, is technically referred to as $R_{work}$ [61], but in most tables just named R-value. Because the $R_{free}$-value is not adapted in a refinement process, it is typically higher than the R-value by a factor of 1.2 [10]. An even higher $R_{free}$ factor would

indicate overfitting.

# Chapter 3

# Materials and Methods

Even though experimental phasing has been used for more than 35 years, not all questions have been answered yet. With the availibility of fast detectors and cheap data storage, it recently became popular to use high redundancy, low dose data sets for sulphur SAD phasing [34, 68, 105]. From this practise, one question arises. Despite a low-dose strategy, there will be radiation damage, especially to the disulphide bonds. So it has to be found a criterium for the point where adding data will add more noise than signal and in this way influence the structure determination process negatively. All data for the two projects presented have been collected at the EMBL beamline P14 at PETRA III. Consequently, first of all the set-up of the beamline will be described, followed by a detailed description of the materials and methods used in the project. The chapter ends with a description of the self-written pipelines for data processing.

## 3.1 Experimental Apparatus

Photon science at DESY (short for Deutsches Elektronen-Synchrotron) started as a parasitically used by-product of particle physics. Nowadays, the storage ring PETRA III is nearly exclusively used for experiments with a wide spectrum of photon energies in the X-ray regime.

### 3.1.1 PETRA III storage ring

The third 'reincarnation' of the positron electron tandem ring accelerator (PETRA) mostly stores positrons which circle the 2304 m long ring with a velocity close to the speed of light. To achieve these velocities, a cascade of accelerators is used prior to the injection of the positrons or elec-

trons into the ring. First of all, electrons are generated by a thermoionic gun, then accelerated to 450 MeV in a linear accelerator (LINAC 2). Then, they are hitting a tungsten target, generating positrons and electrons. Before getting separated, the electrons and positrons are stored in a solenoid coil. Afterwards, the positrons are accelerated to 450 MeV in LINAC 2 and collected in a ring structure named positron intensity accumulator (PIA), forced in bunches with a defined time structure by a high-frequency system. From there, they are transferred to the PETRA III ring. Here, the particles can be stored in average for about 10 hours, when their energy loss arising from the emitted radiation is compensated by radio frequency cavities [6]. Using a current of 100 mA in combination with undulators in close spacing, the achieved emittance $\epsilon$ defined as the product of source size and source divergence is currently the worldwide smallest (compare Table 3.1), rendering a beam with low divergence at the experimental endstations of the beamlines, in a distance of up to 100 m from the insertion device in the ring.

| | |
|---|---|
| circumference [m] | 2304 |
| positron energy [GeV] | 6 |
| positron beam current [mA] | 100 |
| horizontal positron beam emittance (rmsd) [nmrad] | 1 |
| vertical positron beam emittance (rmsd) [nmrad] | 0.01 |

Table 3.1: Key parameters of the PETRA III storage ring as taken from http://photon-science.desy.de/facilities/petra_iii/machine/parameters/index_eng.html

### 3.1.2 The EMBL beamlines

The 300 m long experimental hall of the PETRA III hall holds 14 beamlines. Three of them are operated by EMBL, and two of them, *P*13 and *P*14 are designed for macromolecular crystallography experiments, while the third is used for small angle X-ray scattering experiments (*P*12). Both beamlines for macromolecular crystallography

- have a MAATEL *MD2* diffractometer including a mini-$\kappa$ goniometer,
- provide automated sample centring and 4D-scans,
- can quickly change their energy based on multi-segmented piezo-electric, i.e. adaptive bimorph focusing mirrors in Kirkpatrick-Baez geometry,

- use a *CryoJetXL* from Oxford Instruments for cooling the crystals to 100 K,
- are equipped for the recording of fluorescence spectra,
- include a *PILATUS6M* detector from Dectris operating with up to 25 Hz for shutter-less oscillation data collection,
- are equipped with a MARVIN sample changer system (in-house design),
- operate with the same interface which is an adapted version of the mxCuBE v2 (ESRF).

Despite these similarities, the two beamlines differ in their specifications. The main difference of *P*13 in comparison to *P*14 is the availibility of lower energies, lower photon flux, larger beam sizes and different beam shapes [1, 2]. In the following, the beamline layout of *P*14 is described in greater detail, as the experiments for this project were performed there.

#### 3.1.2.1 The beamline P14

P14 provides beam sizes ranging from 5 µm x 5 µm and a divergence below 0.3 mrad up to a maximum size of 300 µm with an unfocused beam. This is achieved by the set-up schematically shown in Fig. 3.1. After passing the undulator, the white beam can be focused by water-cooled compound refractive lenses (CRL) made of beryllium to a point approximately three meters behind the detector to enable large beam sizes with a top-hat profile and high flux density. These CRL became available in summer 2015, so that only some experiments could be performed with this set-up. Consequently, the user can easily and quickly move between an unfocused, half-focused and fully focused beam. A double silicium-(111)-monochromator selects the desired energy in the range of 6 to 20 keV, enabling data collection to subatomic resolution. The beam is then further focused with the adaptive bimorph focusing mirrors in Kirkpatrick-Baez geometry, also referred to as KB-mirrors. With a fully focused beam, the photon flux density becomes so high that the lifetime of the crystal in the beam is only 0.5 s. However, the flux can be decreased in a controlled manner by putting an attenuator in the beam. The one used at P14 consists of three wheels. Two of them contain aluminium pieces, the other titanium pieces of varying thickness. If the beam should be focused further, this can be achieved by using slits and apertures made of platinum. To reduce background scattering, a capillary made of molybdenum can be used. The shape and the structure of the beam can be controlled with a scintillator coated with bismuth germanium oxide [12]. The crystals are mounted on a magnetic pin which is then attached to

the mini-$\kappa$ goniostat. For measurements at cryo-temperatures, the cryojet is aligned in a way that the crystal is cooled to 100 K. Due to the small beam size available, the beamline is also suited for *in situ* data collection, serial crystallography [42] and large unit cell sizes of up to 800 Å [2]. Apart from that, SAD and since recently also MAD experiments can be performed routinely at this beamline.

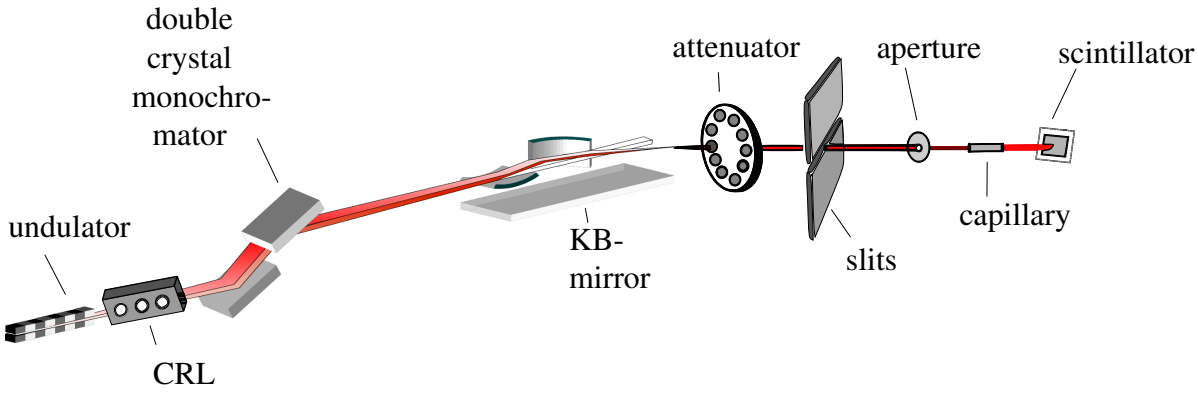



Figure 3.1: Schematic drawing of the set-up for focussing and monitoring the beam depicted in red at *P*14.

## 3.2 The SAD project

For finding the balance between radiation damage and multiplicity, crystals from three test systems have been investigated. This section will start with a short description of the test system thaumatin, its crystallization conditions and the data collection strategy. Afterwards, the structure determination will shortly be described.

### 3.2.1 The test systems thaumatin

The test system thaumatin was chosen, because the protein is commercially available as a powder and its crystallization is easily reproduceble. Besides, the protein contains cysteins and methionines and is therefore suitable for native sulphur SAD phasing.

Thaumatin is a intensely sweet protein is extracted from the seed vessel of the Katemfe plant (*Thaumatococcus daniellii*). As it is many times sweeter than sucrose it is used in food industry

as a sweetener. It consists of 207 amino acids building a monomer, including eight disulphide bonds and one methionine [33].

### 3.2.2 The crystallization conditions

48 mg/ml Thaumatin purchased from Sigma Aldrich was dissolved in 0.1 M Bis-Tris-Propane, pH 6.5. Crystals were grown by the hanging drop method mixing the protein solution in a 1 : 1 ratio with the well solution consisting of 0.1 M Bis-Tris-Propane and 0.6 − 1 M sodium tartrate. At room temperature, crystals with a typical size of 150 μm x 100 μm x 100 μm appeared within two days. The cryo-solution consisted of 0.6 M sodium tartrate, 0.1 M Bis-Tris- Propane and 25% glycerol.

### 3.2.3 Data collection

All data were collected at the EMBL beamline *P*14 described above. Before starting the data collection, the photon flux was measured. For doing so, the beam position is first checked with the scintillator and tuned, if necessary. Then, the scintillator and the capillary are moved out of the beam so that it hits the 50 μm thick silicon diode. It was cross-checked with diodes calibrated by the Physikalisch-Technische Bundesanstalt (PTB), revealing an accuracy of greater than 99% [12]. The diode is connected to an amplifier. Based on the measured current, the photon flux is calculated. Thereby, the measurement error of the flux is ±1%. During the measurement, the flux could only be measured with a diode behind the monochromator. Unfortunately, the amplifier of this diode which was available at the time of the measurements was quickly overloaded, so that this diode showed no change in flux, while measurements behind the aperture at the end of the data collection revealed a change of up to 47% when CRL were used. The reason for this change is a beam drift due to thermal imbalances, mostly effecting the vertical beam size [12].
For all measurements, the beam was collimated with the 150 μm aperture so that a top-hat beam profile was realized (compare Fig. 3.2). When no CRL were used, the beam was unfocused; with CRL, slits and the attenuator were used additionally. The energy was set to 8.01 keV, at which the Bijvoet ratios of the test system is 1.2% indicating that the structure can be solved based on sulphur SAD.
To realize a uniform illumination, only crystals which were smaller than the beam were chosen

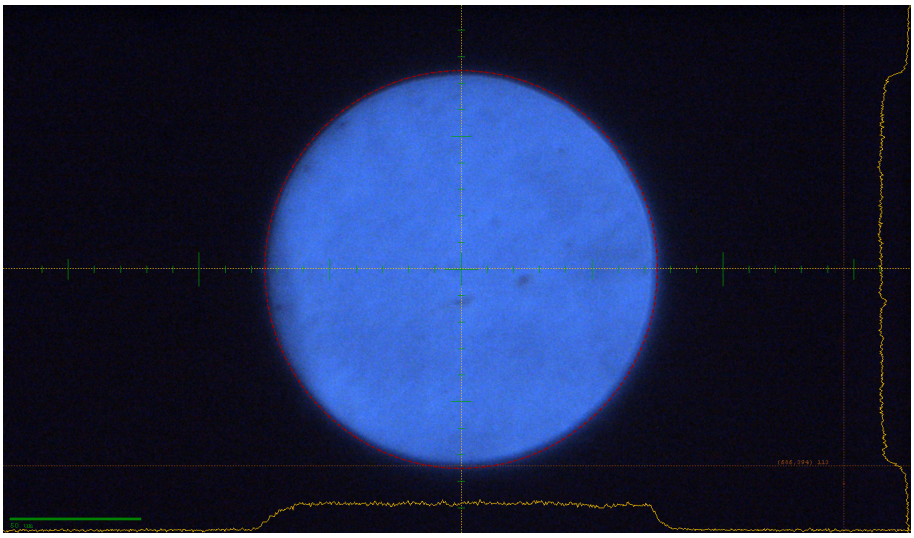


Figure 3.2: Beam with 150µm aperture as shown by the scintillator. The yellow lines at the bottom and the right show the top-hat intensity profile.

(compare Fig. 3.3). After immersing the not yet cryo-protected crystals shortly in the cryo-solution, the crystals were either rapidly transferred to the goniometer or previously frozen in liquid nitrogen and transferred to the goniometer during the beamtime. All measurements have been performed at 100 K to reduce radiation damage. Even though the data were collected with a PILATUS6M detector which can give better results with fine-sliced data [71], all diffraction data were collected with an oscillation range of 1°.

The reasons were threefold. For this project, it was first of all the most important task to collect data for answering the question how radiation damage and multiplicity can be balanced. This can be done based on the so collected data. Secondly, with a measurement strategy aiming for high multiplicity, large amounts of data have to be accumulated. Typically, a crystal was turned

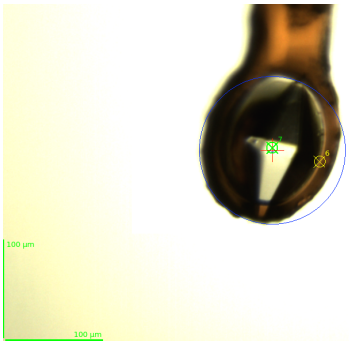


Figure 3.3: Thaumatin crystal smaller than the beam, which is indicated by a blue circle.

15x360°. With a size of 6.2 MB per diffraction image, the raw data collected from one crystal with an oscillation range of 1° already account for 33.48 GB. Despite the possibility to compress a raw diffraction image to 3.2 MB, a state-of-the art data collection with an oscillation range of 0.1° would have led to the tenfold amount of data. At last, one has to consider that beamtime is limited. With the use of CRL, data were partly collected with an exposure time of 0.1 s per diffraction image, allowing for the data collection of one 360° turn within 36 s. Assuming that the exposure time would be reduced to the technical minimum of 0.04 s, the total exposure time per turn would have increased by a factor of 4 if an oscillation range of 0.1 s would be applied. Crystals were turned as long as both diffraction images and XDS results indicated that the crystal was strongly affected by radiation damage. The data collection parameters can be found in Table 3.2. The average diffraction weighted dose calculated with RADDOSE-3D is based on the flux measured at the beginning of the measurements and is therefore rather over- than underestimated. For a top-hat beam profile, a round collimation is not an option within the program, but a rectangular one. Because of this, the square-root of the area of the 150 µm aperture (132.93 µm) was used. The crystal dimensions measured with the MD-software with an estimated measurement error of 5% in each dimension. The absorption coefficients were calculated with RADDOSE-3D based on the unit cell size, the heavy atoms in the protein, the solvent and the solvent composition. The error of the dose can only individually be estimated, based on the set-up and, if available, the change of the flux. Details will be discussed later.

### 3.2.4 Data processing and (sub)structure determination

Data were processed via the self-written Python pipeline *Process_all* (compare Appendix A). First of all, each 360° turn was processed separately with XDS. The automatically written XDS input files are adapted in a way that each turn has the same resolution cut-off, which corresponds to a sensible resolution cut-off for the first turn and which is normally the best achievable (1.7 Å at this wavelength). Obviously, also the file path of the raw data is changed, and Friedel's law is set to false. In the next step, the data of each turn are processed with SHELXC and SHELXD. For SHELXC, the correct space group and unit cell have to be given. The latter always corresponds to the unit cell calculated by XDS with the data of the first turn, as it could be shown that the changes of the unit cell parameters due to radiation damage do not significantly affect the SHELXC and SHELXD results (see chapter results). As SHELXD performs better based on a super-sulphur search in the cases presented here, the 'DSUL' keyword was used and the anomalous data were

| **data collection** | | | | | |
|---|---|---|---|---|---|
| crystal | 150421_tha1 | 150421_tha4 | 150421_tha6 | 150927_tha1 | 150927_tha3 |
| energy [keV] | 8.01 | 8.01 | 8.01 | 8.01 | 8.01 |
| flux [ph/s] | $5.5 \cdot 10^{10}$ | $5.5 \cdot 10^{10}$ | $5.5 \cdot 10^{10}$ | $5.1 \cdot 10^{11}$ | $5.1 \cdot 10^{11}$ |
| CRL | no | no | no | yes | yes |
| aperture [µm] | 150 | 150 | 150 | 150 | 150 |
| exposure [s/image] | 1 | 1 | 1 | 0.1 | 0.1 |
| oscillation range [°] | 1 | 1 | 1 | 1 | 1 |
| temperature [K] | 100 | 100 | 100 | 100 | 100 |
| crystal size [$\mu m^3$] | 100 x 50 x 55 | 110 x 60 x 40 | 145 x 90 x 90 | 130 x 64 x 52 | 152 x 78 x 60 |
| space group | $P4_12_12$ | $P4_12_12$ | $P4_12_12$ | $P4_12_12$ | $P4_12_12$ |
| unit cell parameter a=b [Å] | 57.82 | 57.71 | 57.83 | 57.86 | 57.74 |
| unit cell parameter c [Å] | c=150.52 | 150.17 | 150.17 | 150.18 | 150.45 |
| max. resolution [Å] | 1.9 | 1.9 | 1.75 | 1.7 | 1.7 |
| Bijvoet ratio [%] | 1.23 | 1.23 | 1.23 | 1.23 | 1.23 |
| solvent heavy atom conc. [M/l] | 1.2 Na | 1.2 Na | 1.2 Na | 1.5 Na | 1.8 Na |
| solvent fraction | 0.54 | 0.54 | 0.54 | 0.54 | 0.54 |
| dose [MGy/360°] | 0.69 | 0.69 | 0.67 | 0.65 | 0.63 |
| no. of turns | 15 | 15 | 20 | 15 | 10 |

Table 3.2: Parameters for thaumatin data collected with a top-hat profile at P14. The unit cell parameters refer to the data of the first turn. The dose refers to the average diffraction weighted dose calculated with RADDOSE-3D.

cut at 2.8 Å, as at this resolution the d"/σ value obtained from SHELXC falls below 1 when the first 360° are processed. Because SHELXD involves stochastic aspects when searching for the substructure, 10000 trials were chosen to make the solutions of the different turns comparable. However, solutions can also be obtained with much less trials. Also, the substructure was calculated with ANODE using the default parameters except for the B-value, which is used to damp the noisy $F_A$ data at high resolution. This value was set to 10, as the sulphur-SAD data are weak and therefore require a B-value slightly higher than the default of 8.
The reflections of subsequently recorded turns were merged one after the other by appending the XDS_ASCII.HKL files produced by XDS and adapting the image number. Common scaling programs like XSCALE or AIMLESS could not be used successfully, as it will be demonstrated in the next chapter. The accumulated data were then processed with SHELXC and SHELXD. Depending on the CFOM and the accumulated dose one solved substructure was chosen and the whole protein structure was solved with SHELXE. For this purpose, three rounds of autotracing and 20 rounds of density modification were applied together with site refinement. After that, the phases calculated by SHELXE stored in the .phs-file are converted with the CCP4 programs F2MTZ and FREERFLAG[18, 108] to an .mtz-file, where each reflection is tagged with a flag for cross-validation. The .mtz-file and the sequence in .pir format were then used by ARP/WARP to build a model. The resulting model was refined in REFMAC5 and COOT to a certain degree, where special care was taken to position the sulphur atoms correctly. Even though the cell constants of a previously available refinement model, e.g. from the PDB, were quite similar, it was necessary to obtain a correct reference model in this way for comparing it with the other substructures obtained by adding more and more data using SITCOM and to calculate the anomalous peak heights with ANODE. SITCOM analysed the 100 best substructures in terms of CFOM, the number of found sites and the rmsd to the refined reference structure within 3 Å. The results of the refined structure are discussed in the next chapter.

### 3.2.5 The data analysis and plotting pipelines

To analyse the collected data, two Python pipelines named *Extract_all* and *Plot_all* were written. The first one extracts the relevant information from the output of the programs. The latter plots the results from the above mentioned programs. Because the structure of the ANODE output files is rather complicated, as sometimes more than two anomalous peak heights are assigned to one residue, a separate pipeline named *Anode* was written. For the plots, the anomalous peak heights

assigned to one residue have been added, as the change of the total anomalous peak height is the value of interest. The pipelines can be found in appendix A.

# Chapter 4

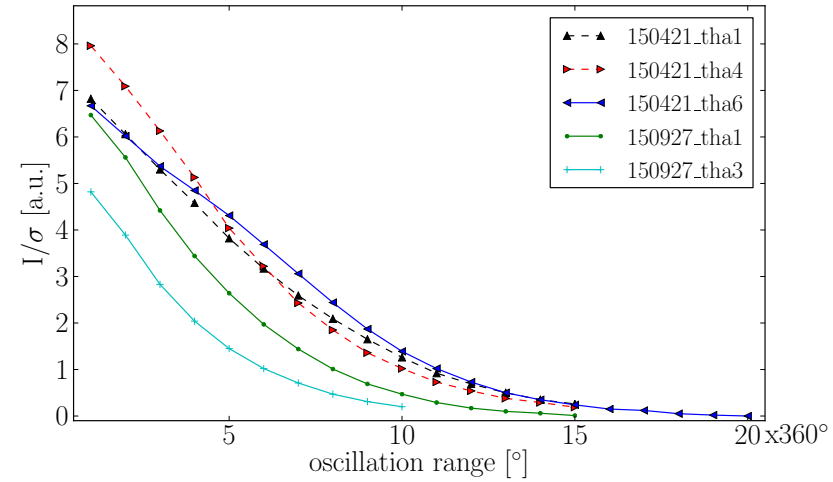
# Results and Discussion I - Case Studies

SAD experiments were performed on the test system thaumatin. In this chapter, the data quality and the comparability of the different data sets in the context of different experimental set-ups will be discussed as well as systematic influences such as systematic error arising from the instrument, the change of the isomorphism of the crystals, and the difficulties in scaling consecutively collected data sets. A detailed analysis of the substructures obtained in comparison with a refined reference model and the anomalous differences will be presented, concluding with a short summary of the chapter.

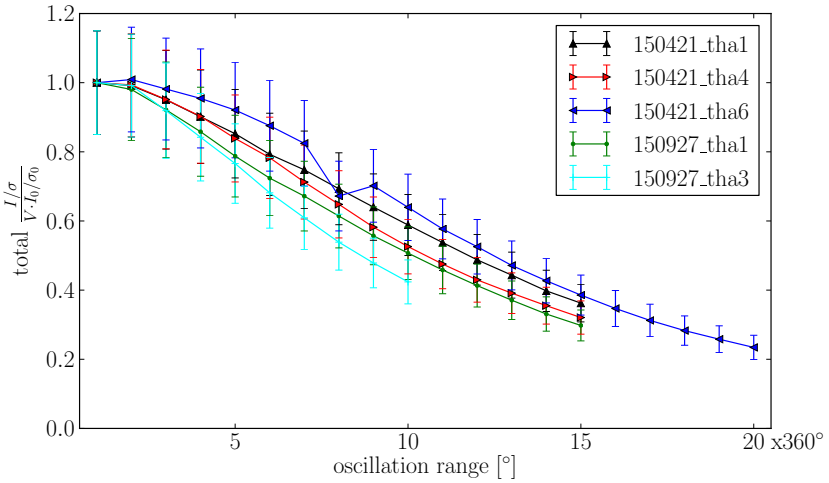
## 4.1 The comparability of measurements

The I/σ ratio in the highest resolution shell plotted versus the dose or respectively, the different wedges or 360° turns for all thaumatin data sets, show comparable curve characteristics (compare Fig. 4.1). The initial I/σ values for the highest resolution shell extracted from the statistic file CORRECT.LP produced by XDS range from 4.8 to 8.0, indicating well diffracting crystals. The difference in the initial value can have various reasons. Apart from the fact that the crystal volumes are different, identical crystals cannot be grown. In this case, the strongest difference is the solvent concentration, ranging from 1.2 – 1.8 M/l Na, while the mosaicity in the first turn is quite low for all data sets, ranging from 0.07° – 0.19°. Taking the volume of the crystal in form of a double pyramid and the different diffracting power of the single crystals into account by normalizing, it becomes clear that the overall signal-to-noise ratio, and with that all data sets are the same within experimental error (compare Fig. 4.1). For clarity, the detailed analysis and

discussion is thus limited to data set 150421_tha1. A detailed representation of the other data sets can be found in appendix B.




(a)




(b)

Figure 4.1: Plot of a) the I/σ ratio in the highest resolution shell ([1.7 – 1.8 Å] for data sets collected on 150927, [1.75 – 1.86 Å] for data set 150421_tha6 and [1.9 – 2.03 Å]) of the different wedges obtained by XDS and plotted versus the subsequent wedges/turns; and b) the overall I/σ values obtained by XDS normed to the volume of the corresponding crystal and the initial signal-to-noise ratio $I_0/\sigma_0$ plotted versus the dose.

## 4.2 Data quality

In the following, the curve progression of the different data quality parameters with the dose will be explained and discussed based on the specific case of data set 150421_tha1. The slight differences within the experimental error of the other data sets will be analysed in a separate section.

### 4.2.1 The specific case

To judge the data quality and to discuss different possible resolution cut-offs, the I/σ values obtained by merging all reflections in the highest resolution shell and the corresponding $CC_{1/2}$ as well as the overall $R_{meas}$ and the overall mean anomalous difference for the subsequently recorded wedges were plotted versus the dose. All these parameters were extracted from the statistic file CORRECT.LP produced by XDS. Therefore, the mean anomalous difference is normalized to its estimated standard deviation ($|F^+\text{-}F^-|/\sigma$), where $F^+$ and $F^-$ are structure factor amplitude estimates obtained from the merged intensity observations (referred to as SigAno in the CORRECT.LP file). As an example, the data collected from the thaumatin crystal 150421_tha1 should be considered, where the highest resolution shell comprises the range from 1.90 – 2.02 Å. Despite the fact that the subsequently recorded data sets were not put on a common scale, the results from the subsequently recorded data sets follow quite smooth curves. The $I/\sigma$ curve plotted versus the subsequently recorded 360° turns decays exponentially (compare Fig. 4.2 a), dropping below 2.0 after the eighth turn. In contrast, the $CC_{1/2}$ curve starts with 96.6% for the first turn and decreases only by 17.1% in the first seven turns, but then drops strongly by 40% within the next four turns, before it decreases less strongly again to 5% in the last turn. The $R_{meas}$-values increase exponentially (compare Fig. 4.2 c), while at the same time the mean anomalous differences decrease linearly from 1.0 to 0.75 (compare Fig. 4.2 d).
The decrease in the I/σ plot is due to the fading of the high resolution reflections. As the data collection parameters of subsequent turns were not varied, it is most likely that radiation damage is responsible for this effect. This is also reflected by the moderate decrease of the ISa value from 42.82 (a=1.060) for the first turn to 35.95 (a=1.068) for the last turn, indicating that radiation damage is responsible for this curve progression (compare 2.3.1). Theoretically, the individual scaling of the different wedges could also be responsible for varying signal-to-noise ratio. However, if so, the ISa values would not be expected to change to this extent.

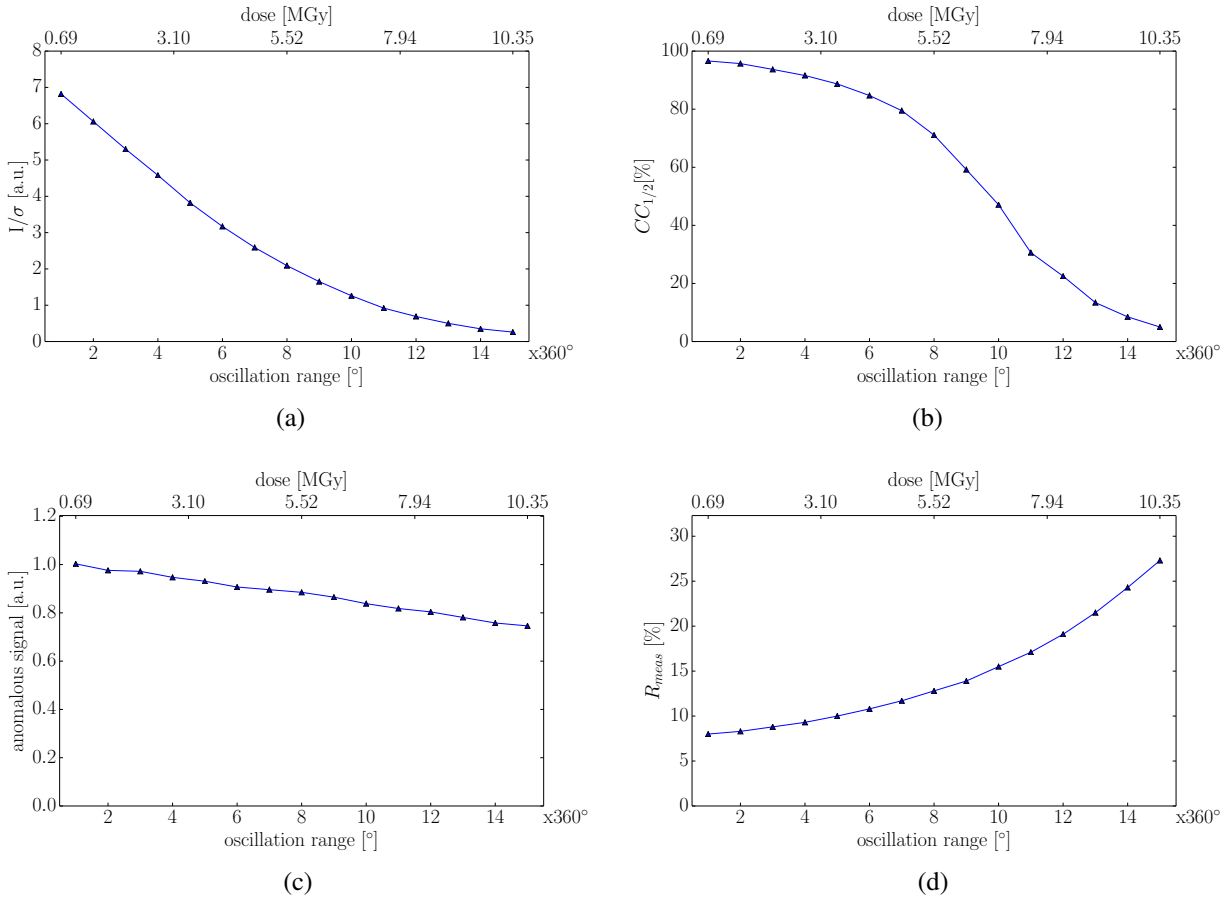



Figure 4.2: Plots of XDS results obtained by processing subsequently collected 360° wedges from thaumatin crystal 150421_tha1 with a) the merged I/σ values and b) the $CC_{1/2}$ values, both in the highest resolution shell [1.90 – 2.02 Å], c) the total anomalous signal and d) the total $R_{meas}$; all plotted versus the subsequently recorded 360° wedges and the dose, respectively.

At cryo-temperatures, an exponential decay of the intensity according to

$$\frac{I}{I_0}(\vec{k}, D) = e^{\frac{-B(D)k^2}{2}} = e^{\frac{-\beta D k^2}{2}} \tag{4.1}$$

is expected [13], where $I_0$ is the intensity at zero dose, B the Debye-Waller factor, β a constant scale factor representing the intensity-decay rate, D the dose, and $\vec{k}$ the wave vector. Even though the σ values are dependent on the background, the I/σ curves should not differ very strongly from the intensity curves, as the background should not change much.

With increasing radiation damage, the precision of the high resolution reflection decreases. According to Karplus & Diederichs [62], there is a link between the $CC_{1/2}$-value in the high resolu-

tion shell and the corresponding I/σ value:

$$CC_{1/2} \propto \frac{1}{1 + \frac{4}{(I/\sigma)^2}}. \tag{4.2}$$

This proportionality is shown in Fig. 4.3. The internal consistency of the data measured by the overall $R_{meas}$ value declines steadily, as the $R_{meas}$-values increase exponentially. This is due to the increase of the radiation-induced non-isomorphism, which will be discussed in the next section. Another explanation is that $R_{meas} \propto \frac{1}{I/\sigma}$.

Considering the resolution cut-off, there are various options. According to Wlodawer et al. [110], the data after the fifth turn provide sub-optimal data quality, as the overall $R_{meas}$ exceeds 10%. Based on the I/σ value, data from the highest resolution shell should not be included any more after the eighth round. In contrast, the $CC_{1/2}$ values indicate that there can still be valuable information in the last turn leading to an improvement of the electron density map, as far as this is not only valid for data sets from different crystals, but also from subsequently recorded data sets (compare chapter 2.4).

However, none of the above statistical parameters are directly correlated with the anomalous signal in an X-ray diffraction data set and can therefore neither indicate whether the data are good enough to solve the substructure, nor where they should ideally be cut [36]. For the former

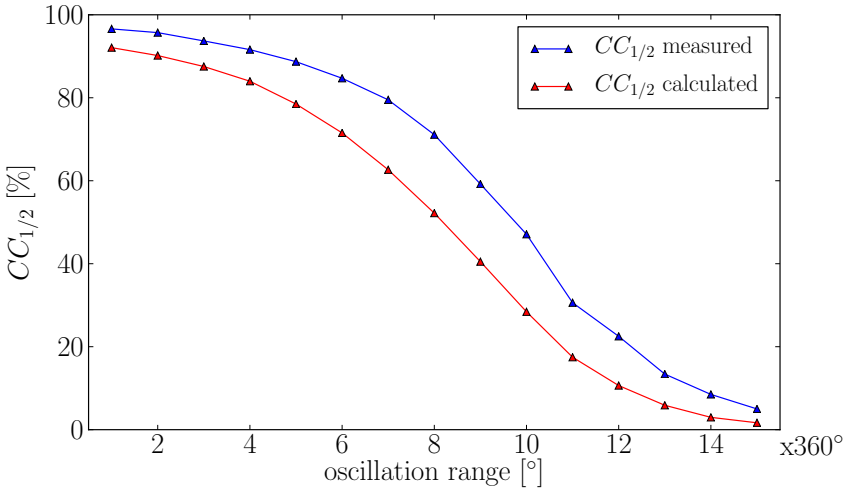



Figure 4.3: Plot of the $CC_{1/2}$ value in the highest resolution shell [1.90 – 2.02 Å] as obtained by XDS of the thaumatin crystal 20150421_tha1 (blue) as a function of turn number and the corresponding $CC_{1/2}$ values calculated with equation 4.2 based on the merged I/σ values, i.e. $\langle I \rangle / \sigma(\langle I \rangle)$, also obtained by XDS for this shell.

criterion, the mean anomalous difference in units of its estimated standard deviation is used. The overall mean anomalous difference should be 1.0 or greater for solving the substructure, indicating that only the data from the first 360° are of sufficient quality to do this. The linear decrease of the overall mean anomalous differences is probably due to the continuous decrease of the occupancy of the anomalous scatterers [8, 117]. Yet it can also be explained by the decrease of the average signal-to-noise ratio, so that a separation of global and local radiation damage based on this plot alone is difficult.

### 4.2.2 Analysis of the other data sets

To compare the thaumatin measurements from different crystals, the results from processing the data individually with XDS were plotted versus the dose (compare Fig. 4.1, 4.4, 4.5). The highest resolution shell for the $CC_{1/2}$ plot comprises 1.7 – 1.8 Å for the data sets 150927_tha1 and 1509271_tha3, 1.75 – 1.86 Å for data set 150421_tha6 and 1.9 – 2.03 Å for the data sets 150421_tha1 and 150421_tha3, where 1.7 Å was the resolution corresponding to the geometrical highest 2θ angle at the energy used for data collection. As the $CC_{1/2}$-values are connected to the I/σ values plotted in Fig. 4.1 a, the corresponding curves follow the same tendency. To be more precise, the 150927_tha1 and the 1509271_tha3 data have smaller I/σ and $CC_{1/2}$ values in the

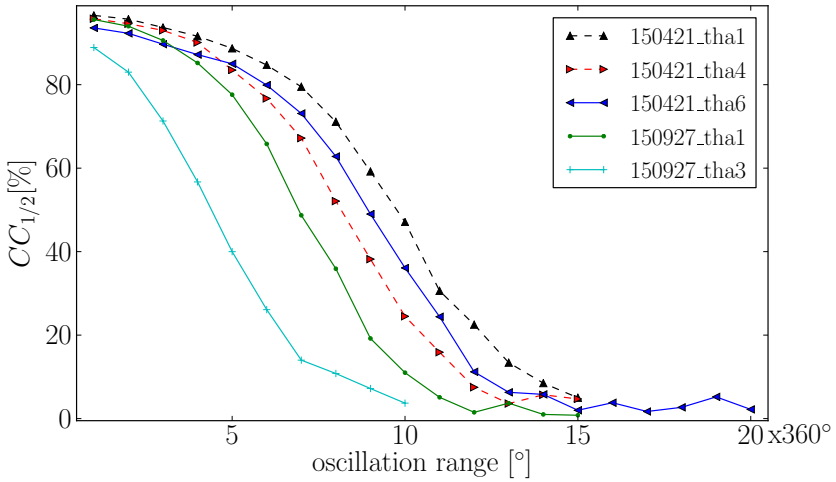
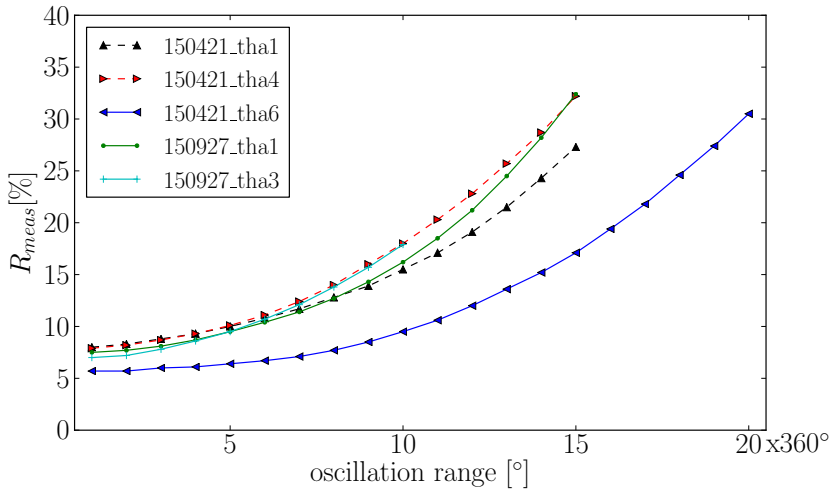
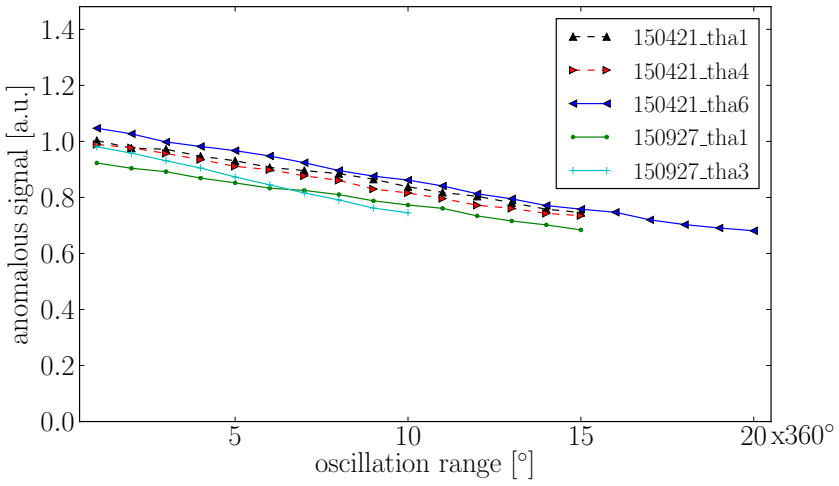



Figure 4.4: Plot of the $CC_{1/2}$ values in the highest resolution shells ([1.7 – 1.8 Å] for data sets collected on 150927, [1.75 – 1.86 Å] for data set 150421_tha6 and [1.9 – 2.03 Å]) against the dose, obtained by processing subsequently collected 360° wedges of all thaumatin data sets in XDS.




(a)




(b)

Figure 4.5: Plots of a) the total $R_{meas}$ and b) the total anomalous signal against the dose, obtained by processing subsequently collected 360° wedges from all thaumatin data sets with XDS.

highest resolution shell, and also the correlation of the random halves drops much more rapidly than the other curves (compare Fig. 4.1, green and cyan curve).

The internal consistency of the data measured decreases, as $R_{meas}$ increases exponentially for all data sets. However, data set 150421_tha6 shows the $R_{meas}$-values are lower from the beginning and also increase less strongly (compare Fig. 4.5 a, blue curve). Because the parameters with which all data sets collected on 150421 were not varied, it can be concluded that this crystal was the best diffracting one. This is not only due to the fact that this crystal has the highest volume,

as it can be seen from Fig. 4.5 b where the total I/σ values were normalized to the volume of the individual crystal and the I/σ value of the first 360° wedge. The bump in 150421_tha6 in the eighth turn is due to a processing problem with XDS which could not be resolved. Apart from that, most curves are the same within experimental error.
The differences in the I/σ and the $CC_{1/2}$ plots for the data set collected on 150927 are likely to be explained by two major effects. Firstly, if data are collected at the geometrical highest 2θ angle, the XDS algorithm does not perform background scaling as well as at smaller 2θ angles. As the data collected at 150927 are at the technically minimum detector distance, they are affected more than the data sets 150421_tha1 and 150421_tha3, as it can be seen in Fig. 4.6. Secondly, the different data collection set-ups with and without CRLs play a role, as the beam tends to drift at the energy used for data collection until a thermal equilibrium is reached. When CRLs were used and the equilibrium is not reached, the beam is much smaller and drifting leads to a decreased flux without the user being able to measure or notice this at the given time of the measurements. This also affects the background [12].

## 4.3 Systematic influences on the data

In principle, the data are effected by systematic errors, including radiation damage. The latter depends on the dose, and the accuracy with which it could be determined will be discussed in the following as well as its effects on the isomorphism of the crystal. The attempt to deal with some

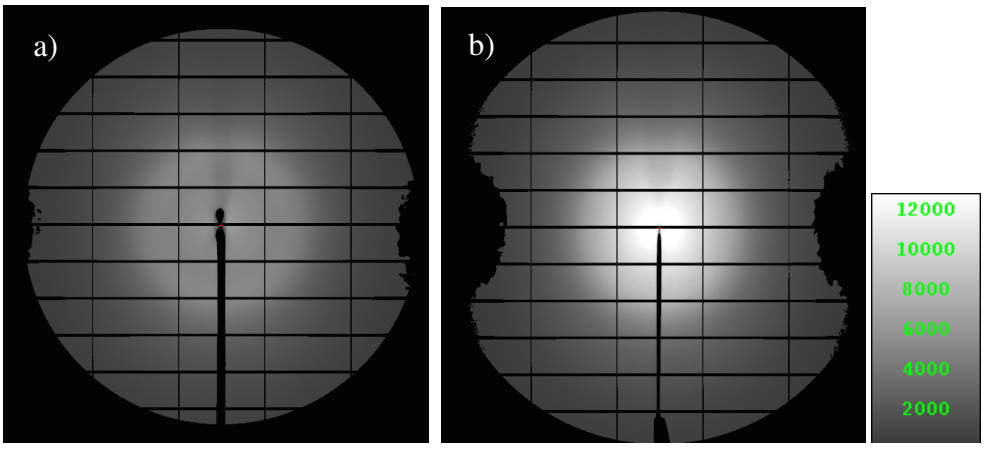



Figure 4.6: Inverted background images of the first turn of a) 150421_tha1 and b)150927_tha3, generated by XDS and pictured with ADXV on the same scale.

of these effects by scaling will be analysed as well.

### 4.3.1 The systematic error

The systematic error of the data is composed of three parts: the instrumental error, the unspecific and the specific radiation damage. The instrumental error can be estimated based on the value 1/ISa (compare section 2.3.1) [62]. The ISa values including the a and b values and the resulting error for the first 360° turn can be found in Table 4.1. The unspecific damage can be calculated with the program BEST [13]. The non-isomorphism introduced due to the non-specific damage in low resolution shells is approximately 5% in the first 360° turn, assuming an average diffraction weighted dose of 0.7 MGy. The specific damage cannot be quantified by BEST [13].
As it can be seen from Table 4.1, the instrumental errors of the data collected at 150421 are about the same, where this is not true for the data collected at 150927. As there are no other obvious error sources, it is likely that this in general greater error and the inconsistency are due to beam instabilities.

| data set | ISa | a | b | 1/ISa |
|---|---|---|---|---|
| 150421_tha1 | 42.82 | 1.07 | $5.10 \cdot 10^{-4}$ | 0.02 |
| 150421_tha4 | 42.79 | 1.07 | $5.09 \cdot 10^{-4}$ | 0.02 |
| 150421_tha6 | 41.66 | 1.16 | $4.97 \cdot 10^{-4}$ | 0.02 |
| 150927_tha1 | 21.96 | 1.19 | $1.76 \cdot 10^{-3}$ | 0.05 |
| 150927_tha3 | 36.76 | 1.12 | $6.64 \cdot 10^{-3}$ | 0.03 |

Table 4.1: ISa and the corresponding a and b values from the thaumatin data sets as obtained from the CORRECT.LP files produced by XDS. The value 1/ISa is an estimate of the systematic error in the data set that limits the precision of strong reflections [62].

### 4.3.2 The dose estimate

The average diffraction weighted dose calculated with RADDOSE-3D is very similar for all data sets (compare Table 4.2). The average diffraction weighted dose of the most exposed crystal is only by 0.06 MGy higher than the least exposed crystal. However, the dose estimate is defective due to the inaccuracy of the diode ($\Delta$ d), the measurement error ($\Delta$ m), the limitation, that RADDOSE-3D cannot work with circular apertures and the change of the flux during the measurement.

Adding $\Delta$ d and $\Delta$ m results in an error of $\sqrt{2}\%$ in the flux. Using RADDOSE-3D, the resulting error for the dose could be quantified as approximately $\sqrt{2}\%$, so that a direct proportionality

| | 150421 | | | 150927 | |
|---|---|---|---|---|---|
| data set | tha1 | tha4 | tha6 | tha1 | tha3 |
| av. diffraction weighted dose [MGy/360°] | 0.69 | 0.69 | 0.67 | 0.65 | 0.63 |

Table 4.2: Average diffraction weighted dose accumulated within 360° for the single data sets calculated with RADDOSE-3D.

is assumed. Very recently, a member of the Garman group made it possible to use the beam profile measured with the scintillator by enabling the use of a non-released version of RADDOSE-3D. The global error of using a rectangular instead of a circular collimation is about 1.4%. Apart from these errors, there is an insecurity in the dose due to beam drift, as it can be seen from the difference in flux measured at the beginning and the end of the data collection of all data sets. For the data collected at 150421, the flux decreased by 6.1% ± 4% within eleven hours. Assuming a linear process and taking the 20 turns of 150421_tha6 within two hours of measurement into account, the flux and with that the dose decreased by 1.1% ± $\sqrt{2}$% from the first to the last frame of this data set. However, as the flux is not measured inbetween or during the data sets, the dose was calculated based on the initially measured flux and the changes are considered in the error estimate. To calculate the error of the dose in percent for a certain point $t_{meas}$ after the begin of the measurement, it follows:

$$\Delta D(t) = \sqrt{2 + 1.4^2 + (0.55 \cdot t_{meas})^2} \tag{4.3}$$

for data collected on 150421. For the data collected with CRLs, the error sums up to

$$\Delta D(t) = \sqrt{2 + 1.4^2 + (5.57 \cdot t_{meas})^2}, \tag{4.4}$$

as the beam drift is much higher due to the CRL, so that the flux decreased by 47.4% ± 4% over the total beamtime of approximately 8.5 hours. Although the initial dose value might be too high for data collected at 150927, the decrease in flux from the first to the last frame within fifteen turns is only 0.8% ± $\sqrt{2}$% due to the shorter exposure.

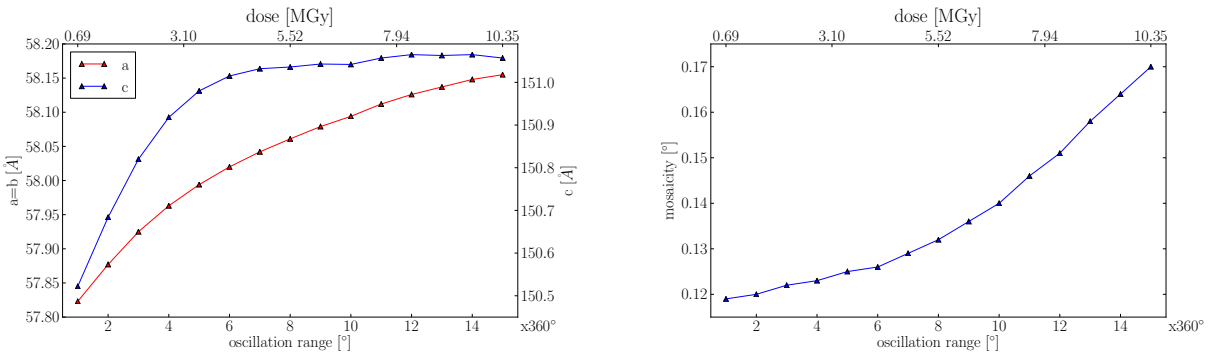
### 4.3.3 Radiation-induced non-isomorphism

As mentioned before, radiation damage is introducing non-isomorphism into the sample. This can be seen from the change in unit cell constants and in the mosaicity of the crystal (compare

Fig. 4.7), here shown by way of example for 150421_tha1. The trend for the change in the unit cell constants is as expected [8, 67]. Cell constant a increased by 0.57 %, while cell constant c increased by 0.69%, leading to a total volume change of 1.49% within the 15x360° turns.
The reason for this expansion is presumed to be due to the accumulation of an electrostatic potential within the crystal and to the build-up of an internal pressure due to decarboxylation [8, 84]. Consequently, the mosaicity also increases. However, using crystals smaller than the beam leading to an average signal over the whole crystal, it cannot be discriminated to which extent this is due to the increase of the unit cell constants or whether this happens on account of the angular distribution of mosaic blocks might be responsible [74].

### 4.3.4 Scaling

If data sets are processed separately, it is conventional to put them on a common scale, as a processing program would not refine all parameters in exactly the same way. In this case, where the different data sets are consecutively collected, it is especially interesting to see how the program compensates for radiation damage. Consequently, XSCALE was used to perform the scaling, leaving the symmetry related reflections unmerged as SHELXC enables subsequent statistics.
Fig. 4.8 shows the scale factors K and B with which the intensities of the 15x360° turns of data set 150421_tha1 are scaled (compare section 2.3.2). The increase of the K- and the B-values is expected, as the overall intensity decreases due to radiation damage. However, a closer look at the average intensity of Bijvoet positive and Bijvoet negative reflections, which should be put on the same scale via scaling, reveals that the performed scaling is not as useful as expected. Fig.




(a) (b)

Figure 4.7: Change of a) the unit cell parameters and b) the mosaicity as obtained by XDS.

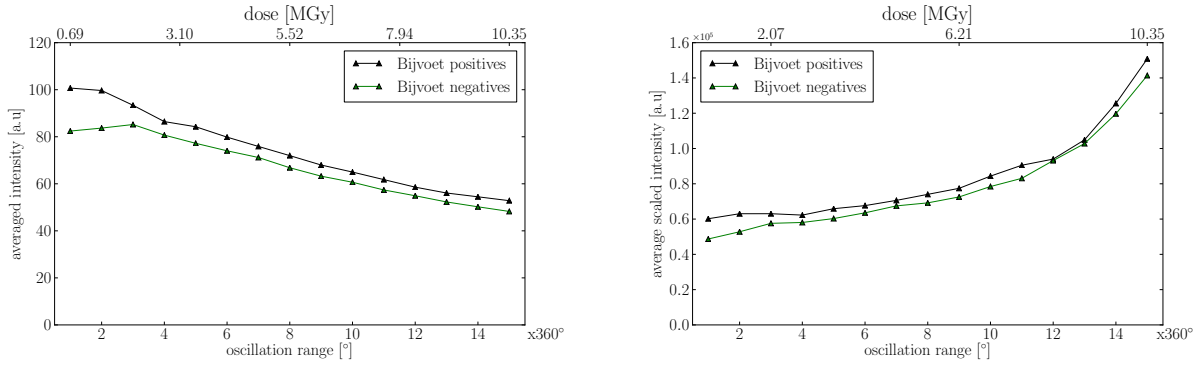
```
K*EXP(B*SS) = Factor applied to intensities
   SS       = (2sin(theta)/lambda)^2

    K        B          DATA SET NAME
 1.0000   0.000    /data/storm/20150421/tha1/PROCESSED_DATA/xds_tha_1_02_1_1//XDS_ASCII.HKL
 1.0009   0.464    /data/storm/20150421/tha1/PROCESSED_DATA/xds_tha_1_02_2_1//XDS_ASCII.HKL
 1.0010   1.012    /data/storm/20150421/tha1/PROCESSED_DATA/xds_tha_1_02_3_1//XDS_ASCII.HKL
 0.9938   1.612    /data/storm/20150421/tha1/PROCESSED_DATA/xds_tha_1_02_4_1//XDS_ASCII.HKL
 1.0093   2.266    /data/storm/20150421/tha1/PROCESSED_DATA/xds_tha_1_02_5_1//XDS_ASCII.HKL
 1.0279   2.983    /data/storm/20150421/tha1/PROCESSED_DATA/xds_tha_1_02_6_1//XDS_ASCII.HKL
 1.0250   3.764    /data/storm/20150421/tha1/PROCESSED_DATA/xds_tha_1_02_7_1//XDS_ASCII.HKL
 1.0260   4.630    /data/storm/20150421/tha1/PROCESSED_DATA/xds_tha_1_02_8_1//XDS_ASCII.HKL
 1.0200   5.573    /data/storm/20150421/tha1/PROCESSED_DATA/xds_tha_1_02_9_1//XDS_ASCII.HKL
 1.0018   6.625    /data/storm/20150421/tha1/PROCESSED_DATA/xds_tha_1_02_10_1//XDS_ASCII.HKL
 0.9822   7.743    /data/storm/20150421/tha1/PROCESSED_DATA/xds_tha_1_02_11_1//XDS_ASCII.HKL
 0.9620   8.934    /data/storm/20150421/tha1/PROCESSED_DATA/xds_tha_1_02_12_1//XDS_ASCII.HKL
 0.9486  10.169    /data/storm/20150421/tha1/PROCESSED_DATA/xds_tha_1_02_13_1//XDS_ASCII.HKL
 0.9661  11.461    /data/storm/20150421/tha1/PROCESSED_DATA/xds_tha_1_02_14_1//XDS_ASCII.HKL
 0.9355  12.814    /data/storm/20150421/tha1/PROCESSED_DATA/xds_tha_1_02_15_1//XDS_ASCII.HKL
```

Figure 4.8: Scale parameters K and B for scaling the subsequently recorded data sets of 150421_tha1.

4.9 a shows the unscaled averaged and accumulated intensities for the reflection 25 15 10, which corresponds to a resolution of 1.97 Å, and Fig. 4.9 b shows the same scaled intensities. While the accumulated, unscaled intensities decrease and establish a constant difference between the Bijvoet positive and negative reflections after the first four turns, the scaled averaged intensities increase approximately exponentially. Also, the difference between the Bijvoet positive and negative reflections vary, indicating a varying anomalous difference. As the intensities of single reflections can both increase and decrease in subsequent measurements, these plots are only of limited informative value, even though they are based on averaged intensities. Nevertheless, random checks of reflections in different resolution shells revealed a similar behaviour of other reflections recorded at high resolution.

Significantly worse results for the substructure solution based on scaled data show that this is indeed not a problem of single reflections being incorrectly scaled (compare Fig. 4.10). A possible explanation for the incorrect scaling is that the σ(I) values are not entirely correct for already quite damaged data sets, as the reflection profiles might deviate from a Gaussian distribution with increasing mosaicity. This situation might be aggravated by the fact that nearly all reflections are fully recorded instead of fine-sliced, leading to a higher background and hampering profile fitting [71]. The fact that a zero-dose extrapolation does not lead to any improvement supports this theory. The main reason for the scaling to fail is probably the assumption implemented in XSCALE that the most accurate reflections are recorded in the middle of the measurement [12]. Obviously, this is not true when already quite damaged data are included.

The attempt to use AIMLESS as a scaling program failed as well, but due to other reasons. The pro-




(a) (b)

Figure 4.9: a) Averaged and b) accumulated intensities of the symmetry equivalents of the unique reflection 25 15 10 corresponding to a resolution of 1.97 Å, splitted in Bijvoet positive and Bijvoet negative reflections.

gram either failed to perform the scaling with many data sets, or it took very long, i.e. more than six hours to scale fifteen data sets together. On account of that, this method is not applicable at the beamline to decide whether more data sets from the same crystal should be collected or not. However, the scaling within the different wedges performed by XDS seems consistent enough, as it could for example be shown in Fig. 4.1 a.

## 4.4 The refined structure solution

The structure of data set 150421_tha1 was solved as described in section 3.2.4, based on the data collected in two 360° turns, as they were leading to significantly higher CFOM values than the first 360° turn by itself and were therefore facilitating the structure solution as it can also be seen from the high $CC_{partial}$ value of 48.21% (compare Table 4.3). The R-value of the final model obtained by using COOT and REFMAC5 was 16.22%, the $R_{free}$ value was 18.16%, i.e. the data are not overfitted.

For the other data sets, the number of 360° turns was chosen based on the same reasons, leading to comparable results in most processing steps (compare Table 4.3). Only the number of residues built by SHELXE is lower for the data sets collected on 150927, possibly because these data sets are not as strong as the others (compare section 4.2.2). However, ARP/WARP seems to be able to compensate for this. The precision-indicating merging R-factors $R_{p.i.m.}$ are quite low for all data sets, indicating a high precision of the averaged measurements [106].

| **refinement** | | | | | |
|---|---|---|---|---|---|
| crystal | 150421_tha1 | 150421_tha4 | 150421_tha6 | 150927_tha1 | 150927_tha3 |
| data [°] | 2x360 | 3x360 | 5x360 | 2x360 | 1x360 |
| mosaicity [°] | 0.12 | 0.16 | 0.16 | 0.16 | 0.19 |
| average anomalous multiplicity | 25.5 | 38.4 | 56.5 | 22.6 | 11.6 |
| completeness [%] | 99.9 | 98.9 | 99.0 | 100 | 99.1 |
| anomalous completeness [%] | 99.9 | 99.2 | 99.9 | 98.4 | 99.1 |
| residues built (SHELXE) | 203 (207) | 191 (207) | 195 (207) | 192 (207) | 182 (207) |
| $CC_{partial}$ (SHELXE) [%] | 48.21 | 43.89 | 45.27 | 42.72 | 40.26 |
| residues built (ARP/WARP) | 203 (207) | 205 (207) | 204 (207) | 201 (207) | 205 (207) |
| $R_{work}$ [%] | 16.22 | 16.47 | 15.83 | 16.08 | 15.71 |
| $R_{free}$ [%] | 18.16 | 19.35 | 18.76 | 18.53 | 18.56 |
| $R_{p.i.m}$ (all $I^+$ & $I^-$) [%] | 1.3 | 1.1 | 1.2 | 1.2 | 1.5 |

Table 4.3: Data quality and structure determination parameters of the five collected thaumatin data sets.

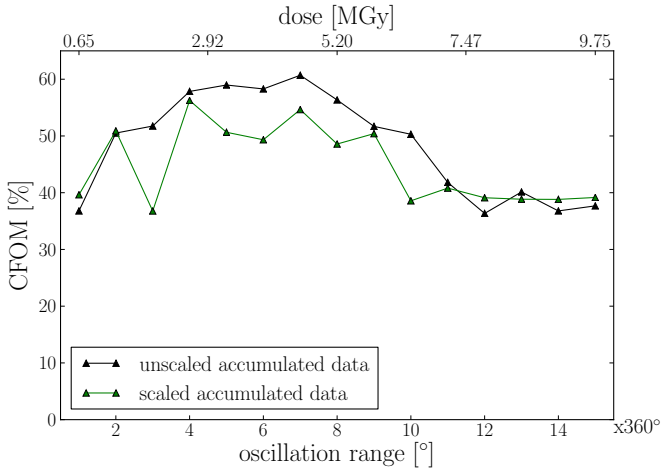



Figure 4.10: CFOM of the unscaled accumulated data and the ones scaled with XSCALE of data set 150421_tha1 obtained with the same processing parameters in SHELXD plotted versus subsequent turns.

## 4.5 Substructure solution

In the following, the substructures of the data processed in wedges and the accumulated data will be solved and then further analysed with the aim to determine a best substructure. For this, they are compared to a refined reference structure in SITCOM. Specific radiation damage will be assessed by analysing the results produced by ANODE.

### 4.5.1 Data preparation

As explained in section 3.2.4, the data collected in wedges and the accumulated data are first processed in SHELXC and then cut at 2.8 Å for further processing in SHELXD. To get an overview on how the anomalous signal behaves for the data processed in wedges and for the successively accumulated data, the d”/σ value of data set 150421_tha1 is plotted for the resolution shell 2.61 – 2.90 Å which is set by SHELXC (compare. Fig. 4.11).
From now on, blue curves are used for data processed in wedges and red curves for the accumulated data, where for instance 3x360° in the plot for the accumulated data are based on the data of the first three turns.
Fig. 4.11 shows that the d”/σ value drops directly after the first 360° turn for the data processed in wedges and that it even falls below 0.8 after the 10th 360° turn, indicating that the anomalous signal even in this resolution shell is not sufficient any more. In contrast, the values for the accu-

mulated data increase up to the point where four data sets were accumulated, and only then start to decrease, reflecting that the noise is increasing only then in comparison to the signal (compare section 2.3.3.1).

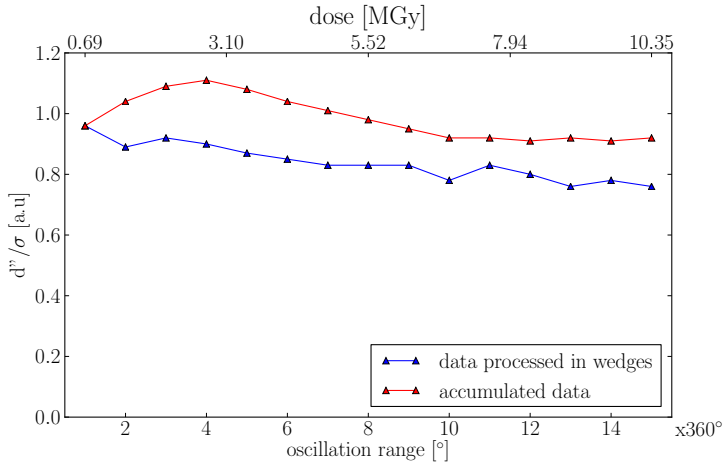



Figure 4.11: The d"/σ value in the resolution shell 2.61 – 2.90 Å as obtained by processing data set 150421_tha1 in SHELXC plotted versus the number of subsequently measured wedges for data processed in wedges and sequentially.

### 4.5.2 Substructure solution

To investigate the substructure quality of both the data evaluated in 360° wedges and the accumulated data, the corresponding CFOM values obtained by SHELXD were plotted for 150421_tha1 (compare Fig. 4.12 a and b). While for the data processed in wedges, only the first 360° lead to a CFOM significantly greater than 30% and with that in this case to a substructure solution, the CFOM values of the accumulated data increase up to the seventh turn, before it decreases, implying that the substructure is solved for data including up to the eleventh 360° turn. After that, there is no group of correlation coefficients separated from the other values, even though the CFOM is higher than the initial CFOM which indicates a solution.

### 4.5.3 Substructure validation

The obtained substructures will be validated in the following with the programs SITCOM and ANODE by using the refined structures as references.

#### 4.5.3.1 SITCOM

To judge the quality of the substructure, the substructure sites were compared to the ones of a reference structure using SITCOM. This reference structure was obtained by solving the structure based on 720° of data (compare section 4.5, Table 4.3). The number of identified sulphur sites and the rmsd as calculated by SITCOM were plotted for the data processed in wedges and for the accumulated data (compare Fig. 4.12 c and d). In the best case, all seventeen sulphur sites in the substructure of thaumatin would agree with the ones from the reference model. However, this is rarely the case at this resolution. In the following, a substructure is still described as (partially) correct, if not all , but the majority of sites are found and there is a group of solutions well separated from the rest.

For the first 360°, fourteen of seventeen sulphur sites are correct, indicating that the substructure is indeed solved and mainly correct. This is not the case for the following turns processed in wedges. In contrast, there is a maximum of sixteen sites found for the data accumulated within the first four turns, where the missing site is very likely the one from methionine which is hard to localize because its high B-factor indicates a variable conformation. The number of sites decreases to twelve for data accumulated within eleven turns and goes down to nine and less sites afterwards. Accordingly, there is a correlation between a (partially correct) substructure solution in SHELXD and the number of found sites in SITCOM. In this case, the substructure is only solved if twelve sulphur sites are correct as there is a group of solutions well separated from the rest (compare section 2.3.3.2). The best substructure solution is the one with the highest CFOM and the maximum number of found sites with the smallest rmsd.

The plots for the other data sets can be found in appendix B. Unfortunately, it is not always that clear which substructure solution is the best one. For data set 150927_tha3, the highest CFOM (82.09%) is reached with the data of four turns, but one site less is found than for the three previous data sets (compare Fig. B.4). As the rmsd from one and three turns are quite comparable, but the CFOM is higher by 13.58% for the data of three turns, one could argue that this is the best substructure solution. However, this weighting does not work for data set 150421_tha4, where the highest CFOM (74.74%) was obtained with the data of twelve 360° turns, while most sites are found with data from two and seven 360° turns (compare Fig. B.1). The data of two 360° turns have a considerably lower rmsd but also a lower CFOM. In this single case, it cannot clearly be said which substructure solution is the best. Considering the other data sets and the curve progression of the different parameters discussed in section 4.2.2, it is likely

| | 150421 | | | 150927 | |
|---|---|---|---|---|---|
| data set | tha1 | tha4 | tha6 | tha1 | tha3 |
| best substructure / 360° turns | 4 | 2-7 | 7 | 4 | 3 |

Table 4.4: The different data sets and the number of sequential 360° turns to get the best substructure as far as it could be determined.

that the best substructure solution should also be achieved after three or four 360° turns.

Table 4.4 sums up the results. Of particular note is here that the best substructure is not always obtained with the same dose, despite the fact that the dose per 360° turn is quite comparable for all data sets. Especially data set 150421_tha6 with seven instead of three or four turns stands out. However, if one considers that this data set is considerably better diffracting than the others (compare section 4.2.2), this can be understood. Taking the change of the intensity $I_{hkl}$ and its variance $\sigma^2_{hkl}(I_{hkl})$, from now on referred to as I and $\sigma^2$ , into account [26]:

$$\sigma^2 = \sigma^2_{counting} + KI^2. \tag{4.5}$$

$\sigma_{counting}$ represents the variance from Poissonian counting statistics including the background term and K corresponds to proportional errors such as random intensity fluctuations and detector efficiency variances and is therefore connected to the ISa value, i.e. mostly independent of whether the crystal is diffracting weakly or strongly. Dividing by $I^2$ leads to

$$\left(\frac{\sigma}{I}\right)^2 = \left(\frac{\sigma_{counting}}{I}\right)^2 + K, \tag{4.6}$$

where $\frac{\sigma}{I} \propto R_{meas}$. At low doses, K can be seen as a constant and I is much larger than $\sigma_{counting}$, so that the $(\frac{\sigma_{counting}}{I})^2$ term can be neglected. Increasing the dose has a decrease of intensity as a consequence, while $\sigma_{counting}$ stays approximately the same. At some point, $\sigma_{counting}$ becomes comparable to K and $\frac{\sigma}{I}$ starts to increase. For weaker data, this increase can be observed at lower doses, as the intensities are lower from the start [12]. Contrarily, the diffraction spots of stronger data such as 150421_tha6 fade at higher doses and higher multiplicity can be achieved.

#### 4.5.3.2 ANODE

In other sulphur SAD experiments, the anomalous peak heights determined with ANODE were used to decide whether data should be added or not [75, 105]. On account of that, ANODE was also

used here, using the complete refined structure as a reference (compare Fig. 4.12 e for the data processed in wedges and Fig. 4.12 f for the accumulated data). Bunkóczi et al. discovered that most substructure sites are found with an average anomalous peak height greater than 9.4 [16]. In the first 360° turn of data set 150421_tha1, fifteen sulphur sites were found, i.e. one more than identified as correct by SITCOM. For the data processed in wedges, all anomalous peak heights decrease with dose. It is remarkable that some anomalous peaks are not found in all data sets. e.g. the anomalous peak close to cystein residue 177 shows up in the first turn, disappears in the second and shows up again in the third 360° turn for the data processed in wedges. The explanation for this behaviour is that the cut-off for the distance between the found sites and the reference sites is by default set to 1 Å. If a single sulphur site then moves so much that it exceeds this value, it is not considered any more and the anomalous peak height is set to zero. Therefore, the residues showing a peak height alternating between zero and other values can be considered as especially affected by radiation damage, as for example cystein residue 177 which is known to be part of one of the two disulphide bonds the most susceptible to radiation damage [8].
Regarding the average anomalous peak heights of the accumulated data, it can be said that the anomalous peak heights increase for all but for the cystein residue 177, however up to which turn is very different from residue to residue, showing again that the behaviour of disulphide bonds towards radiation damage cannot be generalized. The majority of the average anomalous peak heights of the accumulated data exceeds the 9.4 σ value even in the fifteenth turn, in comparison to the four sites found by SITCOM. This is due to the fact that much weaker anomalous scattering leads to anomalous peaks, as the phases are known.

Apart from the individual anomalous peak heights close to sulphur-containing residues in a references structure, the averaged anomalous density calculated by ANODE could also be analysed. Table 4.5 shows the number of 360° turns required to get the maximum average anomalous density for the cysteins and the methionine residues. As the anomalous density is an average of 14 cysteins behaving quite differently, it is not surprising that the values are lower than for the values obtained for the one methionine residue. In comparison to the best substructure analysed in Table 4.4, there is no agreement, which is not surprising considering how differently SHELXD SITCOM and ANODE operate.

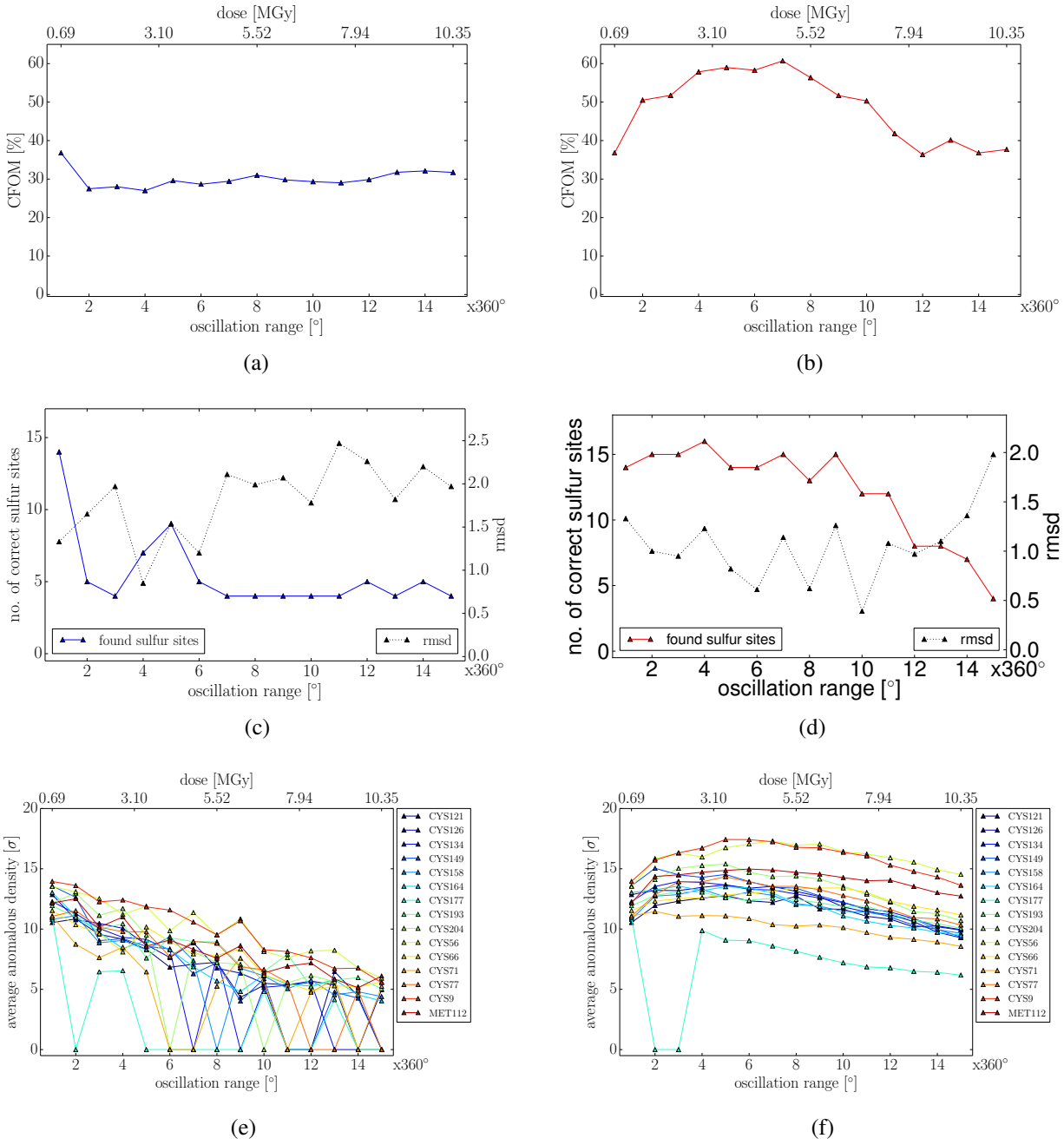



Figure 4.12: Plots of a) the CFOM obtained by SHELXD via processing the data in 360° wedges, b) the CFOM obtained by SHELXD via processing the accumulated data, c) the number of correct heavy atom sites and the corresponding rmsd with respect to the reference structure obtained by SITCOM via processing the data in wedges, d) the number of correct heavy atom sites and the corresponding rmsd to the reference structure obtained by SITCOM via processing the accumulated unscaled data. Fig. 4.12 e) shows the anomalous peak heights of the sulphur atoms calculated with anode for the data processed in wedges and f) the same for the accumulated data. All plots are based on data set 150421_tha1.

| | 150421 | | | 150927 | |
|---|---|---|---|---|---|
| data set | tha1 | tha4 | tha6 | tha1 | tha3 |
| max. av. anom. density (CYS) / 360° turns | 5 | 4 | 4 | 3 | 3 |
| max. av. anom. density (MET) / 360° turns | 6 | 7 | 6 | 4 | 5 |

Table 4.5: The different data sets and the number of 360° turns to get the maximum averaged anomalous density from ANODE for the cysteins and the methionine residues.

## 4.6 Specific radiation damage

Apart from calculating the anomalous peak heights, ANODE also generates anomalous differences maps, which can be used to depict the anomalous density around the anomalous scatterers. These maps were now contoured for data set 150421_tha1 at the 4σ level and displayed together with the refined model in COOT, with a zoom on the disulphide bridges which are the most (CYS126-CYS177) and the least susceptible (CYS9-CYS204) to specific radiation damage [8]. Additionally, the methionine and another peak, which is very likely a sodium ion in the solvent, can be found in this map section. For the data processed in 360° wedges, the maps of the first, fifth, tenth and fifteenth were depicted (compare Fig. 4.13). Fig. 4.13 a shows that initially, the disulphide bridges are not affected by radiation damage yet, as it can be seen from the well separated density around the corresponding two sulphur atoms building the individual disulphide bridge. This is already different in the fifth wedge, where the disulphide bridge CYS9-CYS204 is elongated and the peak around the sulphur atom of residue CYS177 seems to disappear. Also, the density around the methionine and the sodium ion seems to be reduced (compare Fig. 4.13 b). In the tenth wedge, the anomalous density does not change very much for the disulphide bridge CYS9-CYS204, and the one around the disulphide bridge CYS126-CYS177 is smeared out. In addition, the sodium ion seemed to have lost most of its density (compare Fig. 4.13 c). In the last turn, it looks like the sulphur atoms in residue CYS177 and CYS204 have completely lost their anomalous density, while the second bonding partner still show some. The sodium ion is no longer visible but instead, a new peak appeared, which corresponds to the sulphur atom in CYS145 displayed in this map at a slightly different perspective.

The different susceptibility of cystein residues and the elongation of disulphide bonds are well known responses of disulphide bonds caused by radiation damage [8, 83]. disulphide bonds are already radicalised at average doses of 5 kGy [94] and detoriation of the disulphide bonds can become visible at average doses of 1.2 MGy [78]. As the average diffraction weighted dose used in this thesis is approximately smaller by a factor of two in comparison to the average dose, it

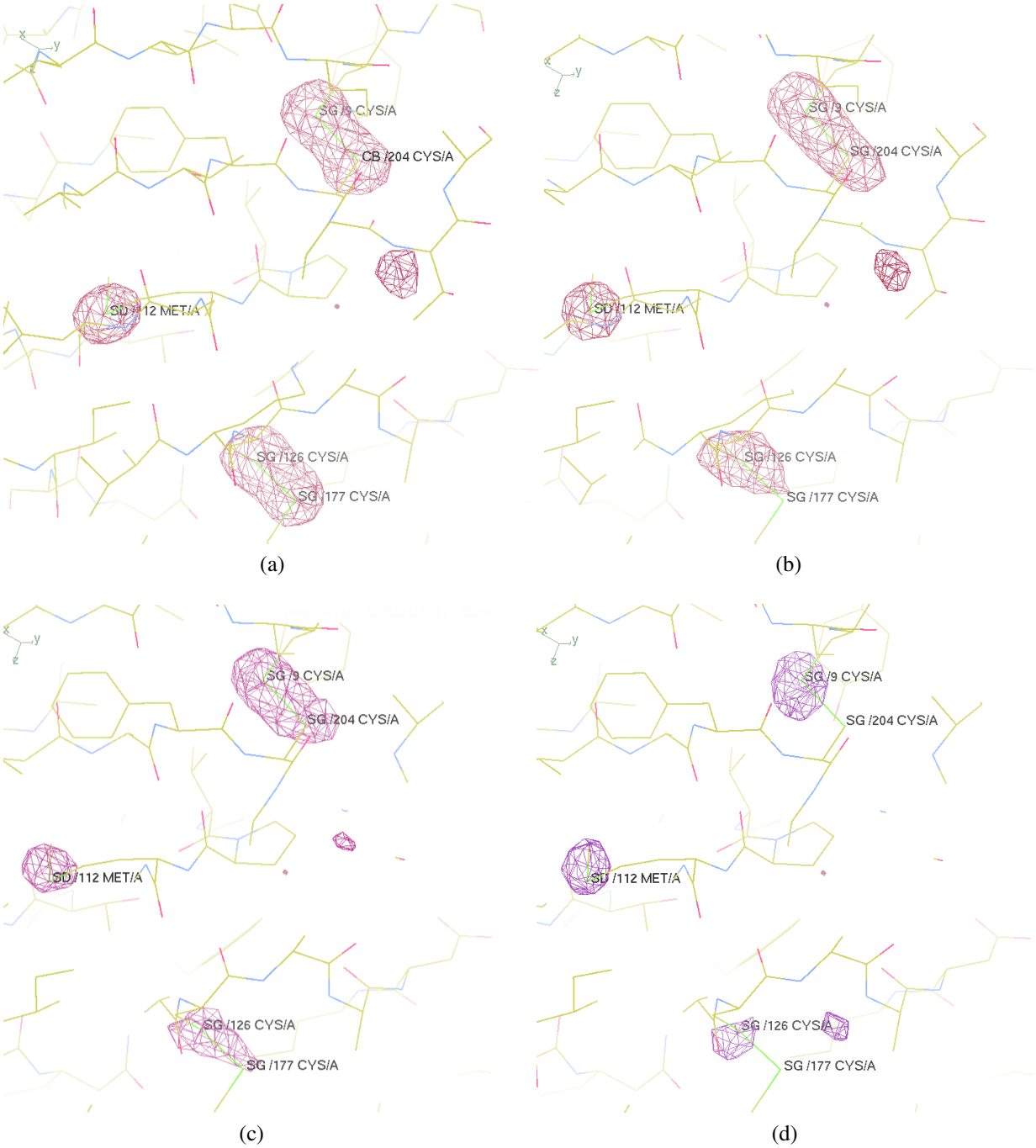



Figure 4.13: Anomalous difference maps ($\Delta F_{obs}, \varphi_{calc} - 90°$) calculated by ANODE of the sulphur atoms within the two disulphide bridges CYS126-CYS177 and CYS9-CYS204 and the one in methionine for a) the first 360° wedge, b) the fifth 360° wedge, the tenth 360° wedge and d) the fifteenth 360° wedge of data set 150421_tha1. The additional peak in the right part of the map is probably a sodium ion. The map was contoured at the 4 σ level and displayed with coot.

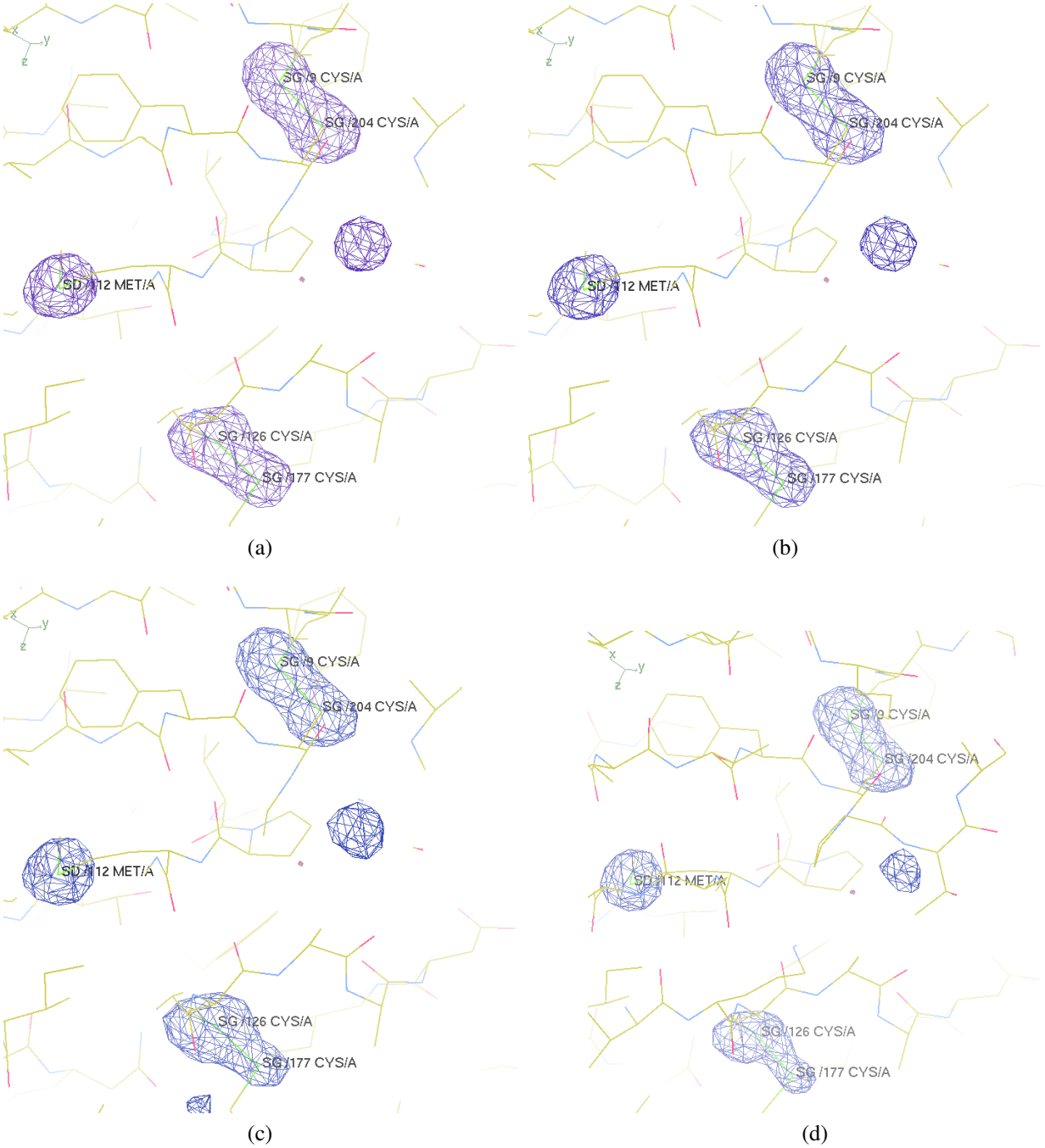


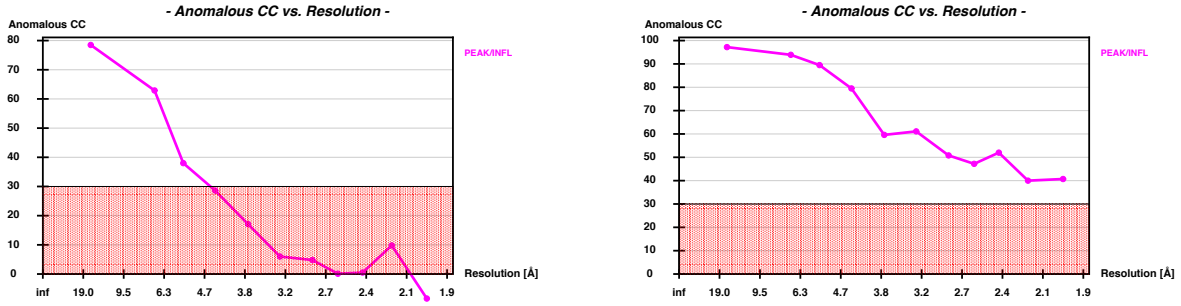

Figure 4.14: Anomalous difference maps ($\Delta F_{obs}, \varphi_{calc} - 90°$) calculated by ANODE of the sulphur atoms within the two disulphide bridges CYS126-CYS177 and CYS9-CYS204 and the one in methionine for a) four 360° wedges, b) five 360° wedges, ten 360° wedges and d) fifteen 360° wedges of data set 150421_tha1. The additional peak in the right part of the map is probably a sodium ion. The map was contoured at the 4 σ level and displayed with coot.

is likely that also the disulphide bonds in Fig. 4.13 a are already distorted in the first wedge. However, this is not visible due to the limited resolution. At near atomic resolution and higher doses, it becomes visible that cysteins start to rotate and build new conformations after a certain dose has been absorbed [78]. Additionally, one has to keep in mind that if a broken bond, a single ion or a new conformation should become visible in the electron density map, it must have happened reproducibly in a significant number of unit cells [64]. If this is not the case, it seems like atoms would disappear from the anomalous difference map, like the sodium ion seemed to be disintegrated.

Interestingly, the effect of specific radiation damage is much less visible in the anomalous differences maps generated for the accumulated data. Fig. 4.14 shows the maps with data from four turns which are used to calculate the best substructure; Fig. 4.14 b-d are based on data of five, ten and fifteen turns. The only clear difference can be observed in the disulphide bridge CYS126-CYS177 which slightly changes its shape. However, one has to keep in mind that these maps are based on averages of the data leading to the maps shown previously and that the known phases have a large impact.

To get an estimate of how well the average signed anomalous differences from distinct data subsets agree, the correlation coefficients for different resolution shells were calculated by using a pseudo-MAD approach in SHELXC. This was done for data set 150421_tha1 between the first and the last 360° wedge or alternatively, the accumulated data from all turns. The results were visualized with HKL2MAP and depicted in Fig. 4.15. For the wedges, the correlation coefficient is only about 80% in the lowest resolution shell, and it falls below the critical 30% line at a resolution of 4.7 Å. In the highest resolution shell, the anomalous differences are even slightly anti-correlated. With respect to Fig. 4.13, this is as expected, as the total anomalous signal goes down and also the anomalous difference maps show more differences than similarities.

However, the average signed anomalous differences in the lowest resolution shell of the first and all accumulated data sets are nearly perfectly correlated (compare Fig. 4.15 b). Even in the highest resolution shell there is still a correlation of 40%. This can only partially be explained by the fact that the first data set is obviously a part of all accumulated data sets, as its contribution is not much more essential than the following data sets, where the radiation damage firstly affects the high resolution reflections. Another explanation might be that all parts of the crystal were evenly illuminated, so that all reflections were affected in the same way by radiation damage. Unfortunately, this question cannot clearly be answered, as it is not known in detail how SHELXC calculates the average signed anomalous differences.




(a) (b)

Figure 4.15: Correlation coefficient of the average signed anomalous differences in different resolution shells calculated between a) the first and the last 360° wedge and b) between the first wedge and the data accumulated within fifteen 360° turns, both for data set 150421_tha1. The calculations were performed with SHELXC and the graphs were visualized with HKL2MAP.

## 4.7 Summary

It could be shown that the measured data are of high quality and that the corresponding parameters follow an expected course even without scaling. Differences between data sets collected with and without CRL could be explained by incorrect background scaling of XDS and by a varying flux. The latter was considered in the calculation of the dose error and also had an influence on the systematic error, which is generally quite low. The substructures were solved for the data collected in wedges and for the accumulated ones, showing that the d"/σ and the CFOM value are only reliable indicators for the solubility of the substructure if wedges are processed and that an enhanced multiplicity is beneficial. For determining the best substructure, the CFOM is used in combination with the number of correct sites and the corresponding rmsd calculated with SITCOM, which generally indicates a clear solution. As well-diffracting crystals such as 150421_tha6 possess a higher signal-to-noise ratio, higher doses can be tolerated and a higher multiplicity can be achieved. The anomalous peak heights calculated by ANODE are in disagreement with the SITCOM determined best substructure. The analysis of the anomalous difference maps of the wedges shows a strong effect of radiation damage on the disulphide bonds and the sodium ion in the solvent, what is hardly visible in the anomalous difference maps based on the accumulated data. Global radiation damage as monitored by the different data quality indicators of the wedges revealed that the specific damage is indeed happening much earlier than global radiation damage. Consequently, the suggested resolution cut-offs based on I/σ, $R_{meas}$ and $CC_{1/2}$ include data from too many wedges, as it could be shown for the deeply analysed data set 150421_tha1.

# Chapter 5

# Results and Discussion II - a Metric for Assessing Anomalous Data Quality

In this chapter, a metric to assess the quality of the anomalous data will be presented with the aim to define the optimal amount of data to get the best substructure. The metric is based on the **average signed anomalous differences**, which will be calculated, analysed and used to show the connection to the best substructure determined in the previous chapter. To clarify the specific calculations and to verify that the code works correctly, the implementation within the developed Python program will be described and analysed by means of comparing the results to established programs.

## 5.1 A program for calculating and analysing anomalous data

The program for analysing anomalous differences was written in Python and consists of five classes which are controlled by a main program. The corresponding code can be found in appendix C.

The class *input_data* processes the XDS_ASCII.HKL files, i.e. the reflections are read and eventually stored in dictionaries with different reflection categories (compare section 2.1.2.2). As the symmetry equivalents are point group dependent, the program was written as a prototype for four space groups, i.e. $P4_32_12$, $P4_12_12$, $I2_13$ and $P2_12_12_1$, where $P4_32_12$ and $P4_12_12$ share the same symmetry equivalents (method *spacegroup*). The extension to all point groups has to be done in the future.

The resolution of the reflections is calculated based on the unit cell parameters given by XDS

(method *resolution*), which are extracted from the header of the XDS_ASCII.HKL file (method *extract*). Additionally, all reflections with their intensities I and the corresponding σ(I) including the z variable (i.e. the image number) are selected. For all reflections, the resolution is calculated. To exclude outliers which are included into the XDS_ASCII.HKL file and which are marked with a negative σ(*I*) [58], only reflections with σ(I) > 0 are written to dictionaries (method *make_HKL_dic*). In the last step, the reflections in the afore mentioned dictionaries are assigned to the unique reflections which are calculated by the CCP4 program unique (method *unique*) [18, 108]. With the unique reflections as keys of the new dictionaries, the reflections are sorted in Bijvoet positive, Bijvoet negative, centric and systematic absent reflections (method *Bijvoet_Pairs*).

The dictionary with the Bijvoet pairs is then rendered to the class *anomalous_differences*. Here, anomalous differences are calculated in several different ways. Because it was the only one leading to conclusive results, the most important for the later analysis is the method *ano_dif1*, which calculates the so-called **average signed anomalous differences**, from now on referred to as ⟨ΔF⟩ values. For calculating them, the intensities of all Bijvoet positives $I_p$ and all Bijvoet negatives $I_n$ greater zero are averaged. From the averaged values, the ⟨ΔF⟩ values of a unique reflection hkl is calculated:

$$\langle \Delta F \rangle = \sqrt{\frac{1}{n}\sum_{i}^{n} I_{pi}} - \sqrt{\frac{1}{m}\sum_{j}^{m} I_{nj}} \text{ for } I_{pi}, I_{nj} > 0. \tag{5.1}$$

Even though it is known that neglecting negative intensities can result in bias [35], they are omitted here for several technical reasons. The ⟨ΔF⟩ values can be calculated for a defined resolution and image range.

The class *analysing* analyses the anomalous differences and generates statistics of the Bijvoet pairs. For the ⟨ΔF⟩ values, it generates a histogram with 100 bins. The interval of the histogram depend on the distribution of the ⟨ΔF⟩ values and have to be chosen by the user. As signed anomalous differences are used here, the histogram is statistically expected to be symmetric, as the assignment of Bijvoet positive and Bijvoet negative reflections does not imply any statement regarding the intensities. Therefore, the symmetric histogram can be fitted with a Gaussian distribution. Optionally, the histogram is normalized and fitted to a normal distribution. For both options, the fitting errors as well as the statistical errors are calculated (method *analyse_Ano_Dif1*).

To get an estimate of how well a fit agrees with the histogram, the mean square deviation can be calculated with the method *compare_fit_with_histogram*. The change of one average signed

anomalous difference within a certain image interval can as well be analysed (method *reflection_analysis*) as intensity statistics and histograms regarding the multiplicity of Bijvoet positive and Bijvoet negative reflections (method *Bijvoets*).
To be able to merge reflection files, to extract the relevant information from the output of the different programs and to write them in an adequate manner to a pdf file, the class *helping_routines* was written.
The results are further processed in the class *plotting*. Within the most important methods,

- anomalous differences of two data sets can be plotted against each other in a scatterplot (method *scatterplot*),
- the corresponding correlation coefficient of the anomalous differences from two data sets can be depicted for different resolution shells (method *CC_vs_res*),
- the anomalous differences can be plotted versus the resolution (method *plot_ano_dif1_vs_res*),
- the histograms including fits can be calculated and displayed (method *plot_AnoDif1_hist*),

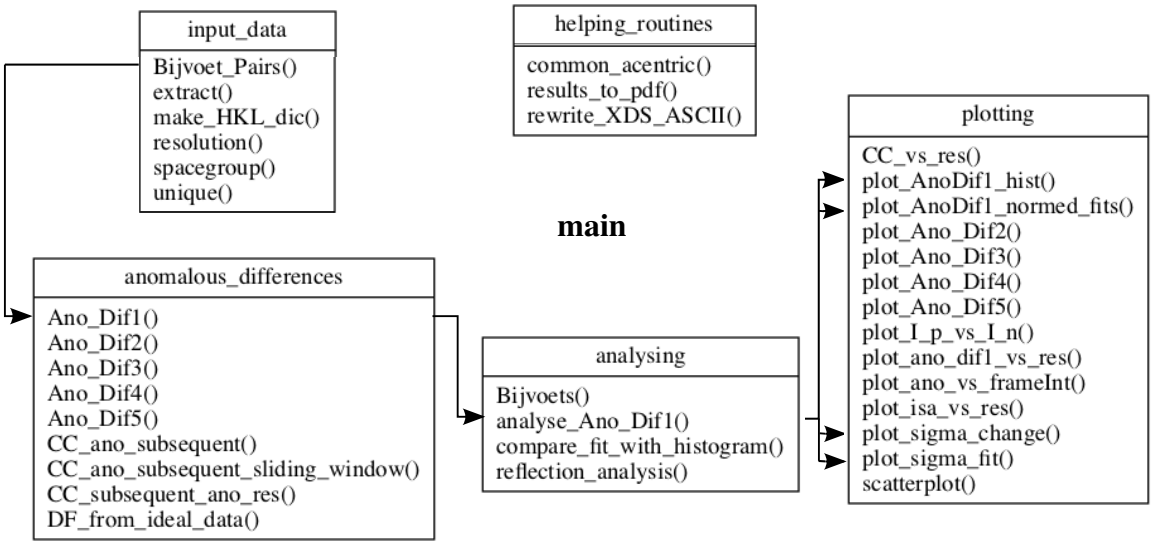



Figure 5.1: Outline of the program for analysing the average anomalous differences including their classes and functions. The flow for calculating the $\langle \Delta F \rangle$ values and the metric is indicated with arrows.

- the normalized histogram including normal distribution fits and corresponding parameters σ and $x_0$ can be analysed for the accumulated data or processed turn by turn (option '2d', method *plot_AnoDif1_normed_fits*) and
- the change of the normal distribution parameter σ can be displayed and fitted (method *plot_sigma_fit*).

A rough outline of the program including their classes and functions can be seen in Fig 5.1.

## 5.2 Program validation

With the output from XDS and the reflection statistics of PHENIX [4] it can be shown that the class *input_data* works correctly (compare Fig. 5.2 and Fig. 5.3). To prove that the single reflections are assigned correctly, the output of XDSCONVERT was used. With the help of the *plotting* class, it can also be proven that the code works, e.g. by calculating the correlation coefficient of two data sets, mapping them against the resolution and comparing them with the output of SHELXC (see Fig. 5.4). The slight differences between the two curves can be explained by the local scaling performed within SHELXC.

output of the function Bijveot_Pairs calling all functions in the class input_data

```
 DEB The data are stored in /Volumes/New_Volume/PETRA_data/20150927/PROCESSED_DATA/lys_11/
 DEB The crystal has the spacegroup P43212.
 ---
 DEB extract started
 DEB The cell parameter are 77.559 77.559 37.854 90.0 90.0 90.0
 DEB class input_data is initialized.
---
 DEB Bijvoet_Pairs started
 ---
 DEB List of theoretical unique reflections written
 DEB there are 13214 theoretical unique reflections.
 ---
 DEB make_HKL start. Reading from XDS_ASCII.HKL
 DEB XDS_ASCII.HKL closed.
 DEB 280795 reflections loaded.
---
 DEB resolution calculated.
 DEB image number done.
 DEB 278866 reflections written to dictionaries, including systematic absences.
 DEB 2 Dictionary(s) filled.
 ---
 DEB The Laue group is 4/mmm

 DEB 38 theoretically unique systematic absent reflections were measured.

 DEB 2132 unique centric reflections were measured.

 DEB 11064 Bijvoets were found and written to dictionary Bijvoets.

 DEB 24294 unique reflections were measured.
 __main__ (main_lys_11_20150927.py:23):
    125.432 seconds
```

Figure 5.2: Output of the self-written methods *Bijvoet_Pairs*.

output of the XDS statistics file CORRECT.LP

```
NUMBER OF REFLECTION RECORDS ON OUTPUT FILE "XDS_ASCII.HKL"      280795
NUMBER OF ACCEPTED OBSERVATIONS (INCLUDING SYSTEMATIC ABSENCES)  278866
NUMBER OF REJECTED MISFITS & ALIENS (marked by -1*SIGMA(IOBS))     1929
```

output of the PHENIX function reflection_statistics

```
Type of data: double, size=24294
Type of sigmas: double, size=24294
Number of Miller indices: 24294
Anomalous flag: True
Unit cell: (77.559, 77.559, 37.854, 90, 90, 90)
Space group: P 43 21 2 (No. 96)
Systematic absences: 38
Systematic absences not included in following:
Centric reflections: 2132
Resolution range: 54.8425 1.70039
Completeness in resolution range: 0.999341
Completeness with d_max=infinity: 0.999341
Anomalous completeness in resolution range: 0.999458
Bijvoet pairs: 11060
Lone Bijvoet mates: 4
Anomalous signal: 0.0247
Wavelength: 1.5498
```

Figure 5.3: Verification of the self-written method with a partial output from XDS in the CORRECT.LP file and PHENIX reflection statistics. Adding the number of Bijvoet pairs and the number of lone Bijvoet pairs from the PHENIX output results in the same number of Bijvoet pairs found by the self-written method *Bijvoet_Pairs* (compare Fig. 5.2).

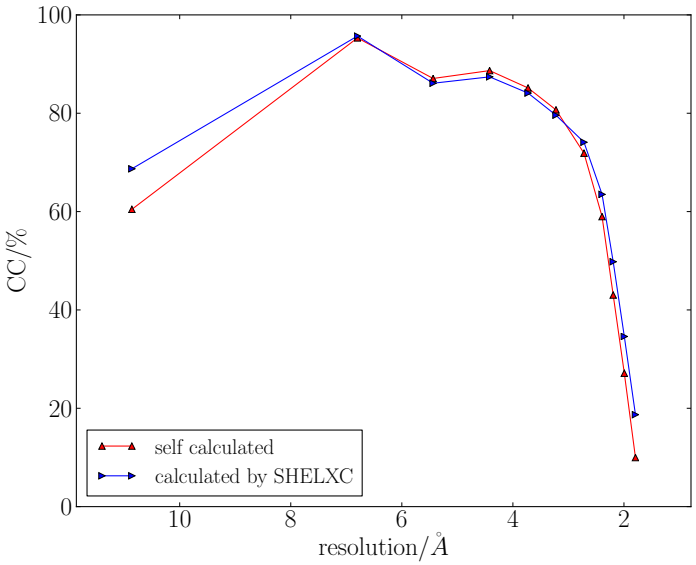



Figure 5.4: The correctness of the code can be proven by different plots, e.g. by comparing the self-calculated correlation coefficient of two subsequently recorded turns with the one calculated by SHELXC.

## 5.3 The ⟨ΔF⟩ values for Thaumatin crystals

The ⟨ΔF⟩ values of all acentric reflections were calculated with the program explained in the first section of this chapter for each wedge and the accumulated data. As before, data set 150421_tha1

will be analysed in detail, while the plots of the other data sets can be found in appendix B. The histograms of the ⟨ΔF⟩ values including their fits are analysed as well as the fits of the normalized histograms with normal distributions, leading to the so-called $\sigma_{\langle\Delta F\rangle}$ metric.

### 5.3.1 Histograms of ⟨ΔF⟩

The analysis of the maximum values and the distribution of the ⟨ΔF⟩ values for the wedges resulted in an interval of ± 3 for the histogram. To investigate the effects of radiation damage in the single wedges, the histograms of the first and the last wedge were plotted in Fig. 5.5.
Due to the bell-shaped form of the histograms, Gaussian distributions were chosen as fitting models. The distribution of measured intensities around their true value can be approximated by a normal distribution, if the central limit theorem is applied to the Poisson-distributed intensity counts [35]. Assuming this, the distribution of the structure factor amplitudes can then as well be approximated by a Gaussian, despite the fact that the errors of the intensities and the structure factors cannot strictly be Gaussian, but however do near a Gaussian distribution [46]. Ursby & Bourgeois [101] proved that based on these assumptions, also the anomalous differences should have a Gaussian distribution.
The histogram of the first and last wedge show clear differences. The one from the last wedge is broader and flatter than the one of the first wedge, which is also reflected in the fitting parameters

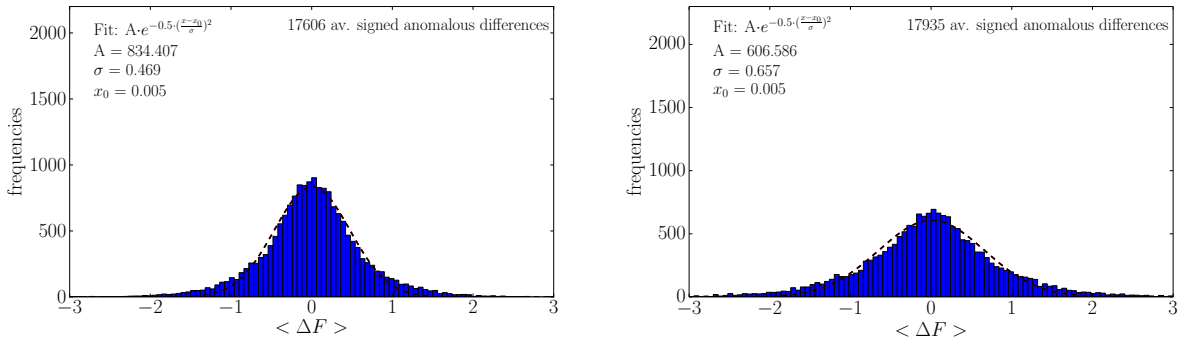
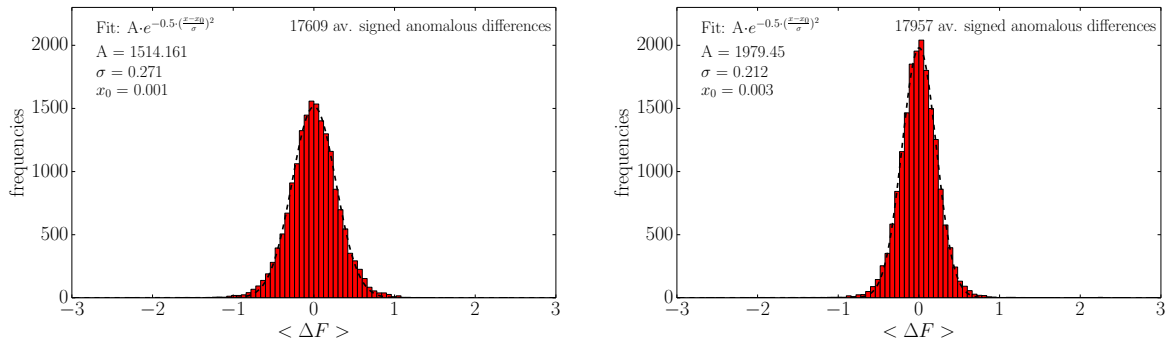



Figure 5.5: Analysis of the ⟨ΔF⟩ values, taking only the acentric reflections into account. Fig. 5.6 a) represents the histogram of the wedgewise processed data based on the first 360° of data, b) depicts the histogram of the last 360° of data (fifteenth turn). The histograms are calculated with the same 100 bins, correlating to a bin width of 0.06, covering 99.9% of all ⟨ΔF⟩. The fitting parameters A, $x_0$ and σ are written into the figures.

of the Gaussian distribution. Furthermore, the quality of the fit is worse for the last wedge, as it can be seen from the mean square deviation of the histogram and the fit, which nearly doubles from $6.68 \cdot 10^{-4}$ for the first to $1.17 \cdot 10^{-3}$ for the last wedge.
The broadening of the histogram as seen for the last wedge can be explained by the fact that with the progress of radiation damage, more and more reflections lose their intensity. While for the first wedge of data set 150421_tha1 6.4% of the reflections have intensities <0, this percentage increases to 25.4% for the last wedge. A detailed analysis of the multiplicity of the Bijvoet pairs with the method *Bijvoets* (compare class *analysing*, appendix C) shows that the mean multiplicity of the Bijvoet positive and Bijvoet negatives went down from 12.2 with a standard deviation of 3.0 in the first wedge to 9.4 with a standard deviation of 3.7 in the last wedge. At high resolution comprising the range from 1.9-2.1 Å, the mean multiplicity is significantly lower, which accounts for 2.6 and a standard deviation of 4.7 in the first wedge and a mean multiplicity of 1.7 with a standard deviation of 3.1 in the last wedge. The decrease in the mean multiplicity shows that especially at high resolution outliers cannot be averaged out so easily, leading to higher anomalous differences and a broader histogram. The slightly higher number of total ⟨ΔF⟩ values in the last wedge can be traced back to the individual scaling of the wedges.

In contrast, the histograms based on the data accumulated in five and fifteen 360° turns presented in Fig. 5.6 show a trend towards narrower and higher distributions, the more data are added. The mean square deviation of histograms and fits decreases by half from $6.68 \cdot 10^{-4}$ based on one wedge to $3.68 \cdot 10^{-4}$ if the data from all fifteen turns are included. Additionally, the number of ⟨ΔF⟩ values increases.
The sharpening of the curve can be explained by the fact that the Gaussian distribution of the ⟨ΔF⟩ values is a convolution of the true differences and the errors. For both, a Gaussian distribution can be assumed [101], as long as the errors are statistical independent. With increasing multiplicity, the data accuracy increases [21], leading to a narrower distribution. The increase in the number of ⟨ΔF⟩ values is due to the fact that with an increasing number of reflections more unique reflections fulfil the requirement to calculate ⟨ΔF⟩ values (I, σ(I) > 0 and at least one Bijvoet positive and one Bijvoet negative reflection).

These observations are true for all data sets, even though the broadening of the histograms of the wedges and the narrowing of the ones for the accumulated data are differently pronounced due to the different diffracting capacity of the single crystals and the different experimental set-




(a) (b)

Figure 5.6: Analysis of the ⟨ΔF⟩ values, taking only the acentric reflections of the accumulated data into account. Fig. 5.6 a) shows the histogram of the data accumulated in five 360° turns and b) shows the histogram based on the ⟨ΔF⟩ accumulated in fifteen 360° turns. The bin width is the same as in Fig. 5.5.

ups (compare Fig. B.13- B.16).

### 5.3.2 Fits with normal distributions

As the number of ⟨ΔF⟩ values and with that the area of the histograms only vary by about 3% between the first and last data set for both wedges and accumulated data, the histograms can be normalized. Accordingly, the histograms can be fitted by normal distributions, thereby reducing the number of fitting parameters. The normal distributions were plotted for all 360° turns for both the wedges and the successively accumulated data of data set 150421_tha1. The plots for the other data sets can be found in appendix B.

#### 5.3.2.1 Wedgewise processed data

For the wedges, the distributions become flatter and broader by and by from the first turn on, but in a non-uniform manner (compare Fig. 5.7). Especially the curves fitting the first three normalized histograms are very close together. For most other data sets plotted in appendix B.3, the first two to five curves become slightly sharper and higher before they broaden and flatten. The curves are centred around 0, showing only slight variations.
The sharpening of the first two to five normal distributions of the wedge-wise processed data is likely to be explained by the increase in disulphide disorder, i.e. specific radiation damage. As it

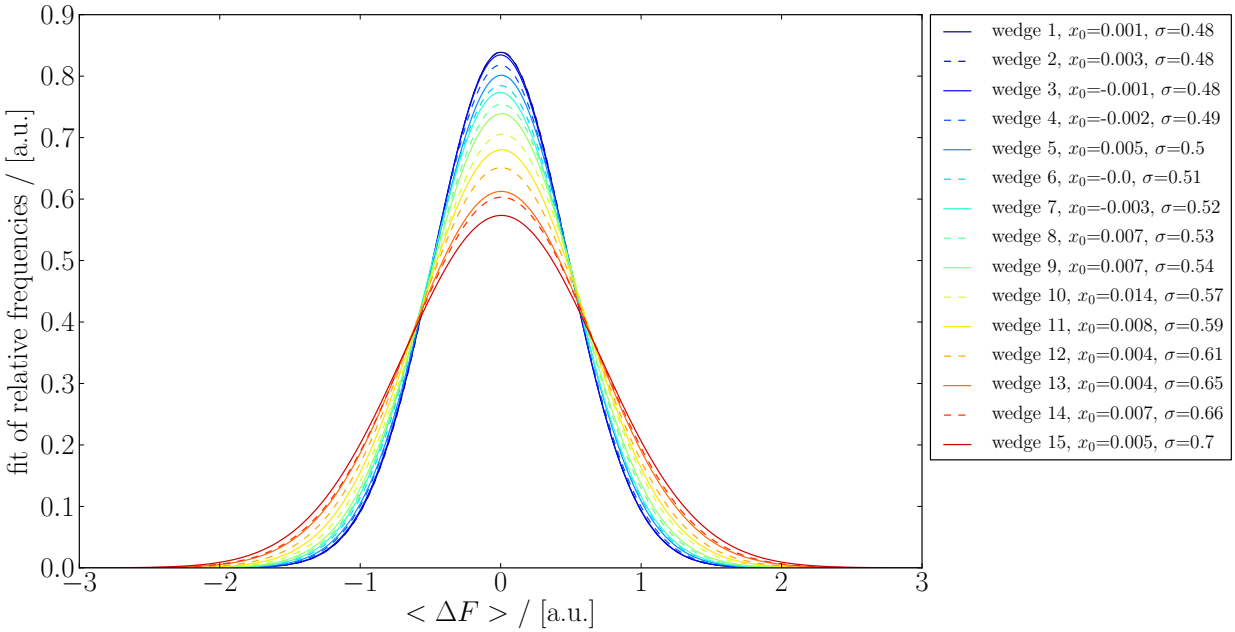



Figure 5.7: Normal distributions obtained by fitting the normalized histograms of the ⟨ΔF⟩ values for the fifteen different wedges.

could be shown in the previous chapter, the specific radiation damage occurs at lower doses and before global radiation damage becomes clearly visible. The successive and clear sharpening of the first five distributions of the strongest data set 150421_tha16 (compare Fig. B.10) supports this assumption. The $x_0$ values are within fitting accuracy, as the slight changes of these values are by one order of magnitude smaller than the histogram bin width of 0.06.

The apparent connection between radiation damage and the change in the histograms can be depicted in the change of the fitting parameter σ, from now on referred to as $\boldsymbol{\sigma_{\langle \Delta F \rangle}}$ **metric** (compare Fig. 5.8). The σ values are affected by two sources of error: a statistical error of $\sqrt{N}$, where N is the number of ⟨ΔF⟩ values, and a fitting error. The errors are calculated for every σ value (compare appendix C, method *plot_AnoDif1_normed_fits* and method *plot_sigma_fit*) and appear as error bars in the plots. The errors in the first wedge account for approximately 1% of the σ value, while they range up to 1.9% of the σ value for the last wedges, where the fitting error is increased due to the changes in the histogram. The σ values for the data processed in wedges seem to stay the same for about three wedges and then slowly starts to increase. Interestingly, the second value is slightly lower than the first one.
The other data sets show a similar behaviour, i.e. the σ values first slightly decrease and then start

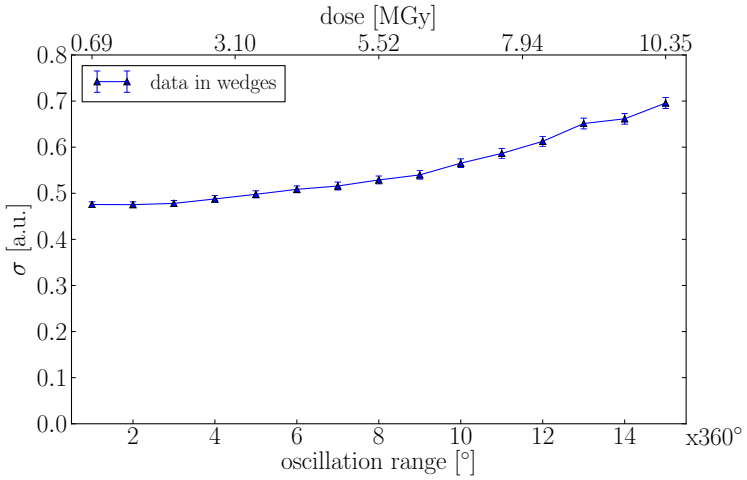



Figure 5.8: Plot of the parameter σ obtained by fitting the histograms of the wedges with normal distributions, plotted versus the number of 360° turns.

to increase. However, the point where the σ values start to increase differs between the different data sets. There are no remarks in literature that explain this behavior, but as it happens in all data sets, it might be a systematic effect.

The initial σ values of the different data sets vary from 0.45 for data set 150421_tha4 to 0.62 for data set 150927_tha3. Despite the fact that the distribution of the ⟨ΔF⟩ values is a convolution of the real anomalous differences and the errors, it seems as if that the systematic error is not the dominating one. Would this be the case, there should be a clear difference between the data sets collected with and without CRL. Terwilliger [98] proposed that apart from the measurement uncertainties, inaccuracies in data collection, e.g. for low resolution reflections, scaling or absorption corrections should also be considered, which might vary from crystal to crystal. It can therefore be hypothesized that the data quality of the crystals plays a major role, which is supported by the fact that the σ values of the best data set increase four 360° turns later than most of the other data sets.

#### 5.3.2.2 Accumulated data

For the accumulated data, the curves become narrower and higher, whereas the difference between the first and the second curve is the largest (compare Fig. 5.9). The more data are added, the smaller the difference between the subsequent curves. The center of the curve is also centred around 0 within fitting accuracy.

As described before, the sharpening can be explained by the fact that doubling the multiplicity

leads to an improvement of the true anomalous differences by a factor of $1/\sqrt{2}$ [21], assuming that the errors are statistically independent. Accordingly, the change between the first two curves is the strongest, as the multiplicity for the next curve is only increased by a third, resulting in an improvement by a factor of approximately $1/\sqrt{3}$. Adding more and more data leads to less improvements, on account of which the curves differ less and less.

The $\sigma_{\langle\Delta F\rangle}$ metric for the accumulated data of data set 150421_tha1 can be seen in Fig. 5.10. The previously described sharpening of the curves is obviously also reflected in the σ values. For this data set, they converge to 0.22. After fifteen turns, the other data sets converge to values between 0.21 for data set 150421_tha4 to 0.26 for data set 150421_tha6. The difference in the values cannot be reduced to different systematic errors, as it can be seen from Table 4.1. Here, specially errors arising from scaling might play a role here, as the data input is based on up to fifteen differently scaled wedges.

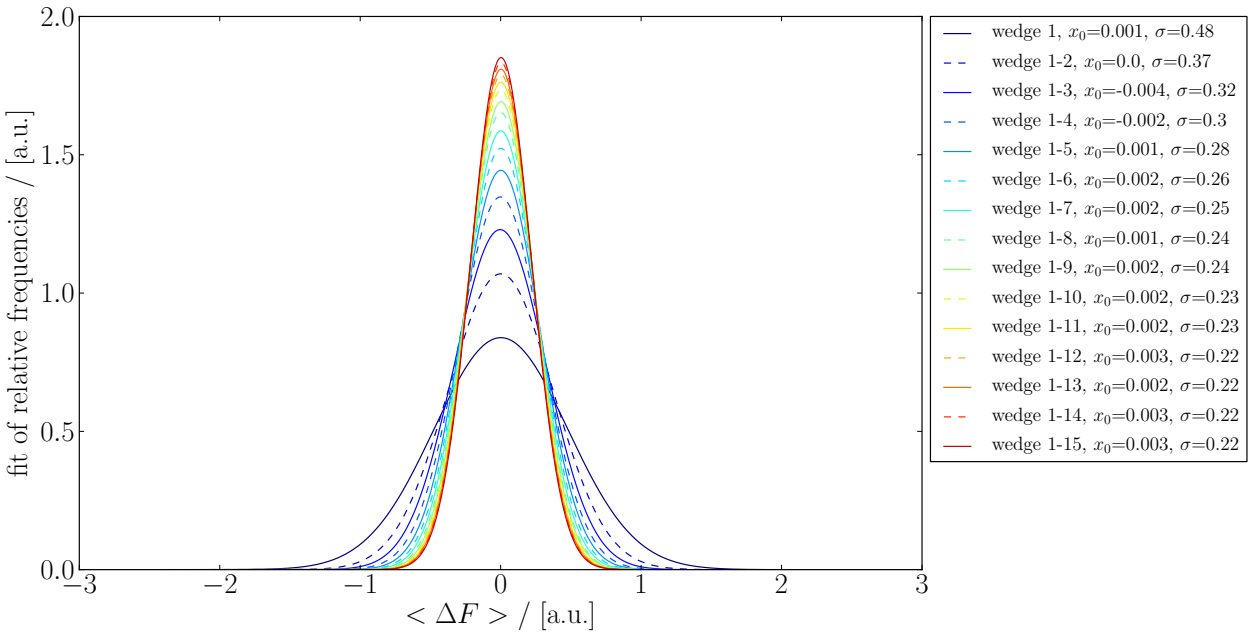



Figure 5.9: Normal distributions obtained by fitting the normalized histograms of the ⟨ΔF⟩ values for the accumulated data.

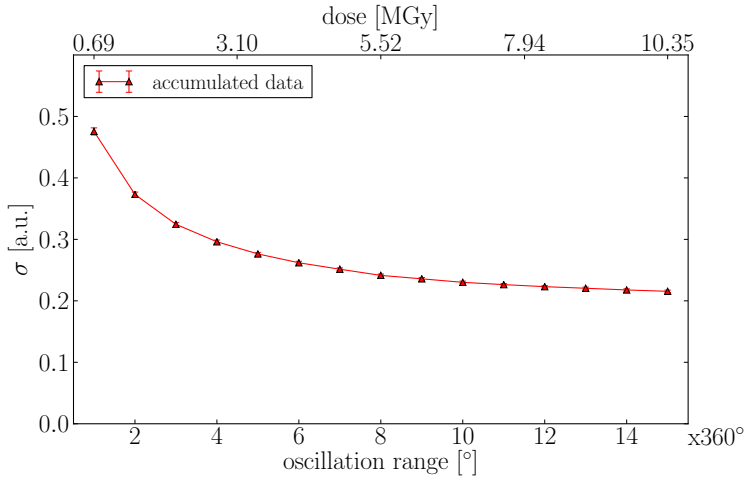



Figure 5.10: Plot of the parameter σ obtained by fitting the histograms of the accumulated data with normal distributions, plotted versus the number of 360° turns.

### 5.3.3 The $\sigma_{\langle\Delta F\rangle}$ metric for different resolution shells

Based on the previous section, one question regarding the $\sigma_{\langle\Delta F\rangle}$ metric remains unanswered: how can the very small, yet systematically occurring decrease of the first few σ values for the wedges can be explained? For obtaining a more detailed picture, the analysis of the $\sigma_{\langle\Delta F\rangle}$ metric was performed for three different resolution shells for the wedges and the accumulated data. These resolution shells range from the highest resolution shell to 2.0 Å, another from 2.0 Å to 2.8 Å and a third one including all reflections with a resolution greater than 2.8 Å. The medium resolution cut-off at 2.8 Å was chosen, because this resolution cut-off turned out to be good for solving the substructure based on experience and which was therefore also used in SHELXD.

The plots for data set 150421_tha1 can be found in Fig. 5.11. As the number of included anomalous differences is lower for the histograms of the different resolution shells, the statistical error and with that the total error is increased, while the fitting error only plays a minor role.

In the resolution shell ranging from 1.9 – 2.0 Å, the σ values of the wedges are higher from the start on and increase earlier compared to the curve which includes all data. The ratio of the first two σ values of the accumulated data is with 0.705 very close to the expected value of $1/\sqrt{2}$. After fifteen turns, the curve of the accumulated data converges to 0.28 (compare Fig. 5.11 b). The medium resolution shell comprises the greatest share of $\langle\Delta F\rangle$ values (compare Fig. 5.11 c). Therefore, its course resembles very much the one of all $\langle\Delta F\rangle$ values, which were analysed in the previous section. The σ values in the lowest resolution shell behave differently (compare Fig. 5.11 d). For the wedges, they decrease slightly after the first turn and increase again after the

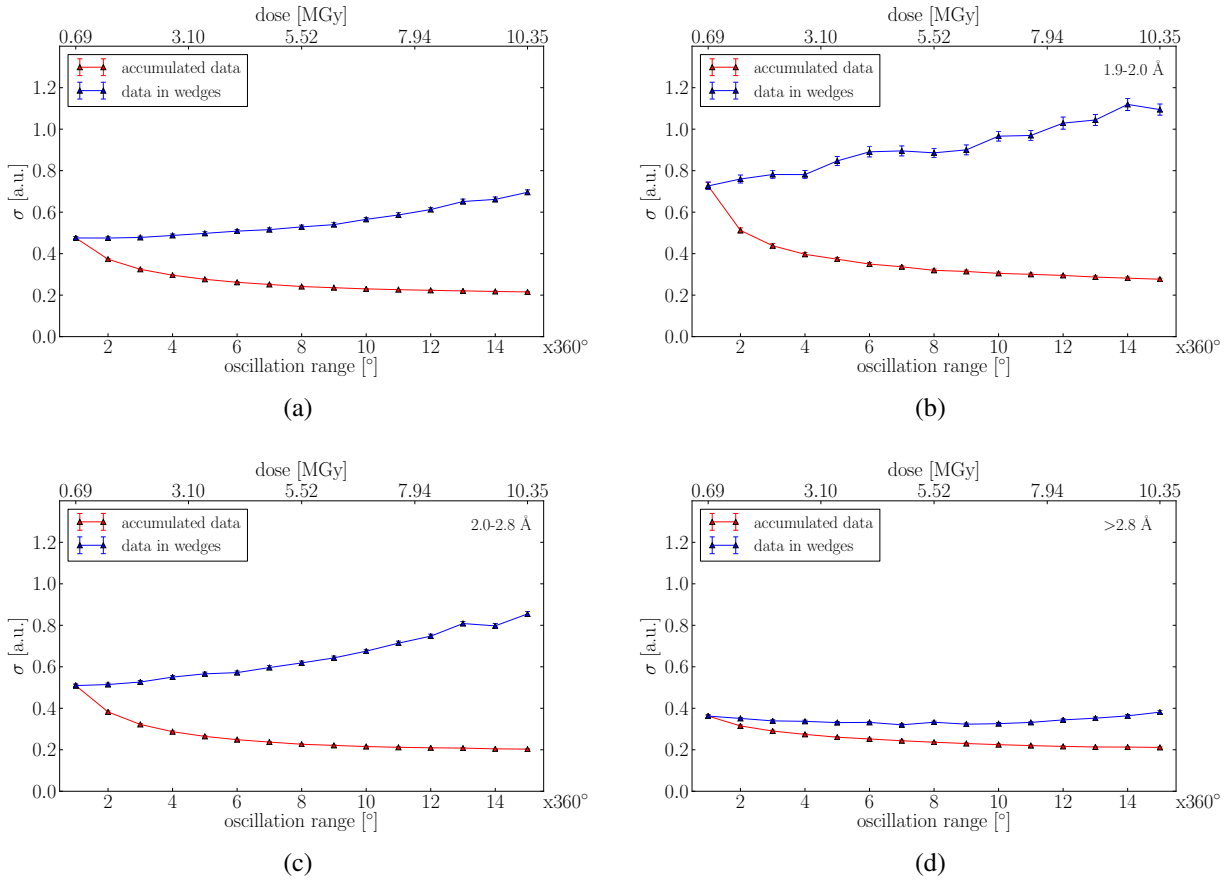



Figure 5.11: Plot of parameter σ of the normal distributions fitting the normalized histograms of the average signed anomalous differences against the number of 360° turns and dose, depending on the resolution of the acentric reflections. The red curves stand for accumulated, the blue ones for wedge-wise processed data. The single plots are a) for all data, plotted here as a reference, b) for data with a resolution of 1.9 – 2.0 Å, c) for data with a resolution of 2.0 – 2.8 Å and d) for data with a resolution of > 2.8 Å.

tenth turn, but do not reach the initial value. Furthermore, the differences between the σ values of wedges and accumulated data is also much less pronounced, i.e. the ratio between the first and the second σ value is 0.87.

The corresponding curves for the other data sets can be found in the appendix, section B.4. It can be seen that they all behave as described above. In data set 150421_tha6, the processing error of XDS becomes visible in the $\sigma_{\langle\Delta F\rangle}$ metrics apart from the one for the lowest resolution shell.

In the highest resolution shell for data processed in wedges, global radiation damage becomes visible first, leading to an increased error. Consequently, the broadening of this distribution and

with that an approximately linear increase of the σ values occurs earlier than in the other resolution shells. The higher σ values from the start can be explained by absolute higher errors, as the corresponding reflections are also weaker (compare section 2.1.4.3). The medium resolution shell is only affected later by global radiation damage. In principle, one can transfer the explanation for the different behaviour of well diffracting crystals discussed in section 4.5.3.1 to the reflections in the medium resolution shell. At some point, the signal becomes proportional to the errors and the distributions broadens, but the broadening sets in later than for the high resolution reflections. At resolutions lower than 2.8 Å, this effect is not visible any more, indicating that the reflections are no longer affected by global radiation damage. The slight reduction of the σ values can probably be explained by specific radiation damage.

The last hypothesis is indirectly supported by the scattering arising from sulphur atoms having different distances from each other. Based on the Debye-equation [24, 116]

$$\mathbb{E}(|\vec{F_h}|^2) = \sum_j \sum_k f_j(k) f_k(h) \frac{\sin(2\pi d_{jk} h)}{2\pi d_{jk} h} \tag{5.2}$$

the expectation value $\mathbb{E}$ of the squared structure factor amplitudes for atoms with a certain distance $d_{jk}$ at a given reciprocal lattice spacing can be calculated. The expectation value varies with resolution d according to $\frac{\sin(2\pi r/d)}{2\pi r/d}$. For the distances between two sulphur atoms, three are interesting. For one, there is the length of a disulphide bond $r_1 = 2.05$ Å. The other two are the Van-der-Waals distance between two sulphur atoms ($r = 3.60$ Å) and the distance of $r_2 = 2.80$ Å, which was used based on experience and which is also approximately the average of the two other lengths. Fig. 5.12 shows the values $\frac{\sin(2\pi r/d)}{2\pi r/d}$ for the three distances plotted in the range from 1.7 − 25 Å. The plot shows that the expectation value only exceed values of ± 0.2 for resolutions lower than 5 Å. These changes are likely to be too small to be seen in the weaker experimental data at high resolution, but may be seen for lower resolution reflections which are less error sensitive and additionally have higher expectation values. Accordingly, changes in the substructure would rather become visible for low resolution reflections, leading to smaller values as if the disulphide bonds would be intact.

To get an estimate, as to whether this is actually true, the ratios of the mean intensity of wedge one and wedge fifteen were calculated for two low resolution bins, the first one ranging from 12.85 − 15 Å and the second from 15 − 25 Å. For the first resolution bin, this yielded a value of 0.75, while the ratio of the function $\frac{\sin(2\pi r/d)}{2\pi r/d}$ for the two distances 2.05 Å and 3.60 Å was 0.71. For the second resolution bin, the ratio of the mean intensities was 0.84, where the ratio of the

function resulted in 0.85. The ratios for the distances of 2.05 Å and 2.80 Å agree less well with the experimental values, as it can be seen from the corresponding values of 0.87 and 0.92 for the two resolution shells. Based on these values, one can assume that the disulphide bonds break and that the single sulphur atoms move apart to their van-der-Waals distance of 3.60 Å within these fifteen 360° turns.

Another indication that specific radiation damage is responsible for the different curve progression of the low resolution reflections is the behaviour of the accumulated data. While the decrease between the first and the second σ value for the other resolution shells is close to a factor of $1/\sqrt{2}$ expected for a random error, this is not the case for the low resolution shell. As the specific radiation damage introduces changes in the electron density, the error is not entirely random any more, explaining the change between the first and second value by a factor of 0.85. The slight, but consistently observable decrease in the first few values of the $\sigma_{\langle\Delta F\rangle}$ metric based on all data can therefore likely be attributed to specific radiation damage, before global radiation damage becomes the major effect leading to an increase of the σ values.

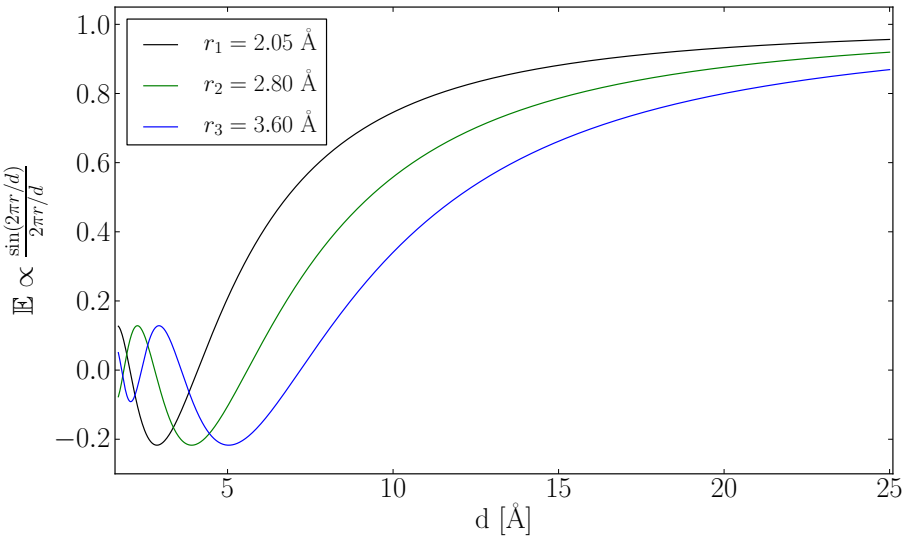



Figure 5.12: Change of the expectation value E which is proportional to $\frac{\sin(2\pi r/d)}{2\pi r/d}$ plotted against the resolution the normal length of the disulphide bond $r_1$, the average length of an intact and a broken disulphide bond $r_2$ and the van-der-Waals distance between two sulphur atoms $r_3$.

## 5.4 Using the $\sigma_{\langle\Delta F\rangle}$ metric for determining the best substructure

Trying to distinguish global and specific radiation damage, the σ values were divided in two parts. First, it can be assumed that the specific radiation damage dominates. In this regime, the σ values are lower than or equal to the initial sigma. The second part of the data is likely to be dominated by global radiation damage, as is witnessed by an increase in the sigma values above the initial one. Fitting both regions with linear functions leads to an intersection of the two fits. This intersection point can be interpreted as the one where specific radiation damage already occurred, but the global radiation damage has not severely damaged the structure yet. The σ values based on the accumulated data can be fitted with a function of type $\sqrt{a/x} - b$, which accounts approximately for the expected improvement for mainly random errors. For all fits, the errors of the single data points are considered.

The σ curves and the corresponding fits for data set 150421_tha1 can be found in Fig. 5.13. Comparing this intersection point of the two lines at 3.6x360° turns with the number of four 360° turns required for the best substructure, one could assume that this metric could give a hint at

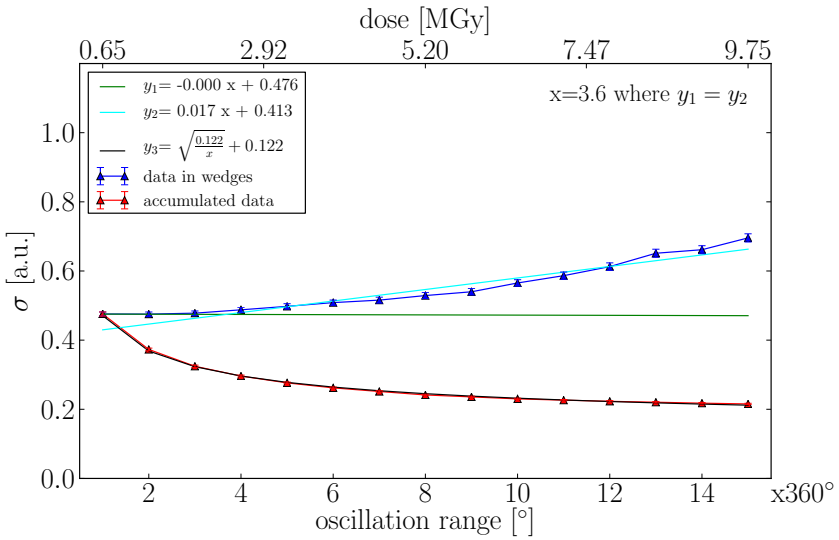



Figure 5.13: Plot of the σ values based on the fit of the normalized histograms with a normal distributions against the number of 360° turns for both wedges and accumulated data. For the wedges, the first three σ values are fitted with a green line and the fourth to the fifteenth with another, cyan line. Their intersection point is at 3.6x360° turns. The accumulated data were fitted with a function of type $\sqrt{a/x} - b$ in black.

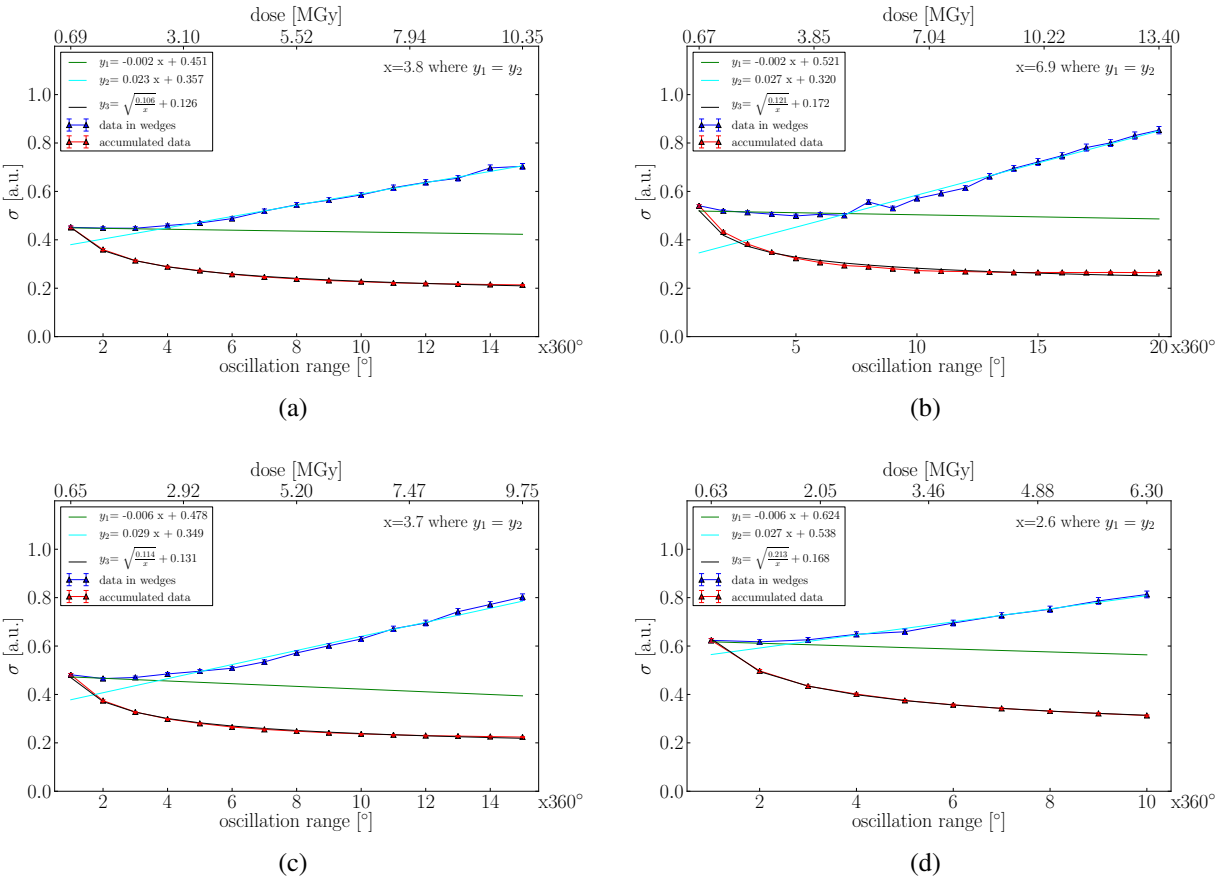



Figure 5.14: Plot of the σ values obtained by fitting the histograms of the ⟨ΔF⟩ values with normal distributions versus the number of subsequent 360° turns, for a) data set 150421_tha4, b) data set 150421_tha6, c) data set 150927_tha1 and d) data set 150927_tha3.

how many data should be included. The fits including the intersection points obtained from the other data sets can be found in Fig. 5.14.

Comparing them to the number of 360° turns for the best substructure determined with SITCOM (compare Table 5.1), it seems like the intersection point of the two lines can indeed be an indi-

| | 20150421 | | | 20150927 | |
|---|---|---|---|---|---|
| data set | tha1 | tha4 | tha6 | tha1 | tha3 |
| intersection point / 360° turns | 3.6 | 3.8 | 6.9 | 3.7 | 2.6 |
| best substructure / 360° turns | 4 | 2-7 | 7 | 4 | 3 |

Table 5.1: The different data sets and the intersection point of the two lines fitting the σ values in numbers of sequential 360° turns in comparison to the corresponding number of 360° turns required to get the best substructure.

cator for the balance between multiplicity and radiation damage, as the values are close to the amount of data required to determine the best substructure. The accumulated data are not useful for this purpose, as they are not sensitive for the changes induced by specific radiation damage, but improve, the more data are added.

## 5.5 Discussion

The very simple fit of the σ curve based on the wedges with two lines indicated a solution, while polynomial and exponential fits and their derivations did not give any hint to how many data to include for getting the best substructure. However, this fitting model is based on the assumption, that the widths of the histograms first decrease and that a fit with two lines is possible. An initial test with thaumatin data suggests that this might not be the case if the dose per 360° wedge is very low, i.e. below 0.25 MGy. In this case, the changes in the distribution of $\langle\Delta F\rangle$ due to radiation damage are likely to be very small and other factors like the individual scaling of the wedges might have a major effect on the widths of the distributions. Furthermore, if global radiation damage is too little pronounced, this fitting model might fail as well. For this cases, there might be a need to use another fitting model which still needs to be developed.

Apart from a certain dose per wedge, the proposed metric requires a certain number of data points and with that a certain multiplicity. Depending on the symmetry of the crystal, it would be possible to reduce the current oscillation range comprising one wedge to half or a fourth for obtaining more data points. In this case, the number of reflections included into the calculation of one $\langle\Delta F\rangle$ value would also be reduced, leading to a much broader histogram, as the precision would be lower. This is especially the case of the oscillation range is as high as 1° like in this experiment. Other effects lowering the precision of the data such as crystal quality, absorption and detector variability [47] which are not fully compensated by the individual scaling of the wedges would have the same effect. If a solution could be found that all the wedges were scaled the same without compensating for radiation damage, the effects of specific and global radiation damage would probably still be visible in the metric. However, this scaling algorithm still needs to be invented.

Initial results obtained by processing insulin and lysozyme data indicate that the Bijvoet ratio affects the course of the $\sigma_{\langle\Delta F\rangle}$ metric. For insulin crystals providing a Bijvoet ratio of about 1.4% at 8 keV, the decrease of the initial σ values is much more pronounced as for the lysozyme and thaumatin data, whose Bijvoet ratios account for about 1.2%. This is likely to be explained by

the increased signal-to-noise ratio. Additionally, the preliminary results suggest that the correct measurement of low resolution reflections is important for the application of the metric.

# Chapter 6

# Conclusions and Perspectives

With the aim of finding a balance between multiplicity and radiation damage in sulphur SAD experiments without the knowledge of the structure solution, high multiplicity data were collected on thaumatin crystals at P14. Data quality, systematic errors and radiation damage were analysed, and based on the known structure, the best substructure for every individual data set was determined. These results were used to verify the so-called $\sigma_{\langle\Delta F\rangle}$ metric, which determines the best substructure on the measured reflection intensities alone by using a self-written program.
Well-diffracting thaumatin crystals could reliably and reproducibly be crystallized. It was possible to perform the data collection under nearly ideal measurement conditions, i.e. with crystals smaller than the beam. Even illumination could be realized with a top-hat beam profile, which could be established due to the low emittance of PETRA III and the optical set-up described in section 3.1.2.1. As it was the aim to investigate the point of diminishing returns, the lifetime limit of the crystals was always exceeded. As it cannot easily be identified during the measurements, the later analysis showed that this point, i.e. a reduction of the mean reflection intensity to 50% of its initial value, was reached when the crystals had absorbed doses of 2.2 – 4 MGy. Three data sets were collected with an unfocused beam, while the other data sets were collected with a set-up using CRL, leading to an increase in the photon flux by a factor of 19. The only recently established CRL set-up was not yet fully commissioned and suffered from beam drifts during data acquisition. Combined with the difficulties in background scaling at short detector distances, the systematic error of the two latter data sets are higher than for the other data sets. However, only in one data set, the systematic error exceeds the value of 3% estimated by default in programs such as BEST [13].
The data evaluation showed the high quality of the five data sets and that the measurements are

nearly identical within experimental error. It turned out that scaling with XSCALE is not successful if many wedges are included from one crystal. However, the local scaling algorithm implemented in SHELXC seems to compensate for this. Determining the substructure with SHELXD by adding more and more data showed that higher multiplicity is indeed beneficial for finding a better substructure. Yet it should be analysed carefully whether the substructure is actually solved, as the CFOM alone obtained from processing accumulated data is not a reliable criterion. The program SITCOM was used to determine the best substructure, which could not always be clearly specified. That is why it is advisable to develop a score that takes the number of found sites, the rmsd, the CFOM and its spread into account to identify the best substructure solution more easily. Within this thesis, only the experienced-based resolution cut-off of 2.8 Å was used in SHELXD. However, the pipeline for processing the data and finding the substructure with its parametric design allows to adapt this and other parameters easily, so that different scenarios can systematically be tested against a known reference.
With the help of ANODE and the knowledge of the final structure, specific radiation damage could be monitored. Using the average anomalous peak heights as a metric for the best substructure would imply to use more data and with that more damaged data than the ones determined with SITCOM and the new developed metric. It is likely that the known phases are responsible for the difference, as density modification can lead to a compensation of the specific radiation damage regarding the quality of the phases. Nevertheless it would be better to use only the data leading to the best substructure, because it cannot be excluded that the final model would include broken disulphide bonds hampering the interpretation of the structure.

To be able to determine the best substructure without knowing the result, a program was written in Python to calculate anomalous differences in various ways, allowing for the total control over all input parameters. It was developed as a prototype for four space groups and only requires the CCP4 program UNIQUE and the reflections obtained by processing the raw data with XDS. Despite the fact that some approaches did not lead to conclusive results, the program and its modular structure allow further testing and development. Additionally, improvements regarding the time-consumption would be advantageous. Therefore, a transfer of the program into C++ will be performed by an experienced software engineer.
The developed $\sigma_{\langle \Delta F \rangle}$ metric monitors changes in the width of the normal distributions with which the histograms of the average signed anomalous differences are fitted (compare Fig. 6.1). Being a convolution of the true anomalous differences and errors, the change in the distributions is very

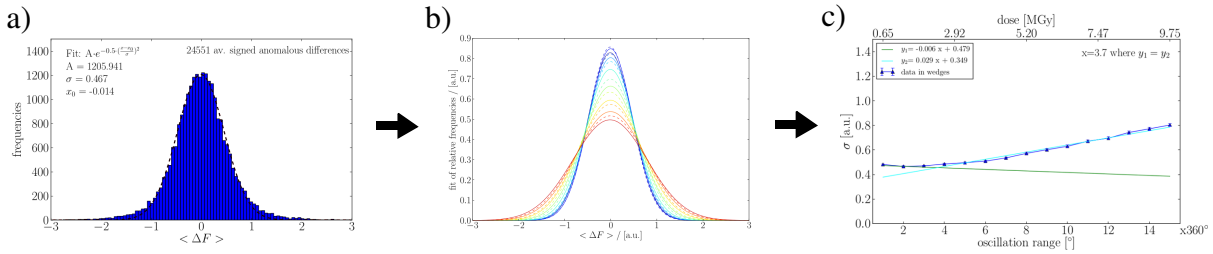



Figure 6.1: Flow chart visualizing the derivation of the $\sigma_{\langle\Delta F\rangle}$ metric. The $\langle\Delta F\rangle$ values are a) plotted in a histogram, which is fitted with a Gaussian distribution. Then, the histogram of every wedge is normalized and fitted with a normal distribution, where b) shows the normal distribution of all wedges. The σ parameters of these normal distributions, referred to as $\sigma_{\langle\Delta F\rangle}$ metric, are plotted in c) for all wedges versus the subsequent 360° turns and fitted with two line.

likely to be caused by radiation damage. The analysis of the $\sigma_{\langle\Delta F\rangle}$ metric for different resolution shells based on the wedges suggests that specific radiation damage mainly affects low resolution reflections, while reflections at higher resolution are rather affected by global radiation damage. Fitting the σ values based on all $\langle\Delta F\rangle$ with two lines can be seen as an approach of separating specific and global radiation damage effects. The intersection point of the two lines corresponds to the best substructure determined based on the known structure. The doses accumulated up to this point range from 1.6-4.6 MGy, which is in agreement with the dose limit of 5 MGy applied by Liu et al. [68] and Weinert et al. [105] and is mostly within the limit for sulphur SAD experiments currently discussed at conferences ranging from 2-4 MGy.

Regarding a further testing of the metric, it would be interesting to see whether additional anomalous scatterers such as calcium in trypsin crystals would have an effect on the metric, as the process of oxidation is different from the breaking of disulphide bonds [66, 112]. Therefore, the metric should be tested on other systems, especially on those, where the Bijvoet ratio is weak and low multiplicity is not sufficient for solving the structure by sulphur SAD. Another option would be to perform fine-sliced data collection protocols, e.g. with an oscillation range of 0.1° to investigate how the improved measurement of the anomalous differences is affecting the metric [71] and whether these improvements would allow to decrease the wedge size. Furthermore, it would be interesting to see whether the intersection point of the two lines could be shifted if sophisticated data collection protocols such as crystal orientation along their symmetry axes, inverse beam or measurements under multiple orientations would be applied.

In view of the fact that limited sample has to be measured within short shifts of beamtime at

a synchrotron, a data collection strategy aiming for low dose and high multiplicity in native SAD experiments may not be the most intuitive one to choose. However, given that the Bijvoet ratio is high enough and that data collection is performed carefully, this strategy can be carried out relatively easy and lead to successful structure solution [75, 105]. As no heavy atoms have to be introduced into the crystals, the diffraction quality is mostly better in native SAD experiments. Based on the obtained experimental phases, *de novo* structure determination is possible. Additionally, the model bias is lower compared to structure solution with molecular replacement. The $\sigma_{\langle \Delta F \rangle}$ metric presented here is an approach to answer the question whether the amount of data collected is already sufficient or whether there is still room for improving the substructure. Requiring only the results from XDS, it could be implemented in a program which could be run at the beamline, helping to optimize the use of the beamtime.

# Appendices

# Appendix A

# Data processing and plotting pipeline

As described in 3.2.4, the different stages in data processing are controlled by the data processing pipeline *Process_all*. It contains the following methods:

- *xds* for processing raw data with a given resolution cut-off by using the program XDS,
- *xscale* to scale all XDS_ASCII.HKL files given in the list with the program XSCALE,
- *XDS_im_shift* for manipulating the image number in the XDS_ASCII.HKL file,
- *xds_merge* for writing the reflections of different XDS_ASCII.HKL files in one file,
- *shelx* for processing the XDS_ASCII.HKL files with the programs SHELXC, SHELXD and SHELXE},
- *sidecheck_sitcom* for comparing substructures determined with SHELXD to a reference substructure with the program SITCOM,
- *emma* for comparing substructures determined with SHELXD to a reference substructure with the program PHENIX and
- *arpwarp* for automated model building with the program ARP/WARP.

The program *Extract_all* contains the method *extract* and selects the results. These results are plotted with the program *Plot_all*, where the method *plot_results* is used for all results except the ones obtained with SITCOM. For the latter, a seperate method *plot_sitcom_results* is used. As running the program ANODE and plotting the results follows a slightly different scheme, a seperate program called *check_ano_peaks* was written.

Process_all

```
import re
import os
import string
import time

# changing the XDS.INP file and processing the data with XDS
def xds(workList, directory, raw_data, res_cut):
    for dataSet in workList:
        xdsDir =directory+ dataSet
        if (not os.path.exists(xdsDir)):
            print "directory does not exist"
        os.chdir(xdsDir)
        no=dataSet.split("_")[-2]
        print no
        f = open('XDS.INP', 'r')
        xds_original=f.readlines()
        f.close()
        os.system("cp XDS.INP XDS_org.INP")
        f_new=open('XDS.INP', 'w')
        f_new.write('! \n')
        f_new.write('! XDS.INP file generated by Selina \n')
        f_new.write('! \n')
        for num, line in enumerate(xds_original):
                if (string.find(line,'INCLUDE_R') == 0):
                        res_num=num
                        print line
                elif (string.find(line, "NAME_TEMPLATE_OF_DATA_FRAME")==0):
                        file_name=num
                        print "num"
        part1=xds_original[0:res_num]
        part2=xds_original[res_num+1:file_name]
        part3=xds_original[file_name+1:]
        for _line in part1:
                f_new.write(_line)
        f_new.write("INCLUDE_RESOLUTION_RANGE=999 "+res_cut+"\n")
        for line_ in part2:
                f_new.write(line_)
        f_new.write("NAME_TEMPLATE_OF_DATA_FRAMES="+raw_data+no+"_?????.cbf"+"\n")
        for l in part3[0:-2]:
                f_new.write(l)
        f_new.write("FRIEDEL'S_LAW=FALSE \n")
        f_new.close()
        print '\n Now running xds_par in directory \'%s\' ... ' % (directory+dataSet)
        os.system("/home/storm/tmp/XDS/xds_par")
        os.system("/home/storm/tmp/XDS/xds_par | tee xds_par.log | awk '/\*\*\*\*\* /||/PROCESSING/'")

#writing input file and running XSCALE
def xscale(directory, workList, space_group, uc):
    List=[]
    for dataSet in workList:
        List.append(directory+dataSet)
        f=open("XSCALE.INP", "w")
        f.write("OUTPUT_FILE=XDS_ASCII_scaled.HKL \n")
        print List
    for el in List:
        f.write("INPUT_FILE="+el+"/XDS_ASCII.HKL \n")
        f.write("SPACE_GROUP_NUMBER="+space_group+"\n")
        f.write("UNIT_CELL_CONSTANTS="+uc+"\n")
        f.write("FRIEDEL'S_LAW= FALSE \n")
        f.write("MERGE= FALSE \n")
        f.close()
        os.system("/home/storm/tmp/XDS/xscale_par")
        print '\n Exiting normally. \n'

# image number of the single reflections is manipulated by adding an "im_shift"
def XDS_im_shift(path, name, im_shift):
    fname = "XDS_ASCII.HKL"
    print " --- \n DEB Reading from %s " % (path)
    file=open(path+fname,'r')
    lines=file.readlines()
    file.close()
    header=lines[0:47]
    data=lines[47:-1]
    end=lines[-1]
    line_counter=0
    new=path+name+"_"+str(im_shift)
    if not os.path.exists(new):
        os.mkdir(new)
    os.chdir(new)
    print new
    new_file=open(new+"/XDS_ASCII_new.HKL",'w')
    for line in header:
        new_file.write(line)
    for line in data:
        z=float(line.split()[7])
        z_n=z+im_shift
        new_line='{:>6} {:>5} {:>5} {:>10} {:>10} {:>7} {:>7} {:>8} {:>9} {:>3} {:>3} {:>7}'.format(line.split()[0], line.split()
  [1], line.split()[2], line.split()[3], line.split()[4], line.split()[5],line.split()[6], str(z_n), line.split()[8], line.split()
```

```
  [9], line.split()[10], line.split()[11])
88          new_file.write(new_line)
89          new_file.write("\n")
90          line_counter=line_counter+1
91      new_file.write(end)
92      new_file.close()
93      print " DEB %s reflections have been written to the new XDS_ASCII.HKL file." %str(line_counter)
94
95  # appending the subsequently recorded XDS_ASCII.HKL files with manipulated image number
96  def xds_merge(_workList, out):
97      refl=[]
98      first_xds=open(_workList[0]+"XDS_ASCII.HKL", "r")
99      file=first_xds.readlines()
100     header=file[0:47]
101     end=file[-1]
102     data=file[47:-1]
103     first_xds.close()
104     new_XDS=open(out+"XDS_ASCII.HKL", "w")
105     for line in header:
106         new_XDS.write(line)
107     for li in data:
108         new_XDS.write(li)
109     for _path in workList[1:]:
110         print _path+"XDS_ASCII.HKL"
111         f=open(_path+"XDS_ASCII.HKL", "r")
112         lines=f.readlines()
113         for line_ in lines[47:-1]:
114             new_XDS.write(line_)
115     new_XDS.write(end)
116     new_XDS.close()
117     print " DEB merged XDS_ASCII.HKL file written."
118
119 # writing an input file and running SHELXC/D/E for SAD or MAD
120 def shelx(workList, directory, name, case, file_name, uc, spag, ha_d, dsul, s_res_cut):
121     for dataSet in workList:
122         os.chdir(directory+dataSet+"/")
123         f = open('run_shelx.csh', 'w')
124         f.write('#!/bin/csh -f\n')
125         f.write('# \n')
126         f.write('# SHELX script generated by Selina \n')
127         f.write('# \n')
128         f.write('/home/storm/tmp/SHELX/shelxc '+name+ ' > '+name+'_shelxc.log <<EOF\n')
129         if case=="SAD":
130             f.write('SAD '+file_name+ ' \n')
131             f.write('CELL '+str(uc)+'\n')
132             f.write('SPAG '+spag+' \n')
133             f.write('SFAC S\n')
134             f.write('SHEL 999 '+str(s_res_cut)+' \n')
135             f.write('FIND '+str(ha_d)+'\n')
136             f.write('DSUL '+str(dsul)+'\n')
137             f.write('NTRY 10000\n')
138             f.write('MIND -3.5\n')
139             f.write('MAXM 20\n')
140         if case=="MAD":
141                 if re.search(r'hrem', dataSet[0]):
142                     f.write("#NAT "+ dataSet[0]+"_scaled.HKL \n")
143                 if re.search(r'pk', dataSet[0]):
144                     f.write("#PEAK "+ dataSet[0]+"_scaled.HKL \n")
145                 if re.search(r'inf', dataSet[0]):
146                     f.write("#INFL "+ dataSet[0]+"_scaled.HKL \n")
147                 if re.search(r'hrem', dataSet[0]):
148                     f.write("#HREM "+ dataSet[0]+"_scaled.HKL \n")
149                 if re.search(r"lrem", dataSet[0]):
150                     f.write("#LREM " + dataSet[0]+"_scaled.HKL \n")
151         f.write('EOF\n')
152         f.write('/home/storm/tmp/SHELX/shelxd '+name+"_fa > "+name+'_fa.log\n')
153 #     f.write('/home/storm/tmp/SHELX/shelxe '+ name + ' ' + name+'_fa -a3 -m20 -s"+str(solvent)+" -z > ' + name+'_shelxe.log\n')
154 #     f.write('/home/storm/tmp/SHELX/shelxe '+ name + ' ' + name+'_fa -a3 -m20 -s"+str(solvent)+" -z > ' + name+'_shelxe.log\n')
155         print 'Finished writing file'
156         f.close()
157         os.system('chmod +x run_shelx.csh')
158         os.system('./run_shelx.csh')
159
160 # run sitcom to find no. of sites and rmsd of different substructures in comparison to a reference structure pdb
161 # required files to corresponding folder
162 def sidecheck_sitcom(directory, workList, name, uc, pdb, space_group, ha, ha_type, no_sol):
163     for el in workList:
164         print directory+el
165         os.chdir(directory+el)
166         os.system('cp '+pdb+ " .")
167         f = open('sitcom.inp', 'w')
168         f.write("unit_cell "+ uc +'\n')
169         f.write("space_group "+str(space_group)+"\n")
170         f.write('read_sol RPDB 1.0 '+pdb+ " "+ str(ha)+ " "+ ha_type+ "\n")
171         f.write("read_set SHELXD 1.0 "+ name+'_fa.lst '+str(no_sol)+" "+str(ha+7)+" \n" )
172         f.write('max_dist 3.0\n')
173         f.write("restrain_comp\n")
174         f.write("fix_asunit\n")
```

Process_all

```
        f.close()
        print "sitcom.inp is written"
        os.system('/Users/storm/Applications/sitcom < sitcom.inp > sitcom.log')
        #os.system('rm sitcom')
        print "sitcom is running"

# to verify the sitcom results, run Fabio Dall'Antonias Script for Phenix Emma called "lemmy.py"
def emma(directory, workList, name):
    for dataSet in workList:
        os.chdir(directory+dataSet)
        print directory+dataSet
        if not os.path.exists("lemmy.py"):
            os.system("cp "+directory+"scripts/lemmy.py .")
        if not os.path.exists("rsites.pdb"):
            os.system("cp "+directory+"rsites.pdb .")
        os.system("python lemmy.py "+name+"_fa.lst 100 > lemmy.log")

# converting the .phs file (output of SHELXE) to mtz and run arpwarp; required: sequence file in pir format in the input folder
def arpwarp(_input, name, uc, spag, residues, res_cut, pir):
    os.chdir(_input)
# converting the .phs file with the CCP4 program f2mtz
    f = open(name+'_phs2mtz.csh', 'w')
    f.write("#!/bin/csh -f\n")
    f.write("# Shell script for converting phs to mtz-format\n")
    f.write("# f2mtz keeps order of columns\n")
    f.write("f2mtz hklin "+ name +".phs hklout t_tmp.mtz > t_f2mtz.log <<END\n")
    f.write("CELL "+uc+"\n")
    f.write("SYMM "+spag+"\n")
    f.write("LABOUT H K L FP FOM PHIB SIGFP\n")
    f.write("CTYPOUT H H H F W P Q\n")
    f.write("END\n")
    f.write("cad hklin1 t_tmp.mtz hklout "+ name+".mtz > t_cad.log <<eof-cad\n")
    f.write("LABIN FILE 1 E1=FP E2=SIGFP E3=PHIB E4=FOM\n")
    f.write("LABOUT E1=FP E2=SIGFP E3=PHIB E4=FOM\n")
    f.write("eof-cad\n")
# run the ccp4 program freerflag to indicate free_r data
    f.write("freerflag HKLIN "+name+".mtz HKLOUT "+name+"_f.mtz > freerflag.log <<END\n")
    f.close()
    os.system("chmod +x "+name+'_phs2mtz.csh')
    os.system("./"+name+"_phs2mtz.csh "+ name+".phs")
    print "mtz file %s was written" %name
# run arpwarp
    g=open("arpwarp_"+name+".com", "w")
    g.write(" auto_tracing.sh ")
    g.write("datafile "+ name+"_f.mtz ")
    g.write("fp FP sigfp SIGFP phibest PHIExp fom FOM freelabin FreeR_flag")
    g.write("residues "+residues+" ")
    g.write("seqin "+pir+" cgr 1 ")
    g.write("buildingcycles 10 ")
    g.write("resol '20.0;"+res_cut+"' ")
    g.close()
    os.system("chmod +x arpwarp_"+name+".com ")
    os.system("./arpwarp_"+name+".com "+name+ " > "+_input+name+"_arp.log")
    print "arpwarp is running for %s" %file


# general parameter
directory="/Volumes/New_Volume/PETRA_data/20150927/PROCESSED_DATA/lys_10/"
workList=[]
for i in xrange(1,16):
    workList.append("xds_lys_10_"+str(i)+"_2/lys_10_1-"+str(i*360)+"/")
print workList
# XDS parameters
raw_data="/d/beamstorage/P14/2015/10300_10300/p3l-tschneider/20150927/RAW_DATA/lys_10/lys_10_"
res_cut=str(" 1.7")
# XSCALE parameter
space_group='96'
#XDS_im_shift parameters
im_shift=360
# xds_merge parameter
_workList=[]
# SHELX parameter
name="lys_10_acc"
file_name='XDS_ASCII.HKL'
uc="77.47 77.47 38.33 90.0 90.0 90.0"
spag="P43212"
case="SAD"
ha_d=6
dsul=4
solvent=0.36
s_res_cut=2.8
# sitcom parameter
ha=10
ha_type="S"
pdb=directory+"rsites.pdb"
no_sol=100
# anode parameter
pdb_path=directory+"lys_10_coot-0_refmac0.pdb"
```

```
263 new_name=name+"_ano_peak"
264 # arpwarp parameter
265 _input="/Users/storm/Documents/Experiments_2015/Anomalous_Diffraction/20150927/lys_10/structure_det/"
266 residues="129"
267 pir=_input+"lys.pir"
268 # call functions
269 xds(workList, directory, raw_data, res_cut)
270 xscale(directory, workList, space_group, uc)
271 for path in workList:
272          XDS_im_shift(path, name, im_shift)
273          _workList.append(path)
274          out=path+name
275          xds_merge(_workList, out)
276 shelx(workList, directory, name, case, file_name, uc, spag, ha_d, dsul, s_res_cut)
277 sidecheck_sitcom(directory, workList, name, uc, pdb, space_group, ha, ha_type, no_sol)
278 emma(directory, workList, name)
279 arpwarp(_input, name, uc, spag, residues, res_cut, pir)
280
```

Extract_all

```
import string
import pickle
import os
import numpy

# extracting the relevant information from the output of the programs
def extract(directory, workList, name, program):
    # lists to store the interesting parameter
    isa, totRmeas, CC_half, totSigAno=[], [], [], []
    CFOM, noOfSites=[], []
    CC_shelxe, CC_shelxe_i, res, res_i=[], [], [], []
    sites, rmsd, _rmsd=[], [], []
    peak_list, ha_list=[], []
    _arr=numpy.zeros(shape=(34, len(workList)))
    # iterate over the different turns
    for aa in workList:
        os.chdir(aa)
        z=[]
        # extracting results from XDS
        if 'XDS' in program:
            log = 'CORRECT.LP'
            xdsfile = open(log,"r")
            lines=xdsfile.readlines()
            for no, line in enumerate(lines):
                if (string.find(line,'total') >= 0):
                    SigAno=float(line.split()[-2])
                    Rmeas_p = line.split()[9]
                    Rmeas = float(Rmeas_p.split('%')[0])
                    if len(lines[no-1].split()[-4].split('*'))!=[]:
                        CC_=float(lines[no-1].split()[-4].split('*')[0])
                    else:
                        CC_=float(lines[no-1].split()[-4])
                    Isa=float(lines[no-1].split()[-6])
            xdsfile.close()
            isa.append(Isa)
            totSigAno.append(SigAno)
            totRmeas.append(Rmeas)
            CC_half.append(CC_)
        # extracting SHELXD results
        if 'SHELXD' in program:
            shelxd=open(aa+name+"_fa.res", "r")
            print aa+name+"_fa.res"
            for line in shelxd:
                if (string.find(line, 'CC(weak)') >=0):
                    CC=float(line.split()[5])
                    CC_w=float(line.split()[7])
                    CFOM.append(CC+CC_w)
            shelxd.close()
        # extracting SHELXE results
        if 'SHELXE' in program:
            shelxe=open(aa+name+".pdb", "r")
            shelxe_i=open(aa+name+"_i.pdb", "r")
            for line in shelxe:
                if (string.find(line,'TITLE') >= 0):
                    res.append(int(line.split()[7]))
                    CC_shelxe.append(float(line.split()[6].split("%")[0]))
            shelxe.close()
            for line in shelxe_i:
                if (string.find(line,'TITLE') >= 0):
                    res_i.append(int(line.split()[7]))
                    CC_shelxe_i.append(float(line.split()[6].split("%")[0]))
            shelxe_i.close()
        # extracting sitcom results
        if 'sitcom' in program:
            _f=open(aa+"sitcom.log", "r")
            print aa
            lines=_f.readlines()
            for num, line in enumerate(lines):
                if (string.find(line, '0   SHELXD     9999') >=0):
                    no= line.split()[-5]
                    sites.append(int(no))
            for num, line in enumerate(lines):
                if (string.find(line, ' #  Solution-ID      CFOM   NM   RMSD   Score') >=0):
                    no_line= num
                    for _line in lines[no_line+4:no_line+103]:
                        if no == _line.split()[-3]:
                            print  _line.split()[-2]
                            z.append(float(_line.split()[-2]))
            rmsd.append(min(z))
            print no, rmsd
    # write the results for the different programs to dictionaries
    if 'XDS' in program:
        XDS={}
        XDS.update({'I/$\sigma$ [a.u.]':isa})
        XDS.update({'$R_{meas}$ [\%]':totRmeas})
        XDS.update({'anomalous signal [a.u.]':totSigAno})
        XDS.update({'$CC_{1/2} [\%]$': CC_half})
        with open(directory+name+'_XDS.pic', "w") as g:
```

```
89            pickle.dump(XDS, g)
90        print " DEB XDS results written to file."
91        return XDS
92    if 'SHELXD' in program:
93        SHELXD={}
94        print CFOM
95        SHELXD.update({'CFOM [\%]':CFOM})
96        with open(directory+name+'_SHELXD.pic', "w") as b:
97            pickle.dump(SHELXD, b)
98        print " DEB SHELXD results written to file."
99        return SHELXD
100   if 'SHELXE' in program:
101       SHELXE={}
102       SHELXE.update({'CC [\%]':CC_shelxe})
103       SHELXE.update({'CC_i [\%]':CC_shelxe_i})
104       SHELXE.update({'res':res})
105       SHELXE.update({'res_i':res_i})
106       with open(directory+name+'_SHELXE.pic', "w") as c:
107           pickle.dump(SHELXE, c)
108       print "SHELXE results written to file."
109       return SHELXE
110   if 'sitcom' in program:
111       sitcom={}
112       print sites, rmsd
113       sitcom.update({'no. of found sites':sites})
114       sitcom.update({'rmsd [a.u.]':rmsd})
115       with open(directory+name+'_sitcom.pic', "w") as s:
116           pickle.dump(sitcom, s)
117       print "sitcom results written to file."
118       return sitcom
119
120# extract parameter
121directory="/Volumes/New_Volume/PETRA_data/20150927//PROCESSED_DATA/lys_10/"
122workList=[]
123for i in xrange(1,16):
124    workList.append("xds_lys_10_"+str(i)+"_2/lys_10_1-"+str(i*360)+"/")
125name="lys_10"
126program="SHELXD"   # can also be XDS, SHELXE, sitcom
127
128extract(directory, workList, name, program)
```

Plot_all

```
from matplotlib import *
import matplotlib.mlab as mlab
from profilehooks import timecall
from interval import interval
from matplotlib import rc
import matplotlib.pyplot as plt
import pickle
import pylab

# setting font and fontsizes
rc('font', weight='bold', **{'family':'serif','serif':['Computer Modern Roman']})
rcParams['lines.linewidth'] = 1
rc('text', usetex=True)
rcParams['figure.figsize'] = 10, 6
title_font = {'fontname':'Arial', 'size':'24', 'color':'black', 'weight':'normal',
              'verticalalignment':'bottom'}
axis_font = {'fontname':'Arial', 'size':'24'}
annotate_font = {'fontname':'Arial', 'size':'20'}
tick_size='24'
tick_weight='bold'
_dpi=300

# plotting the results extracted from the programs XDS and SHELXD/E
def plot_results(directory, name, workList, program, parameter, dose, out):
    for p in program:
# load the extracted values
        print p, directory+name+'_'+p+'.pic'
        if os.path.exists(directory+name+'_'+p+'.pic'):
            with open(directory+name+'_'+p+'.pic', "r") as f:
                params=pickle.load(f)
        else:
            print " DEB values still need to be extraxted."
# plot the values for a certain variable
        for key in params.keys():
            if key in parameter:
                y=params[key]
                print y
            else:
                print "%s is not in parameter list" %key
                continue
            x=numpy.linspace(1, len(y), len(y))
            fig = pylab.figure(figsize=(10,6), dpi = _dpi)
            ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])
            if "acc" in name:
                ax.plot(x, y,  "r^-", label='CFOM')
            else:
              #  ax.plot(x, y,  "b^-", label=r'$'+key.split()[0]+'$')
                ax.plot(x, y,  "b^-", label=r'anomalous signal')
            ax.set_xlabel(r"oscillation range [$^\circ$]", **axis_font)
            test=r"%s" %key
            ax.set_ylabel(test, **axis_font)
            fig.text(0.9, 0.053, r'x360$^\circ$', **axis_font)
            if max(y)<1.5:
                ax.set_ylim(0, max(y)+0.3)
            elif max(y)>50:
                ax.set_ylim(-0.5, max(y)+20)
            else:
                ax.set_ylim(0, max(y)+2.5)
            if min(y)<0.1:
                ax.set_ylim(-0.1, max(y)+1)
            ax.set_xlim(0.5, max(x)+0.5)
            ax.set_ylim(0.1, 80)
            ax2 = ax.twiny()
            ax2.set_xlabel("dose [MGy]", labelpad=10, **axis_font)
            _dose=numpy.linspace(dose, max(x)*dose, 5)
            _dose=[round(d, ndigits=2) for d in _dose]
            ax2.set_xlabel("dose [MGy]", labelpad=10, **axis_font)
            ax2.set_xlim(0.5*dose, len(x)*dose+0.5*dose)
            ax2.set_xticks(_dose)
            ax.legend(loc=2, prop={'size':20})
            ax2.legend(loc=1, prop={'size':20})
            if key.split("/")!=[]:
                key=key.split("/")[0]
            for item in (ax.get_xticklabels() + ax2.get_xticklabels()+ ax.get_yticklabels()):
                item.set_fontsize(tick_size)
                item.set_weight(tick_weight)
            _label=[key[0:2] if key[0]!='$' else key[1]]
            print " DEB Figure is saved to %s" % out[0]+name+"_"+_label[0]+out[1]
            fig.savefig(out[0]+name+"_"+_label[0]+out[1], bbox_inches='tight', dpi=_dpi, transparent=True)

# as sitcom requires an extra axis, it is plotted separatedly
def plot_sitcom_results(directory, name, workList, parameter, dose, out):
    #load the sitcom results
    if os.path.exists(directory+name+'_sitcom.pic'):
        with open(directory+name+'_sitcom.pic', "r") as f:
            params=pickle.load(f)
    else:
        print " DEB Sitcom values still need to be extracted."
```

Plot_all

```
    # plot the sitcom results
    y=params['no. of found sites']
    rmsd=params['rmsd [a.u.]']
    x=numpy.linspace(1, len(y), len(y))
    fig = pylab.figure(figsize=(10,6), dpi = _dpi)
    ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])
    if "acc" in name:
        ax.plot(x, y, "r^-", label="found sulfur sites")
    else:
        ax.plot(x, y, "b^-", label="found sulfur sites")
    ax.set_xlabel(r"oscillation range [$^\circ$]", **axis_font)
    fig.text(0.9, 0.055, r'x360$^\circ$', **axis_font)
    ax.set_ylabel(r"no. of correct sulfur sites", **axis_font)
    ax.legend(loc=3, prop={'size':20})
    ax2=ax.twinx()
    ax2.plot(x, rmsd, "k^:", label="rmsd")
    ax2.set_ylabel(r"rmsd", **axis_font)
    ax2.legend(loc=4, prop={'size':20})
    ax.set_xlim(0.5, max(x)+0.5)
    ax.set_ylim([-0.5, 11])
    ax2.set_ylim([-0.05, max(rmsd)+0.4])
    for item in (ax.get_xticklabels() + ax2.get_yticklabels()+ ax.get_yticklabels()):
        item.set_fontsize(tick_size)
        item.set_weight(tick_weight)
    print "DEB Figure is saved to %s" % out[0]+name+"_sitcom"+out[1]
    pylab.savefig(out[0]+name+"_si"+out[1], bbox_inches='tight', dpi=_dpi, transparent=True)


# parameters for the plotting functions
directory="/Volumes/New_Volume/PETRA_data/20150927/PROCESSED_DATA/lys_10/"
name="lys_10"
workList=[]
for i in xrange(1,16):
    workList.append(directory+"xds_lys_10_"+str(i)+"_1/")
program=["SHELXD"] # can also be XDS, SHELXC, SHELXE
parameter=['CFOM \%'] # SHELXD parameter
#parameter= ['I/$\sigma$ [a.u.]', '$R_{meas}$ [\%]','anomalous signal [a.u.]', '$CC_{1/2} [\%]$'] # XDS parameter
#parameter= [CC [\%]', 'CC_i [\%]', 'res', 'res_i'] % SHELXE parameter
#parameter= ['no. of found sites', 'rmsd [a.u.]'] % sitcom parameter
dose= 0.61
out=["/Users/storm/Documents/Experiments_2015/Anomalous_Diffraction/20150927/lys_10/results/", ".pdf"]

#function calls
plot_results(directory, name, workList, program, parameter, dose, out)
#plot_sitcom_results(directory, name, workList, parameter, dose, out)

```

check_ano_peaks

```
import matplotlib
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
import numpy
from matplotlib import rc
import os
import string
import pylab
from collections import Counter

# define fonts and plotting options
rc('font', weight='bold', **{'family':'serif','serif':['Computer Modern Roman']})
rc('text', usetex=True)
tick_size='20'
tick_weight='bold'
_dpi=300

def write_pathList(path, folder_prefix, max_turn):
        pathList=[]
        for i in xrange(1, max_turn+1):
                pathList.append(path+folder_prefix+str(i)+"_1/")
        return pathList

# run anode
def check_ano_peak(pathList, pdb_path, name, new_name):
    for path in pathList:
        os.chdir(path)
        print path
        #consider that the pdb file has to be named as new_name.pdb
        os.system("cp "+pdb_path+ " .")
        print name+"_fa.hkl"
        print new_name+"_fa.hkl"
        os.system("cp "+name+"_fa.hkl "+new_name+"_fa.hkl")
        os.system("/Users/storm/Applications/anode -b10.0 -d1.0 -h80 -m20 -r5.0 -n99.0 -s4.0 "+new_name+" >anode.log")

# extract the anomalous sulfur peak heights from the output files
def analyse_ano_peak(pathList, new_name, chain_id):
    peak_list=[]
    for path in pathList:
        ha_list=[]
        with open(path+new_name+".lsa", "r") as f:
            lines=f.readlines()
        for line in lines:
            for _id in chain_id:
                if (string.find(line, _id)>=0) and len(line.split())>4:
                    _chain_id= line.split()[-1].split(":")[1]
                    _peak=line.split()[4]
                    ha_list.append((_chain_id, _peak))
        if ha_list!=[]:
            peak_list.append(ha_list)
        else:
            print path+new_name+".lsa"
    return peak_list

def plot_ano_peak(peak_list, max_turn, dose, path):
    _pk={}
    # sort the different anomalous peak heights in plottable lists
    _identifier=sorted(list(set([peak_list[0][i][0] for i in xrange(len(peak_list[0]))])))
    for _id in _identifier:
        _pk.update({_id:[]})
    for ha_list in peak_list:
        _ha_sorted=sorted(ha_list, key=lambda tup:tup[0])
        for _id in _identifier:
            _n=[]
            for el in _ha_sorted:
                if el[0]==_id:
                    _n.append(el[1])
            _pk[_id].append(_n)
    _col=max([len(p) for p in peak_list])
    _arr=numpy.zeros(shape=(28, 28))
    _r_counter=0
    _add=[]
    r=0
    print len(_pk.keys())
    for key in _pk.keys():
        c=0
        for el in _pk[key]:
            if el==[]:
                _arr[r,c]=0
                c=c+1
            else:

                _arr[r,c]=float(el[0])
                c=c+1
        r=r+1
    all_ind=[]
    for p in _pk.keys():
        all_ind.append(p)
```

```
    # start plotting
    fig1 = pylab.figure(figsize=(10,7), dpi = _dpi)
    ax = fig1.add_axes([0.08, 0.15, 0.73, 0.73])
    pylab.subplots_adjust(left=0.1, right=1.1, bottom=0.1, top=0.9, wspace=0.2, hspace=1)
    _to_plot=[]
    colors = len(all_ind)
    cm = plt.get_cmap('jet')
    ax.set_color_cycle([cm(1.*i/colors) for i in range(colors)])
    x=numpy.arange(1, max_turn+1)
    _dose=numpy.linspace(0, dose, num=int(max_turn)/3)
    _dose=[round(i, ndigits=2) for i in _dose]
    for i in xrange(int(_col)):
        check=[_arr[i][k] for k in xrange(0, max_turn) if _arr[i][k]!=0]
        if len(check)>=(0.01*(max_turn+1)):
            _to_plot.append((_arr[i][:], all_ind[i]))
    _to_plot=sorted(_to_plot, key=lambda tup:tup[1])
    for pl in _to_plot:
        ax.plot(x, pl[0][0:max_turn], "^-", label=pl[1])
    ax.set_xlabel(r"oscillation range [$^\circ$]", fontsize=24)
    ax.set_ylabel(r"average anomalous density [$\sigma$]", fontsize=24)
    fig1.text(0.81, 0.11, r'x360$^\circ$', fontsize=24)
    ax2 = ax.twiny()
        ax.set_xlim(0.5, max_turn+0.5)
        ax.set_ylim(-0.5, 24)
        ax2.set_xlabel("dose [MGy]", labelpad=10, fontsize=24)
        _dose=numpy.linspace(dose, max(x)*dose, 5)
        _dose=[round(d, ndigits=2) for d in _dose]
        ax2.set_xlim(0.5*dose, len(x)*dose+0.5*dose)
        ax2.set_xticks(_dose)
    for item in (ax.get_xticklabels() + ax.get_yticklabels()+ax2.get_xticklabels()):
            item.set_fontsize(24)
            item.set_weight('medium')
    ax.legend(bbox_to_anchor=(1.01, 1.0), loc=2, borderaxespad=0., prop={'size':15})
    pylab.savefig("/Users/storm/Documents/Experiments_2015/Anomalous_Diffraction/20150927/lys_10/"+name+"_ano_peak.pdf", dpi=_dpi)

# parameters
path="/Volumes/New_Volume/PETRA_data/20150927/PROCESSED_DATA/lys_10/"
folder_prefix="xds_lys_10_"
max_turn=15
name="lys_10"
new_name="lys_10_ano_peak"
pdb_path="/Users/storm/Documents/Experiments_2015/Anomalous_Diffraction/20150927/lys_10/lys_10_ano_peak.pdb"
chain_id=['CYS', 'MET']
dose=0.61
# function calls
pathList=write_pathList(path, folder_prefix, max_turn)
check_ano_peak(pathList, pdb_path, name, new_name)
peak_list=analyse_ano_peak(pathList, new_name, chain_id)
plot_ano_peak(peak_list,  max_turn, dose,  path)
```

# Appendix B

# Results of the SAD experiments and analysis of the average signed anomalous differences

In the following, the plots performed in the main part for data set 150421_tha1 can be found here for the data sets 150421_tha4, 150421_tha6, 150927_tha1 and 150927_tha3. In general, the plots for the wedges can be identified by the blue lines, the ones for the wedges by red lines. The chapter starts with the SHELXD, SITCOM and ANODE results which are used to judge the substructure quality. In the next section, the histograms including the fit with a Gaussian distribution for the first and the last wedge can be found for the data processed accumulatively and in wedges. This is followed by the fits of the normalized histograms for all wedges and accumulated data with a normal distribution in the following section. The chapter ends with the plots of the σ values of the normalized distributions for the different resolution shells.

## B.1 SHELXD, SITCOM and ANODE plots

The data were processed as described in section 3.2.4. To be able to deviate the best substructure, the results from SHELXD and sitcom have to be seen in context. A compact summary and an interpretation of the data can be found in section 4.5.

### B.1.1 Data set 150421_tha4

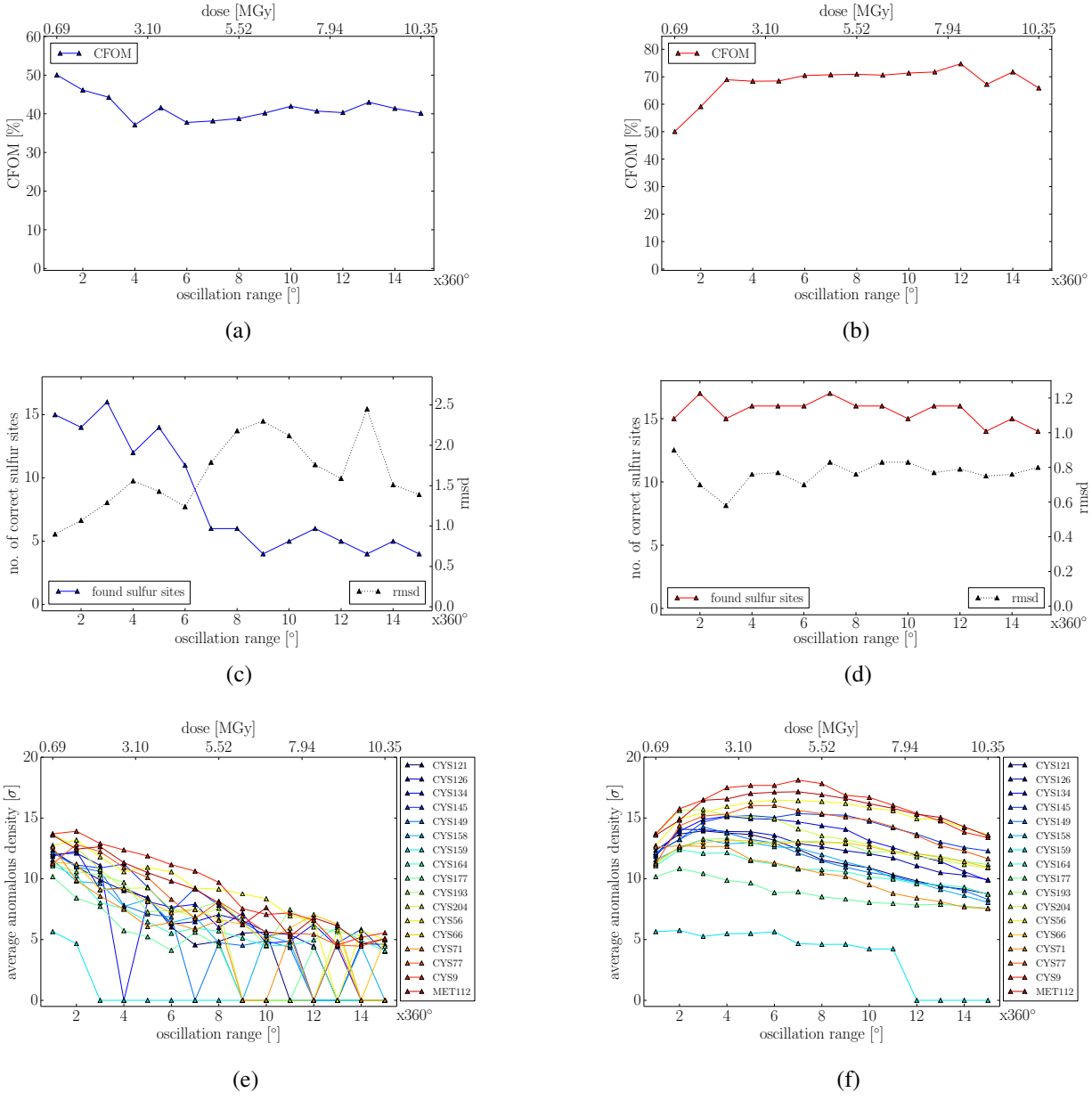



Figure B.1: Plots of a) the CFOM obtained by SHELXD via processing the data in 360° wedges b) and by processing the accumulated data, c) the number of correct heavy atom sides and the corresponding rmsd to the reference structure obtained by SITCOM via processing the data in wedges, d) and for the accumulated data; e) shows the anomalous peak heights of the sulphur atoms calculated with anode for the data processed in wedges and f) the same for the accumulated data.

## B.1.2 Data set 150421_tha6

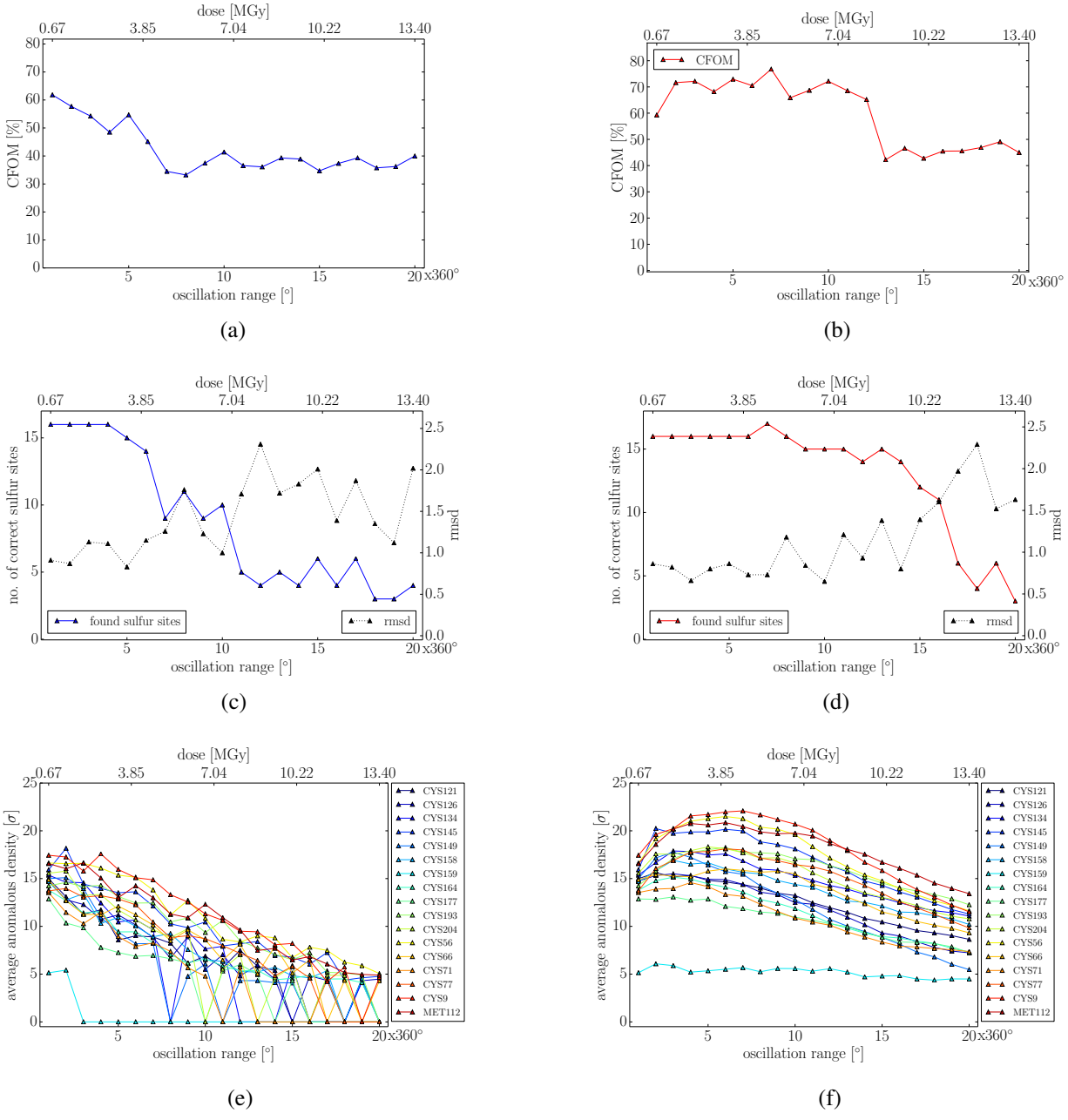



Figure B.2: Plots of a) the CFOM obtained by SHELXD via processing the data in 360° wedges b) and by processing the accumulated data, c) the number of correct heavy atom sides and the corresponding rmsd to the reference structure obtained by SITCOM via processing the data in wedges, d) and for the accumulated data; e) shows the anomalous peak heights of the sulphur atoms calculated with anode for the data processed in wedges and f) the same for the accumulated data.

### B.1.3 Data set 150927_tha1

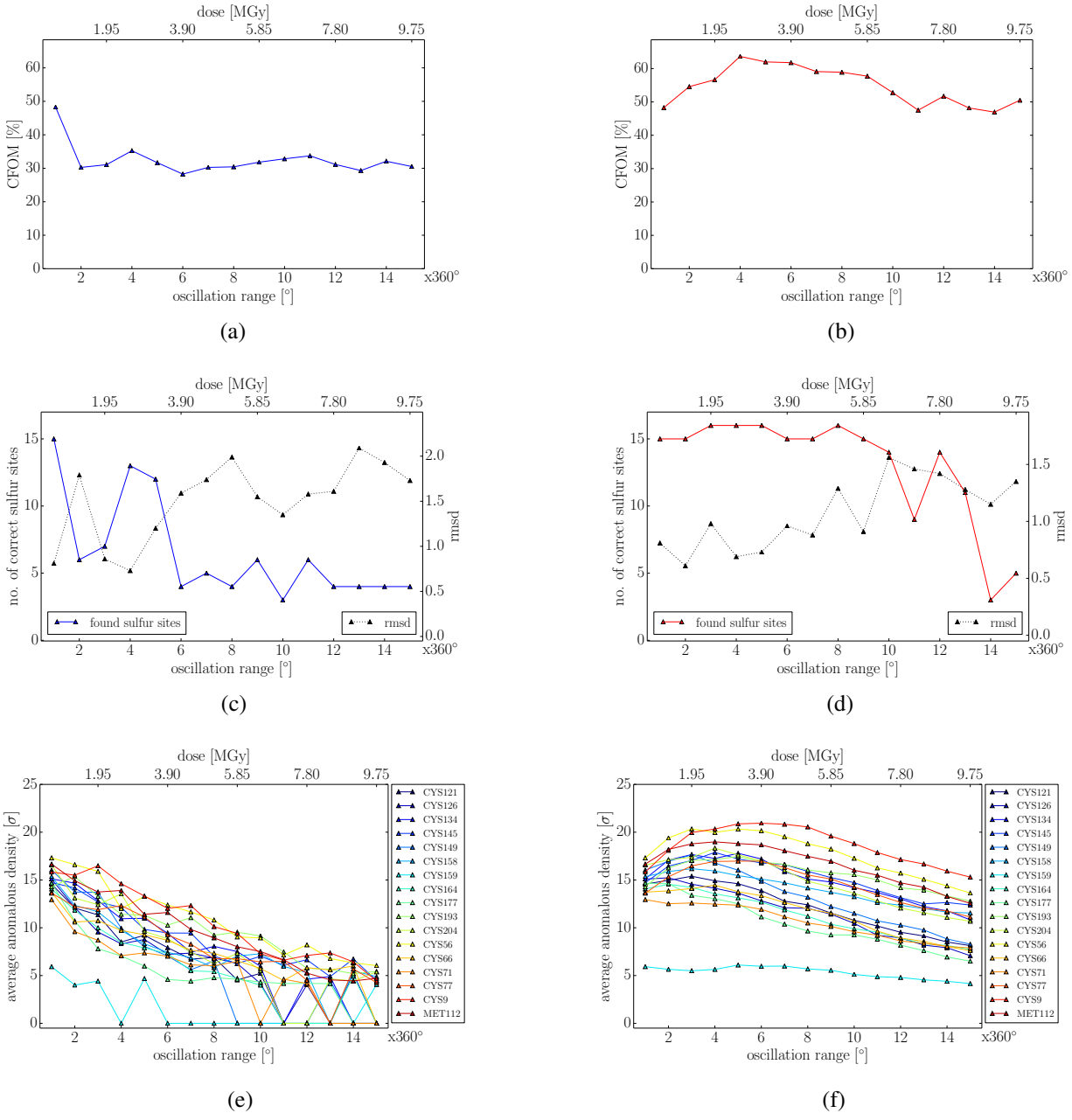



Figure B.3: Plots of a) the CFOM obtained by SHELXD via processing the data in 360° wedges b) and by processing the accumulated data, c) the number of correct heavy atom sides and the corresponding rmsd to the reference structure obtained by SITCOM via processing the data in wedges, d) and for the accumulated data; e) shows the anomalous peak heights of the sulphur atoms calculated with anode for the data processed in wedges and f) the same for the accumulated data.

### B.1.4 Data set 150927_tha3

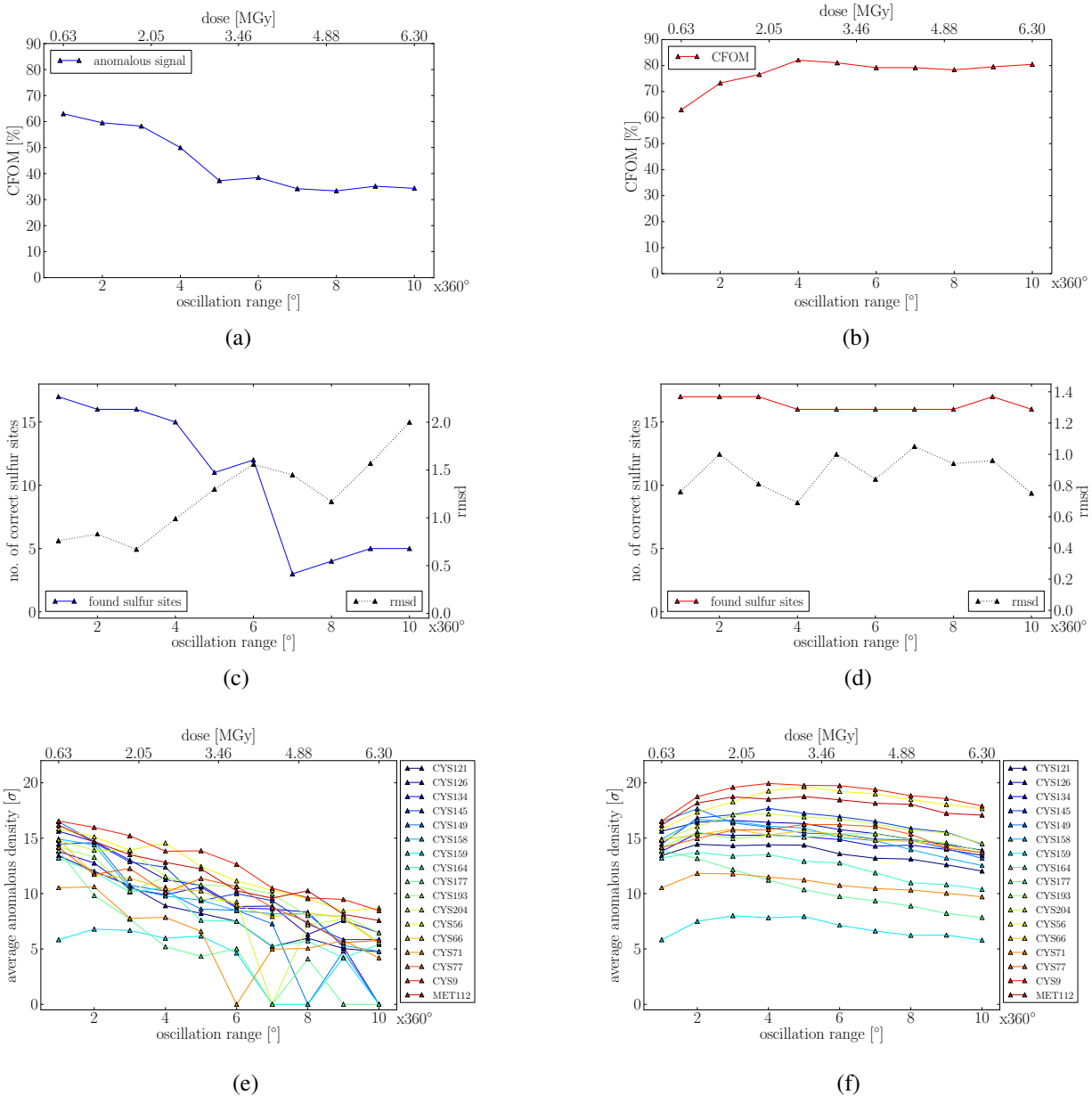



Figure B.4: Plots of a) the CFOM obtained by SHELXD via processing the data in 360° wedges b) and by processing the accumulated data, c) the number of correct heavy atom sides and the corresponding rmsd to the reference structure obtained by SITCOM via processing the data in wedges, d) and for the accumulated data; e) shows the anomalous peak heights of the sulphur atoms calculated with anode for the data processed in wedges and f) the same for the accumulated data.

## B.2 Histograms of the average signed anomalous differences

The histograms of the average signed anomalous differences for chosen wedges and accumulated data including a Gaussian fit are calculated with the program in appendix C. An analysis can be found in section 5.3.

### B.2.1 Data set 150421_tha4

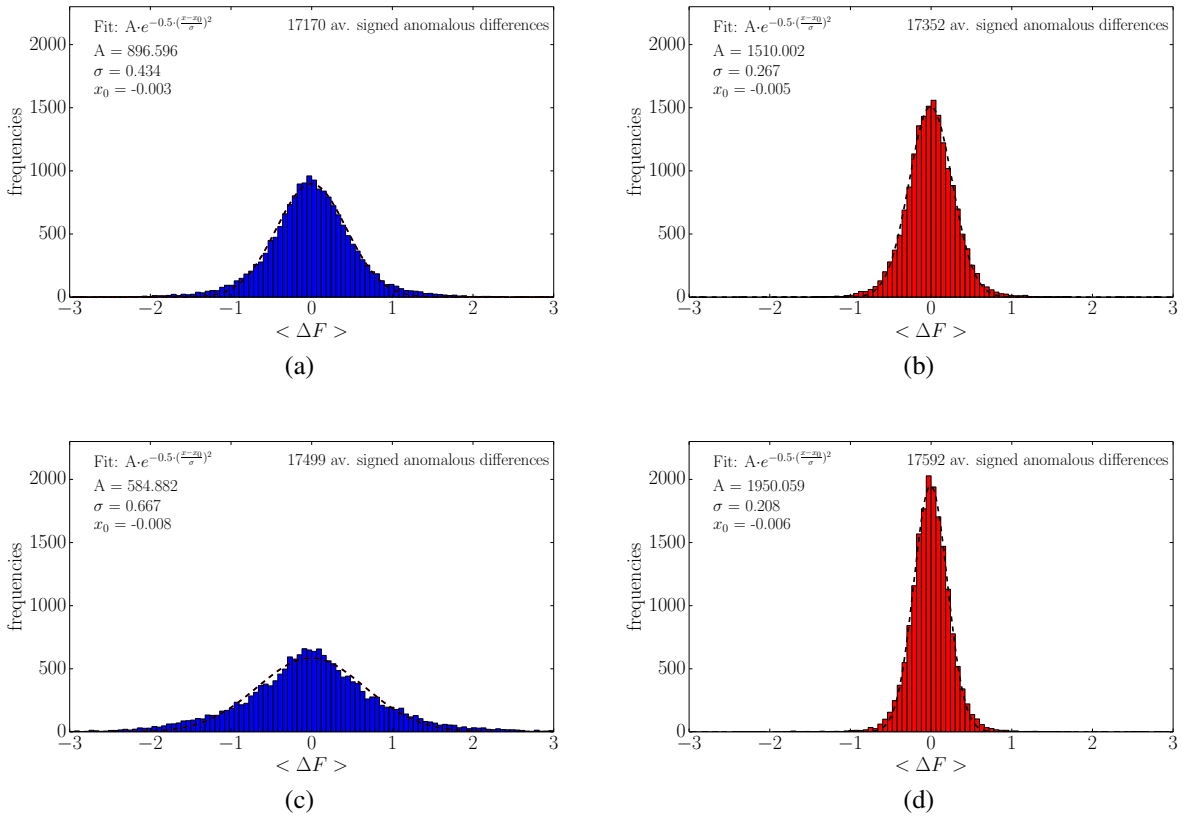



Figure B.5: Analysis of the average signed anomalous differences, taking only the acentric unique reflections into account. Fig. 5.6 a) represents the histogram of the wedge-wise processed data based on the first 360° of data, b) shows the histogram of 5x360° accumulated data, c) depicts the histogram of the last 360° of data (15th turn) and d) shows the histogram based on the average signed anomalous differences accumulated in 15 turns. All histograms are calculated with the same 100 bins.

### B.2.2 Data set 150421_tha6

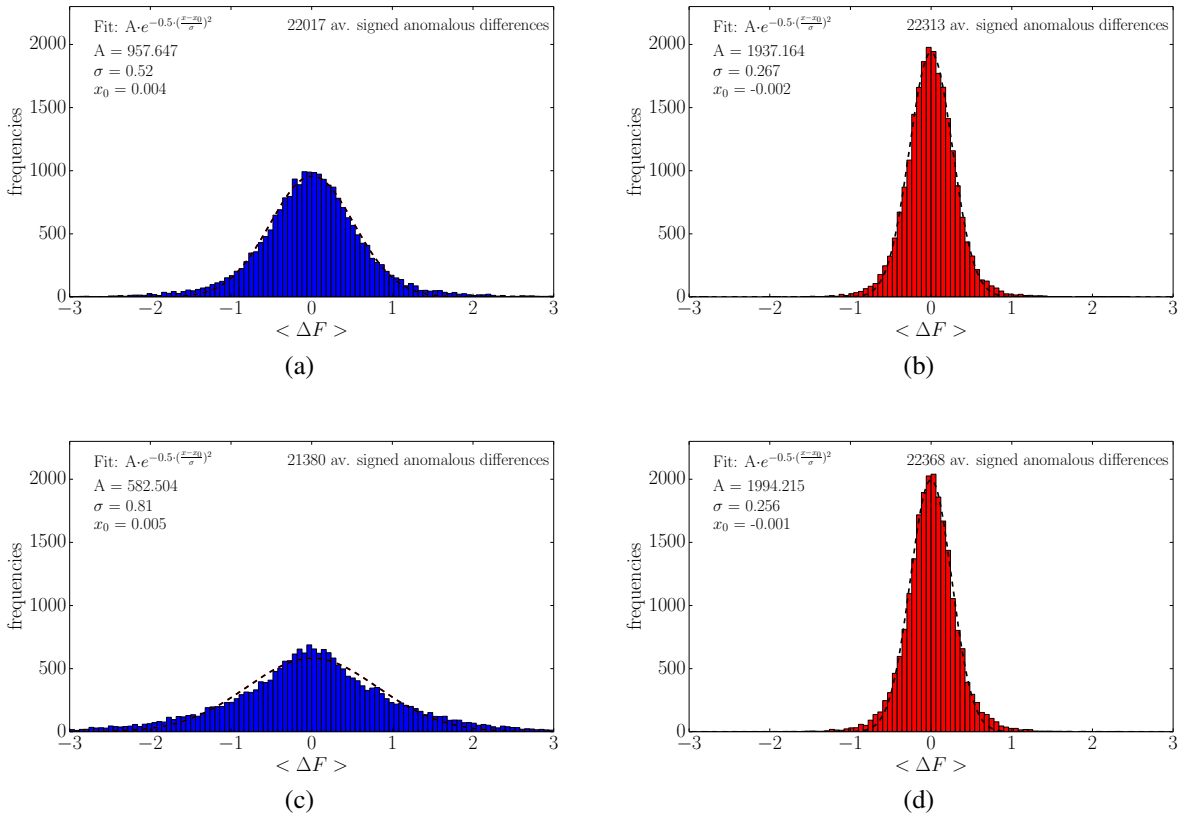



Figure B.6: Analysis of the average signed anomalous differences, taking only the acentric unique reflections into account. Fig. 5.6 a) represents the histogram of the wedge-wise processed data based on the first 360° of data, b) shows the histogram of $10x360°$ accumulated data, c) depicts the histogram of the last 360° of data (20th turn) and d) shows the histogram based on the average signed anomalous differences accumulated in 20 turns. All histograms are calculated with the same 100 bins.

### B.2.3 Data set 150927_tha1

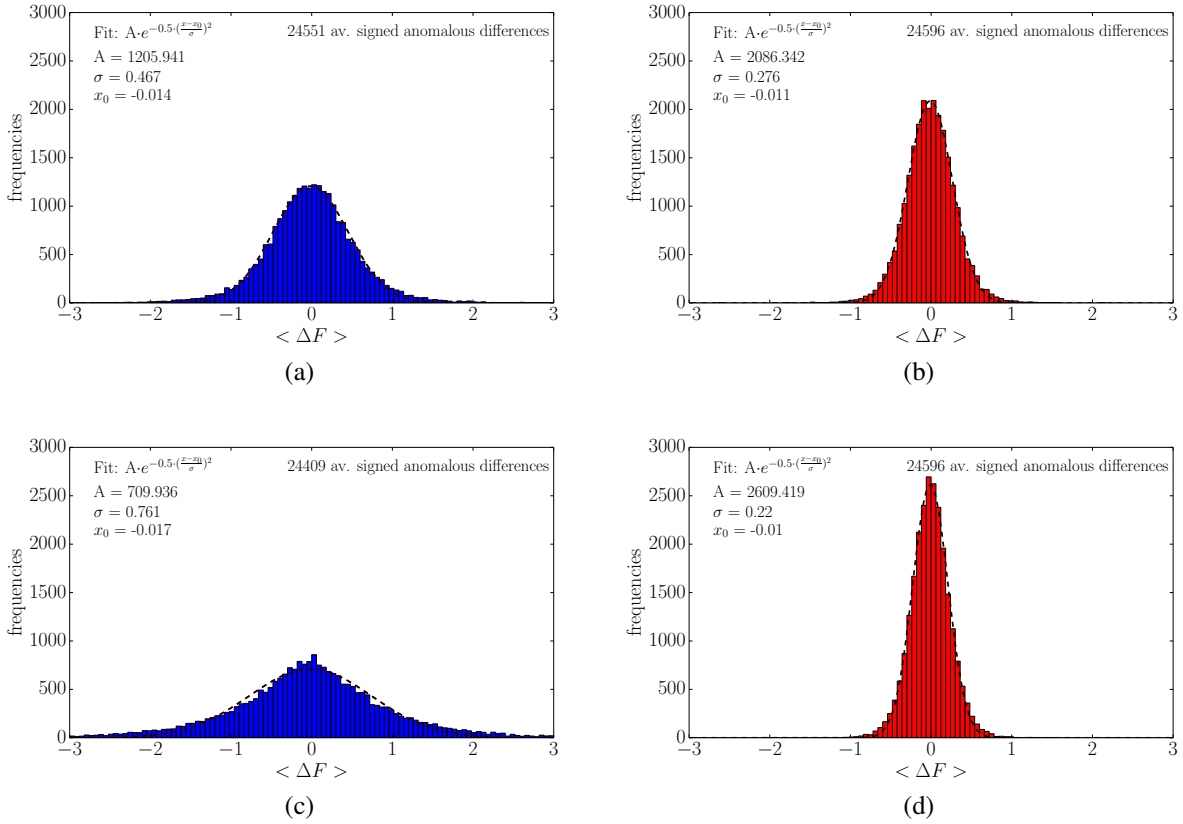



Figure B.7: Analysis of the average signed anomalous differences, taking only the acentric unique reflections into account. Fig. 5.6 a) represents the histogram of the wedge-wise processed data based on the first 360° of data, b) shows the histogram of 5x360° accumulated data, c) depicts the histogram of the last 360° of data (15th turn) and d) shows the histogram based on the average signed anomalous differences accumulated in 15 turns. All histograms are calculated with the same 100 bins.

### B.2.4 Data set 150927_tha3

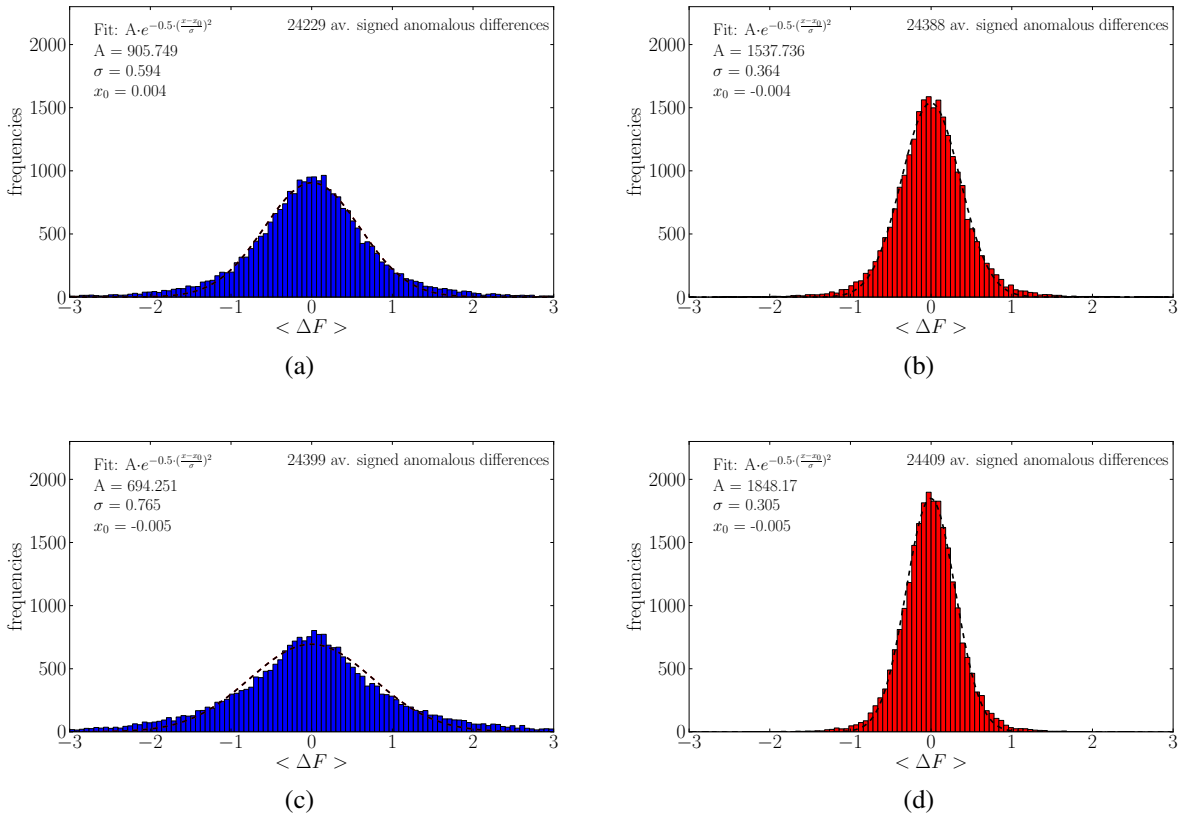
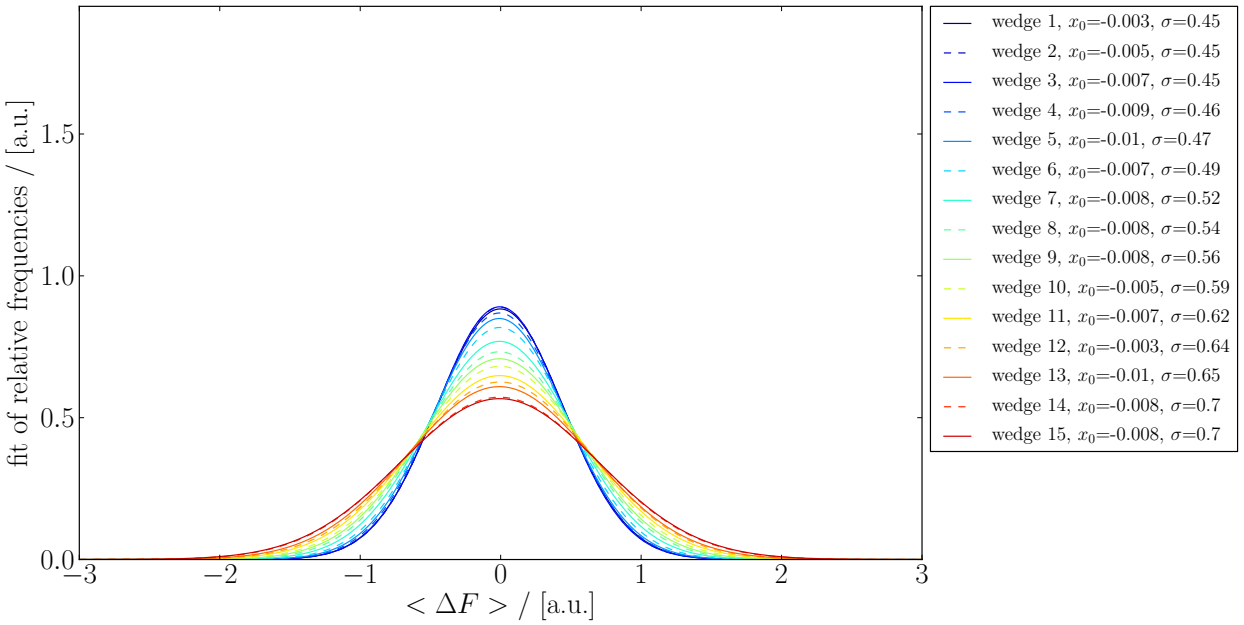



Figure B.8: Analysis of the average signed anomalous differences, taking only the acentric unique reflections into account. Fig. 5.6 a) represents the histogram of the wedge-wise processed data based on the first 360° of data, b) shows the histogram of 5x360° accumulated data, c) depicts the histogram of the last 360° of data (10th turn) and d) shows the histogram based on the average signed anomalous differences accumulated in 10 turns. All histograms are calculated with the same 100 bins.

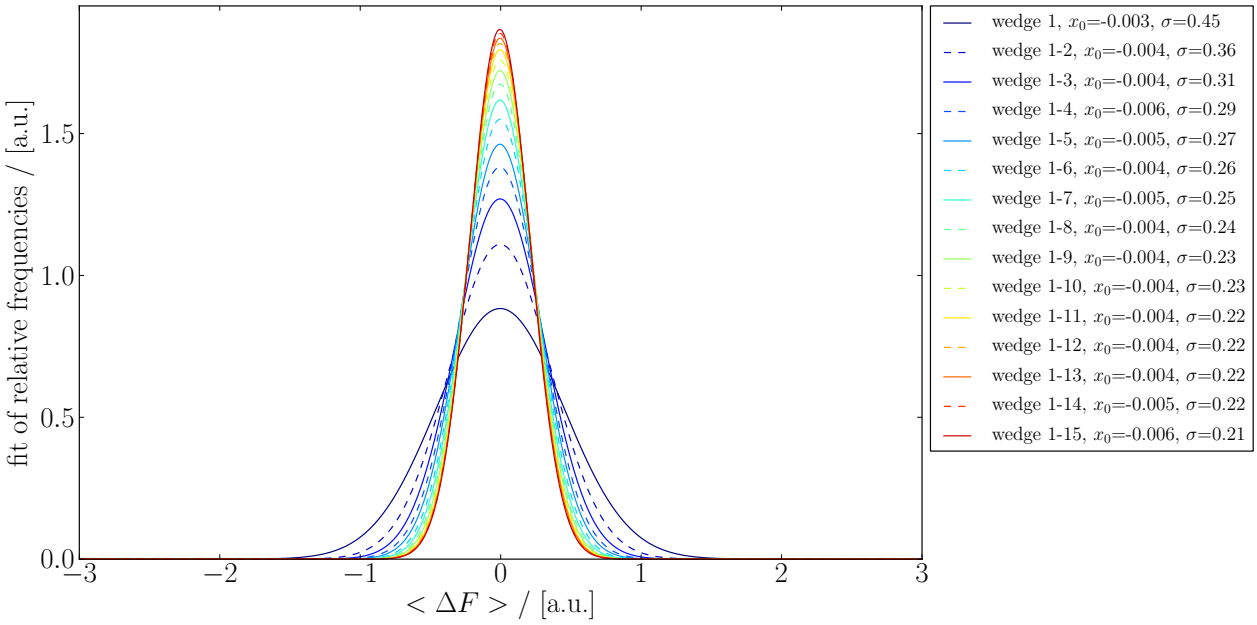
## B.3 Fits of the normalized histograms

The histograms of the average signed anomalous differences calculated and plotted with the program in appendix C were normalized and fitted with normal distributions for both the wedges and the accumulated data. The analysis and discussion of these plots can be found in section 5.3.

### B.3.1 Data set 150421_tha4



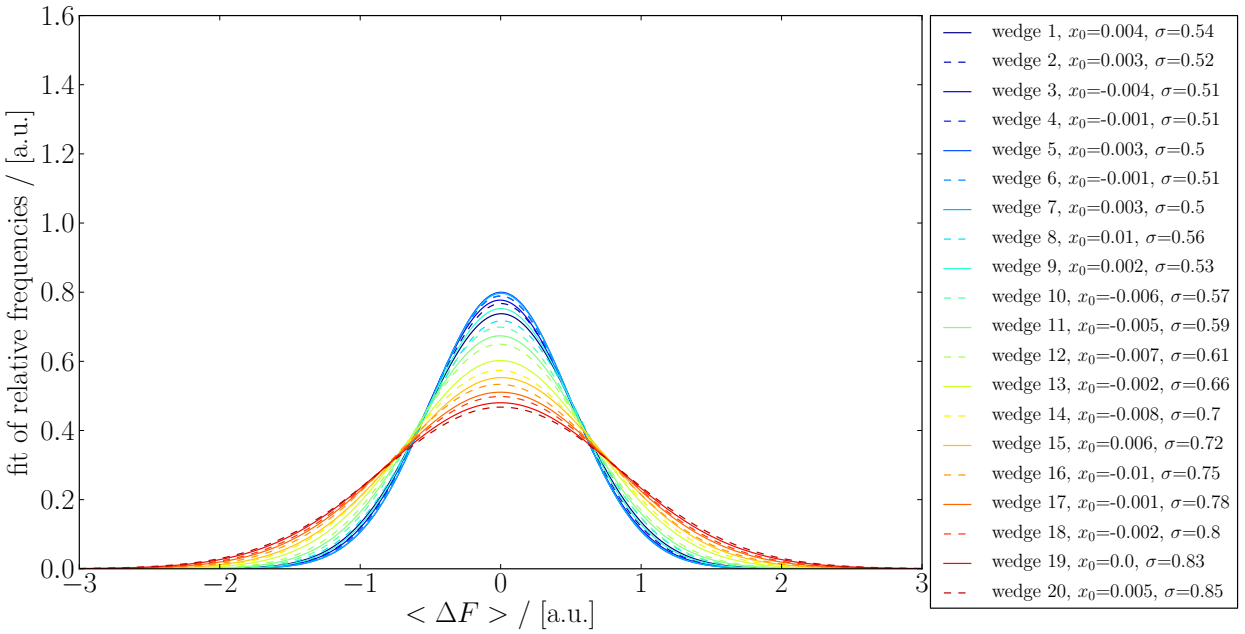

(a)



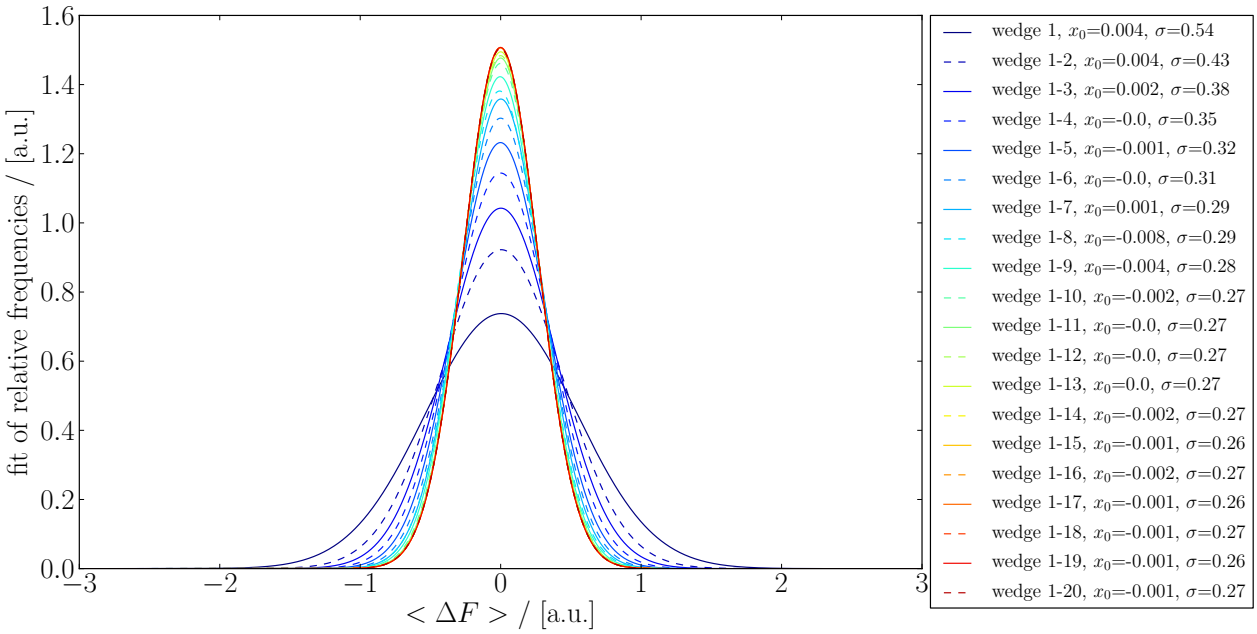

(b)

Figure B.9: The normalized histograms of the signed averaged anomalous differences were fitted with normal distributions for a) the wedge-wise processed data and b) for the accumulated data.

### B.3.2 Data set 150421_tha6



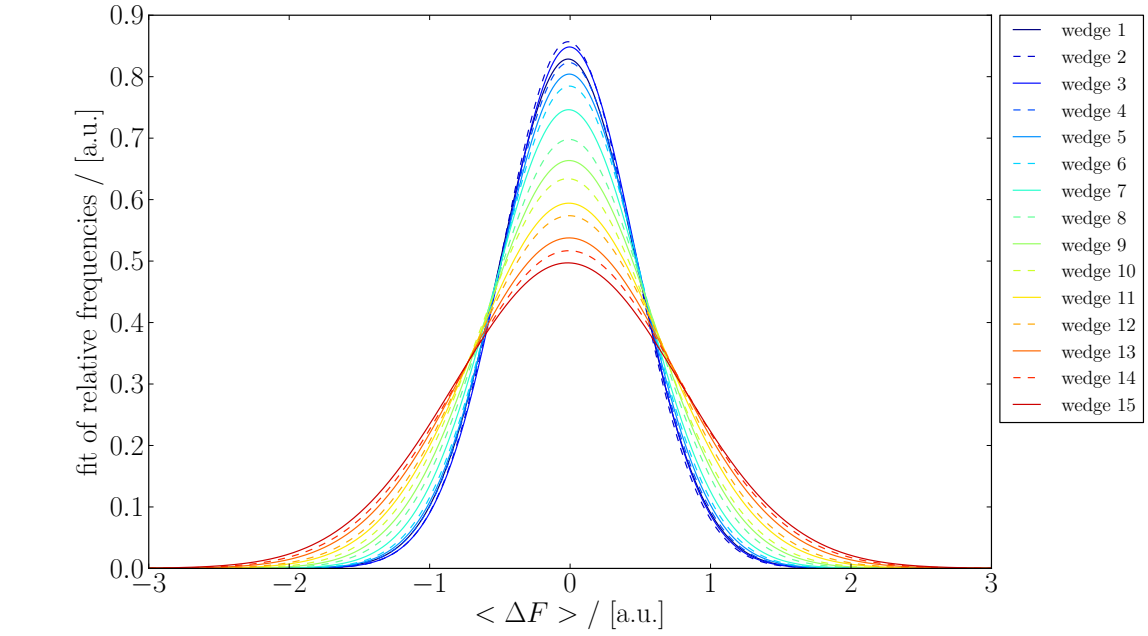

(a)



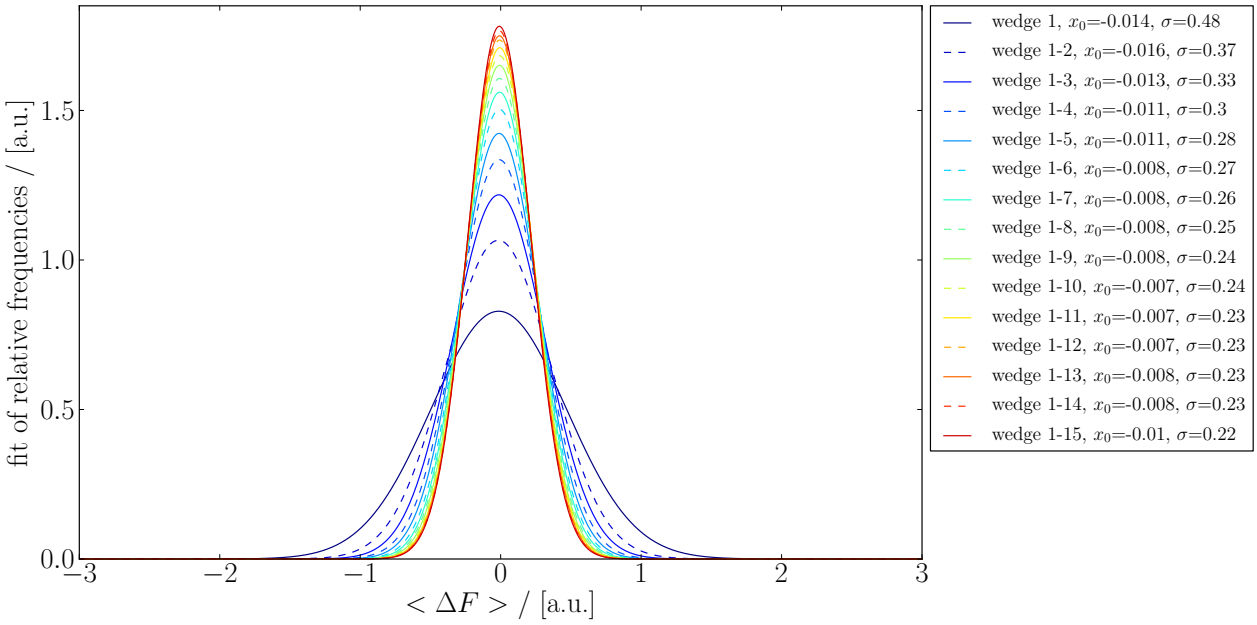

(b)

Figure B.10: The normalized histograms of the signed averaged anomalous differences were fitted with normal distributions for a) the wedge-wise processed data and b) for the accumulated data.

### B.3.3 Data set 150927_tha1




(a)




(b)

Figure B.11: The normalized histograms of the signed averaged anomalous differences were fitted with normal distributions for a) the wedge-wise processed data and b) for the accumulated data.

### B.3.4 Data set 150927_tha3

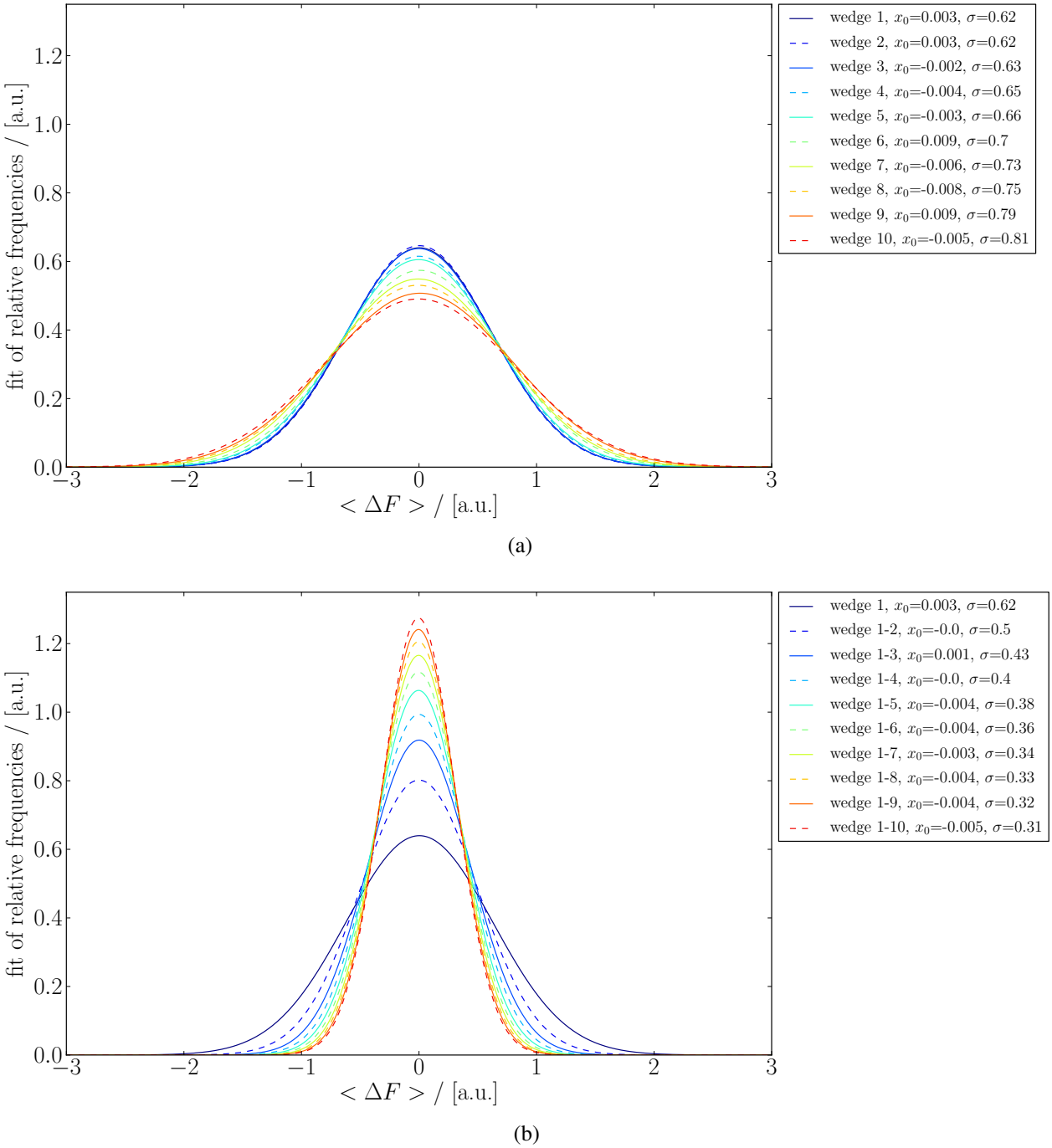



Figure B.12: The normalized histograms of the signed averaged anomalous differences were fitted with normal distributions for a) the wedge-wise processed data and b) for the accumulated data.

## B.4 A metric for determining the best substructure

The σ values of the normal distributions presented in the previous section are plotted with the method *plot_AnoDif1_normed_fits* of the plotting class in appendix C for different resolution shells and all data. The latter is fitted with the method *plot_sigma_fit* by two lines. The results are discussed in section 5.1.

### B.4.1 Data set 150421_tha4

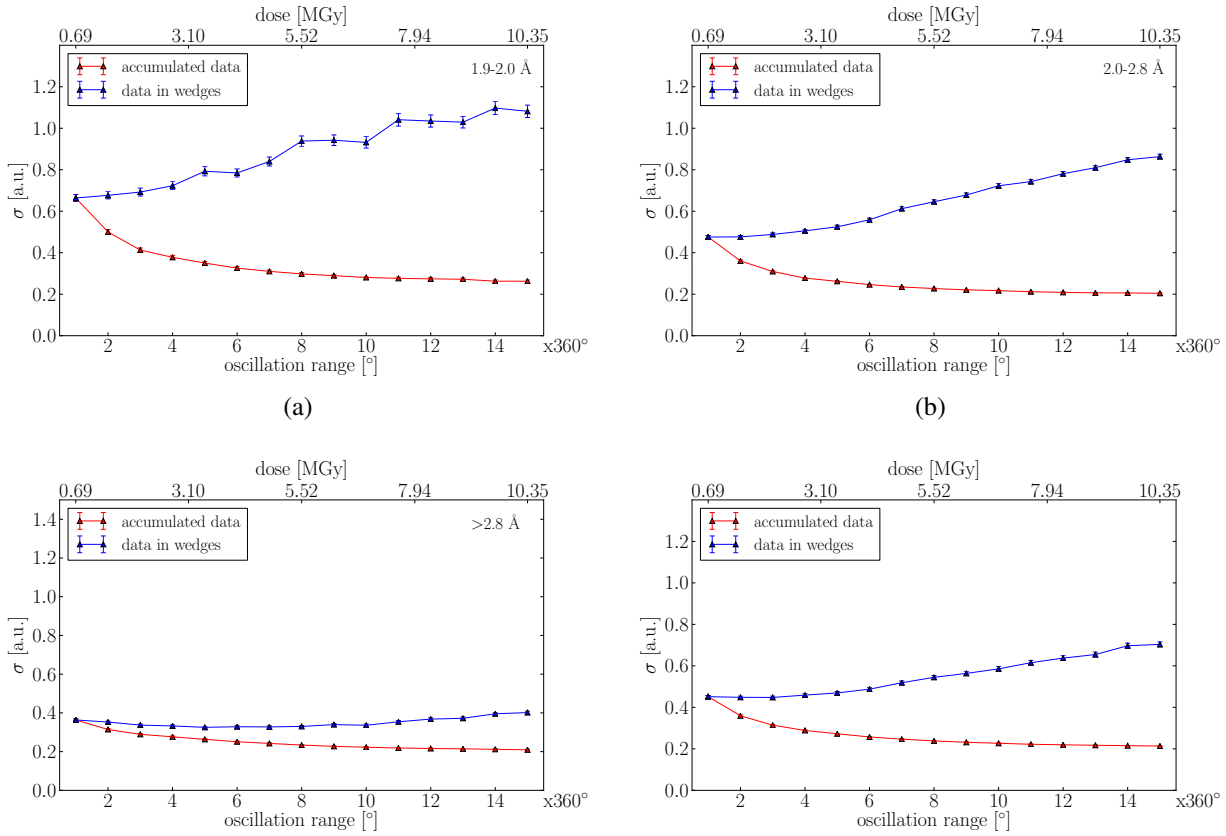



Figure B.13: Plot of the fitting parameter σ for the normal distributions fitting the normalized histograms of the average signed anomalous differences against the number of 360° turns and dose, depending on the resolution of the acentric reflections. The red curves stand for accumulated, the blue ones for wedge-wise processed data. The single plots are a) for data with a resolution of 1.9 – 2.0 Å, b) for data with a resolution of 2.0 – 2.8 Å, c) for data with a resolution of > 2.8 Å. Fig. B.13 d) is based on all average anomalous differences.

### B.4.2 Data set 150421_tha6

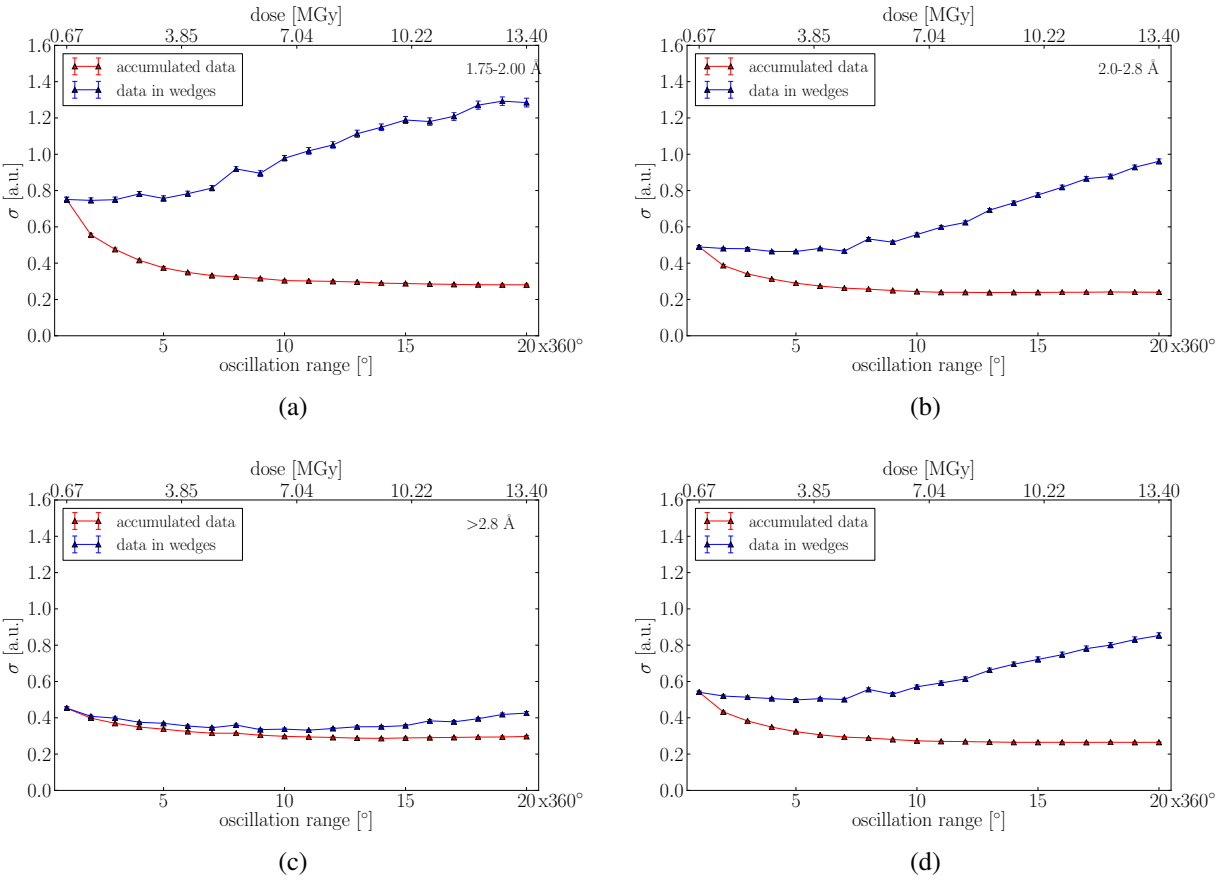



Figure B.14: Plot of the fitting parameter σ for the normal distributions fitting the normalized histograms of the average signed anomalous differences against the number of 360° turns and dose, depending on the resolution of the acentric reflections. The red curves stand for accumulated, the blue ones for wedge-wise processed data. The single plots are a) for data with a resolution of 1.75 – 2.0 Å, b) for data with a resolution of 2.0 – 2.8 Å, c) for data with a resolution of > 2.8 Å. Fig. B.14 d) is based on all average anomalous differences.

### B.4.3 Data set 150927_tha1

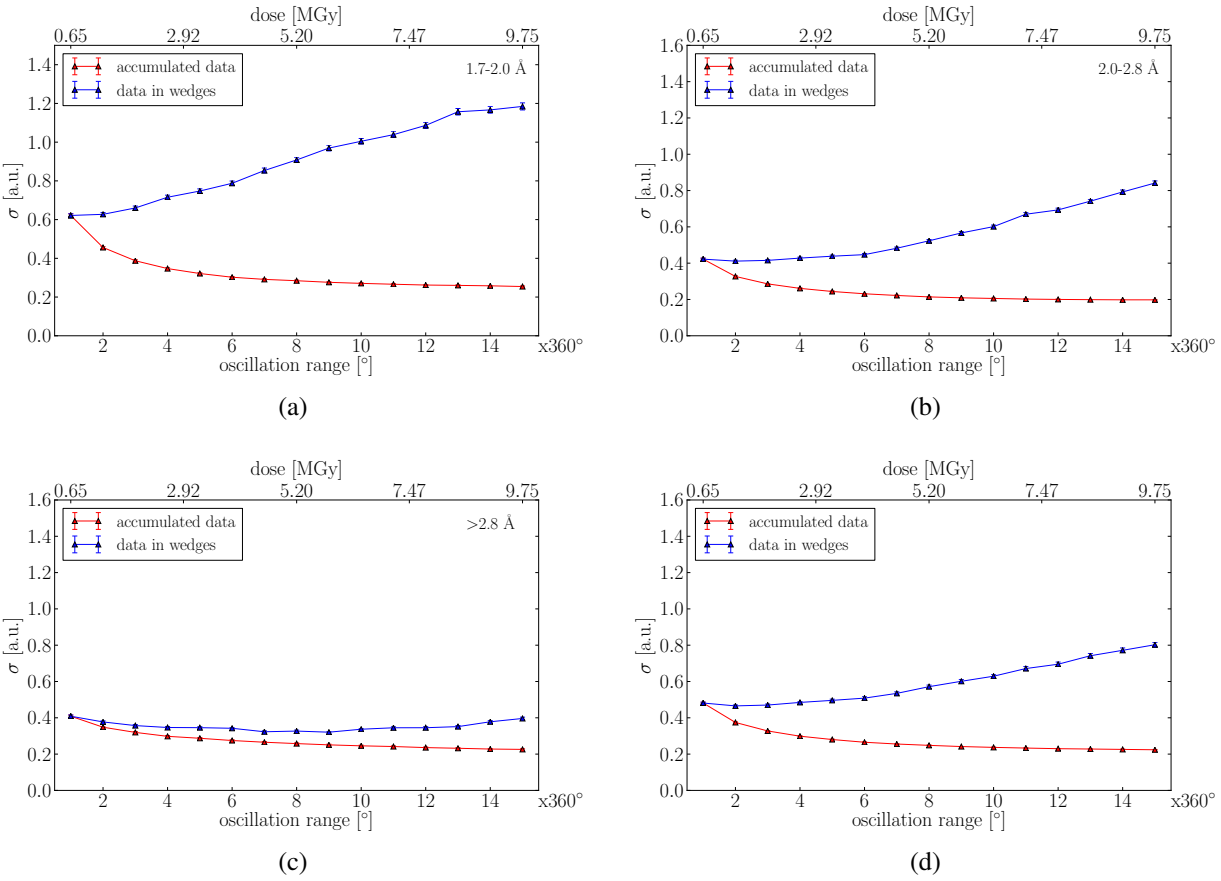



Figure B.15: Plot of the fitting parameter σ for the normal distributions fitting the normalized histograms of the average signed anomalous differences against the number of 360° turns and dose, depending on the resolution of the acentric reflections. The red curves stand for accumulated, the blue ones for wedge-wise processed data. The single plots are a) for data with a resolution of 1.7 – 2.0 Å, b) for data with a resolution of 2.0 – 2.8 Å, c) for data with a resolution of > 2.8 Å. Fig. B.15 d) is based on all average anomalous differences.

### B.4.4 Data set 150927_tha3

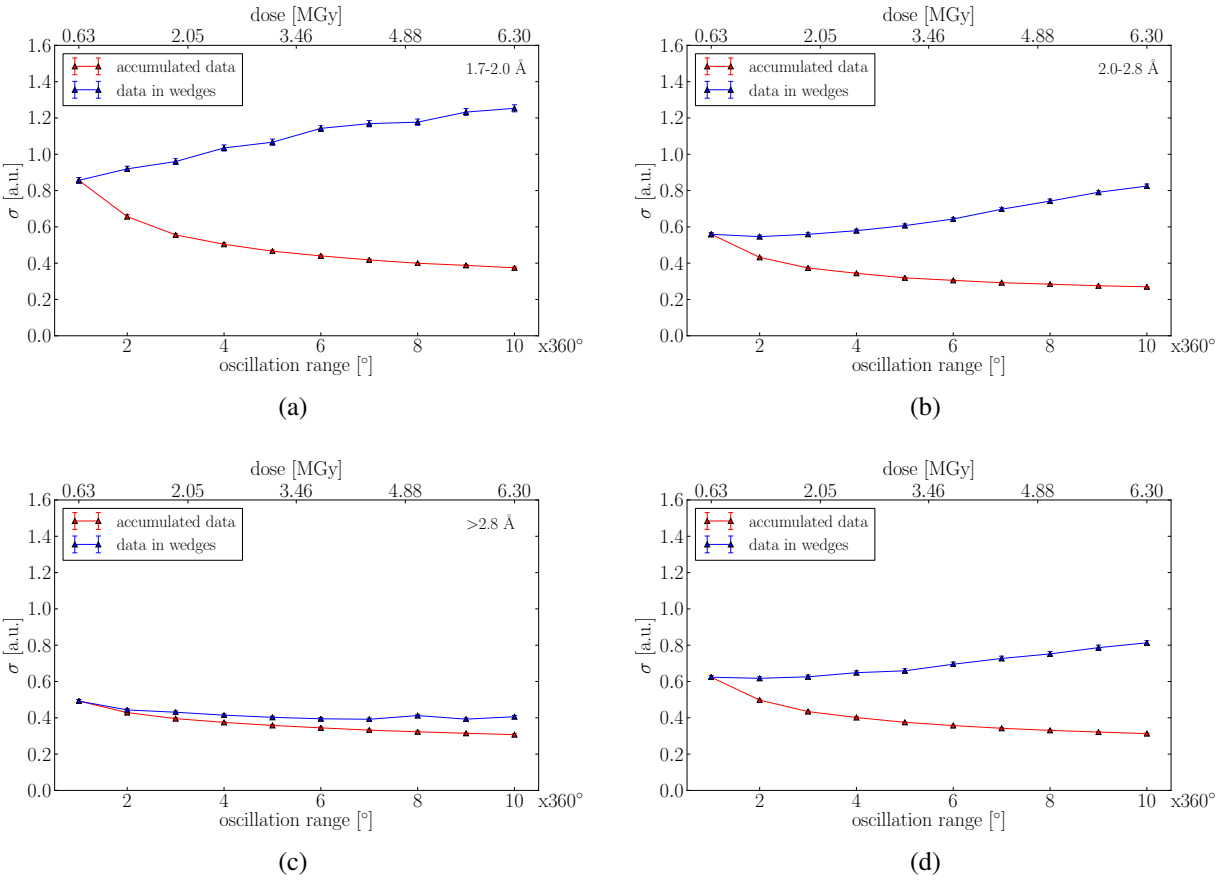



Figure B.16: Plot of the fitting parameter σ for the normal distributions fitting the normalized histograms of the average signed anomalous differences against the number of 360° turns and dose, depending on the resolution of the acentric reflections. The red curves stand for accumulated, the blue ones for wedge-wise processed data. The single plots are a) for data with a resolution of 1.7 – 2.0 Å, b) for data with a resolution of 2.0 – 2.8 Å, c) for data with a resolution of > 2.8 Å. Fig. B.16 d) is based on all average anomalous differences.

# Appendix C

# A program for the data analysis of anomalous differences

In this chapter, the code leading to the development of the metric can be found. It consists of five classes:

- the class *input_data* for sorting the reflections and their intensities into Bijvoet positives and Bijvoet negatives including the methods *unique, resolution, spacegroup, extract, make_HKL_dic* and *Bijvoet_Pairs*,
- the class *anomalous_differences* for calculating anomalous differences in various ways with the methods *Ano_Dif_1 - Ano_Dif_5*, three methods for calculating correlation coefficients of the anomalous differences regarding different aspects and a method to calculate anomalous differences from ideal data named *DF_from_ideal_data*,
- the class *analysing* for analysing the Bijvoets, the anomalous differences and for calculating the mean square difference between a fit and a histogram,
- the class *plotting* for depicting the results in various forms, e.g. in scatterplots, histograms, fits, correlation coefficients plotted versus resolution, and
- the class *helping_routines* for carrying out additional required routines, such as finding common acentric reflections, rewrite XDS_ASCII.HKL files and writing results to a pdf file.

input_data

```
import copy, string, csv
import pickle
from collections import Counter
import numpy, os
import subprocess
from scipy.stats.stats import pearsonr
from matplotlib import rc
from interval import interval
import copy
import time
import sympy
import cProfile, re
from profilehooks import timecall


# remark: extract d_min and lattice constants from file
class input_data(object):
    def __init__(self, case, path, name, r_m):
        self.case = case
        self.path = path
        self.name = name
        self.r_m= r_m
        print " DEB The data are stored in %s" % self.path
        print " DEB The crystal has the spacegroup %s." %self.case
        (self.a, self.b, self.c, self.alpha, self.beta, self.gamma, self.r_max, self.fname) = self.extract(self.r_m)
        print " DEB The cell parameter are %s %s %s %s %s %s" %(self.a, self.b, self.c, self.alpha, self.beta, self.gamma)
        print " DEB class input_data is initialized."


       # print cProfile.run('__init__()')

# determine spacegroup and the symmetry equivalents with symbolic variables according to sfall (CCP4)
    def spacegroup(self):
        h=sympy.symbols('h')
        k=sympy.symbols('k')
        l=sympy.symbols('l')
        if self.case=="P43212" or self.case=="P41212":
            print " ---\n DEB The Laue group is 4/mmm"
            self.sym_eq=[(h,k,l),(-1*k, h, l), (-1*h, -1*k, l), (k, -1*h, l), (h, -1*k, -1*l),  (k, h, -1*l), (-1*h, k, -1*l),
  (-1*k, -1*h, -1*l)]
            self._sym_eq=[(-1*h, -1*k,-1*l), (k, -1*h, -1*l), (h, k, -1*l),  (-1*k, h, -1*l), (-1*h, k, l), (-1*k, -1*h, l),  (h,
  -1*k, l), (k, h, l)]
            return (self.sym_eq, self.sym_eq)
        elif self.case=="I213":
            print " DEB The Laue group is m3bar"
            self.sym_eq=[(h,k,l), (l,h,k), (k,l,h),(-1*h,-1*k,l), (-1*l, -1*h, k), (k, -1*l, -1*h), (h, -1*k, -1*l), (l, -1*h,
  -1*k),  (-1*k, l, -1*h), (-1*h, k, -1*l), (-1*l, h, -1*k), (-1*k, -1*l,h)]
            self._sym_eq=[(-1*h, -1*k,-1*l), (-1*l,-1*h, -1*k), (-1*k, -1*l,-1*h), (h, k, -1*l), (l,h,-1*k), (-1*k, l, h), (-1*h,
  k, l), (-1*l, h, k),  (k, -1*l, h), (h, -1*k, l), (l, -h, k), (k,l,-1*h)]
        elif self.case=="P212121":
            print " DEB The Laue group is Pmmm"
            self.sym_eq=[(h,k,l), (h,-1*k,-1*l),(-1*h, k, -1*l),(-1*h, -1*k,l)]
            self._sym_eq=[(-1*h, -1*k,-1*l), (-1*h,k,l), (h, -1*k, l), (h,k,-1*l)]
        else:
            print "Symmetry equivalents and lattice type still need to be defined."


# the function extract is taking the cell constants and d_min from the header of the XDS_ASCII.HKL
    def extract(self, r_m):
        print " --- \n DEB extract started"
        fname1 = "XDS_ASCII.HKL"
        fname2 = "XDS_ASCII_sled.HKL"
        if os.path.exists(self.path+fname1):
            file=open(self.path+fname1, "r")
            lines=file.readlines(13)
            self.r_max=float(lines[10].rsplit()[-1])
            data_cell=lines[12]
            file.close()
            self.a=float(data_cell.rsplit()[1])
            self.b=float(data_cell.rsplit()[2])
            self.c=float(data_cell.rsplit()[3])
            self.alpha=float(data_cell.rsplit()[4])
            self.beta=float(data_cell.rsplit()[5])
            self.gamma=float(data_cell.rsplit()[6])
            file.close()
            return (self.a, self.b, self.c, self.alpha, self.beta, self.gamma, self.r_max, fname1)
        elif os.path.exists(self.path+fname2):
            file=open(self.path+fname2, "r")
            lines=file.readlines(100)
            self.r_max=float(r_m)
            for line in lines:
                if (re.search("!UNIT_CELL_CONSTANTS=", line)):
                    self.a=float(line.rsplit()[1])
                    self.b=float(line.rsplit()[2])
                    self.c=float(line.rsplit()[3])
                    self.alpha=float(line.rsplit()[4])
                    self.beta=float(line.rsplit()[5])
                    self.gamma=float(line.rsplit()[6])
```

```
85            file.close()
86            return (self.a, self.b, self.c, self.alpha, self.beta, self.gamma, self.r_max, fname2)
87
88 #calculating the resolution depending on the cell constants and the reflection indices, general for all space groups
89 #h,k,l are lists
90     def resolution(self, h, k, l):
91         length=len(h)
92         a=self.a
93         b=self.b
94         c=self.c
95         print " DEB cell parameters loaded"
96         alpha=self.alpha*numpy.pi/180
97         beta=self.beta*numpy.pi/180
98         gamma=self.gamma*numpy.pi/180
99         res=[]
100        i=0
101        t1=(1-numpy.cos(alpha)**2-numpy.cos(beta)**2-numpy.cos(gamma)**2+2*(numpy.cos(alpha)*numpy.cos(beta)*numpy.cos(gamma)))
102        t2=(numpy.cos(alpha)*numpy.cos(beta)-numpy.cos(gamma))
103        t3=(numpy.cos(beta)*numpy.cos(gamma)-numpy.cos(alpha))
104        t4=(numpy.cos(gamma)*numpy.cos(alpha)-numpy.cos(beta))
105 # as cos(90 degree) is not zero, the accuracy is defined to 0.00x degree, i.e. cos(90 degree) is defined to be 0
106        if t2<1e-10 or t3<1e-10 or t4<1e-10:
107            t2=0
108            t3=0
109            t4=0
110        while i < length:
111            re=(1/(a*b*c)**2*t1)*(((b*c*h[i]*numpy.sin(alpha))**2+(a*c*k[i]*numpy.sin(beta))**2+
    (a*b*l[i]*numpy.sin(gamma))**2)+2*a*b*c**2*h[i]*k[i]*t2+2*a**2*b*c*k[i]*l[i]*t3+2*a*b**2*c*h[i]*l[i]*t4)
112            d_hkl=numpy.sqrt(1/re)
113            res.append(d_hkl)
114            i=i+1
115        print "---\n DEB resolution calculated."
116        return res
117
118 #reads the reflections from the XDS_ASCII.HKL file, calculates the corresponding resolution and image number and stores
119 #the reflections with sigma >0 in dictionaries with the structure {(h[i],k[i],l[i]):[float(I[i]), float(sigma[i]), res[i],
    im_no[i]]
120     @timecall
121     def make_HKL_dic(self):
122         if self.fname=="XDS_ASCII.HKL":
123             print " --- \n DEB make_HKL start. Reading from %s" %(self.fname)
124             file=open(self.path+self.fname,'r')
125             lines=file.readlines()
126             file.close()
127             print " DEB XDS_ASCII.HKL closed."
128             data=lines[47:-1]
129             length=len(data)
130             h,k,l,I,sigma,z=numpy.loadtxt(data, usecols=(0,1,2,3,4,7), unpack=True)
131             print " DEB %s reflections loaded." %str(length)
132             res=self.resolution(h,k,l)
133         elif self.fname=="XDS_ASCII_scaled.HKL":
134             print " --- \n DEB make_HKL start. Reading from %s" %(self.fname)
135             file=open(self.path+self.fname,'r')
136             lines=file.readlines()
137             file.close()
138             print " DEB XDS_ASCII_scaled.HKL closed."
139             st_search=lines[0:100]
140             for num, line in enumerate(st_search):
141                 if (string.find(line, "     0      0")>0):
142                     print num, line
143                     break
144                     data=lines[num+1:-1]
145                     length=len(data)
146             h,k,l,I,sigma,z=numpy.loadtxt(data, usecols=(0,1,2,3,4,7), unpack=True)
147             print " DEB %s reflections loaded." %str(length)
148         res=self.resolution(h,k,l)
149 #calculate image number j from z
150         im_no=[]
151         i=0
152         while i <length:
153             j_n=round(z[i])
154             im_no.append(int(j_n))
155             i=i+1
156         print " DEB image number done."
157
158 # make a dictionary, considers the order in the XDS_ASCII.HKL file, don't take reflections with sigma <0
159 # problem: each key in a dictionary is unique, data are not merged -> multiple gives number of dictionaries which need to be
    created
160         dic={}
161         all=[]
162         n_I=0
163         for i in xrange(0,length):
164             all.append((h[i], k[i], l[i]))
165         multiple= Counter(all).most_common(1)[0][-1]
166         dlist = []
167         for m in xrange(multiple):
168             dlist.append({})
169         for i in xrange(length):
```

```
170             #print i
171             d=0
172             while d < multiple:
173                 if not dlist[d].has_key((h[i], k[i], l[i])) and sigma[i]>0:
174                     el={(h[i],k[i],l[i]):[float(I[i]), float(sigma[i]), res[i], im_no[i]]}
175                     dlist[d].update(el)
176                     if I[i]< 0:
177                         n_I=n_I+1
178                     break
179                 else:   d=d+1
180         ref=0
181         for i in xrange(len(dlist)):
182             ref+=len(dlist[i])
183         self.dlist = dlist
184         print " DEB %s reflections written to dictionaries, including systematic absences." %ref
185         print " DEB %s reflections have negative intensities." %n_I
186         print " DEB %s Dictionary(s) filled." %len(dlist)
187        # cProfile.run('re.compile("foo|bar")')
188         return (dlist, ref)
189
190 # to generate a dictionary with unique reflections, the program "unique" from ccp4 is used, as the unique reflections are dependent
    on the Laue group;
191 # systematic absences are identified (space group dependent as well)
192 # attention:sometimes there some problem in the first run; but the second run is always successful
193
194     def unique(self):
195         unique={}
196         devnull=open(os.devnull, "w")
197         f=open("%s.csh" %(self.path+self.name), "w")
198         f.write("# generate .mtz file with the ccp4 program unique containing the unique reflections\n")
199         f.write("unique HKLOUT %s.mtz >> %s_unique.log << EOF\n" % (self.path+self.name, self.path+self.name))
200         f.write("SYMM %s\n" % self.case)
201         f.write("RESOL %s\n" % str(self.r_max))
202         f.write("CELL %s %s %s\n" % (str(self.a), str(self.b), str(self.c)))
203         f.write("EOF\n")
204         f.write("# convert .mtz to ascii\n")
205         f.write("mtzdump hklin %s.mtz >> %s.log << EOF1\n" % (self.path+self.name, self.path+self.name))
206         f.write("NREF -1\n")
207         f.write("GO\n")
208         f.write("EOF1")
209         f.close()
210         os.system("chmod +x %s.csh" % (self.path+self.name))
211         subprocess.Popen(self.path+self.name+".csh", shell=True, executable="/bin/csh", stdout=devnull)
212         print "  DEB List of theoretical unique reflections written"
213         time.sleep(2)
214         file=open(self.path+self.name+".log","r")
215         lines=file.readlines()
216         file.close()
217         for num, line in enumerate(lines):
218             if 'LIST OF REFLECTIONS' in line:
219                 start=num+3
220                 data=lines[start:-6]
221         h,k,l=numpy.loadtxt(data, usecols=(0,1,2), unpack=True)
222         for i in xrange(len(h)):
223                 unique.update({(h[i], k[i], l[i]):[]})
224         print " DEB there are %s theoretical unique reflections."  %str(len(unique))
225         os.system("rm "+self.path+self.name+".log")
226         os.system("rm "+self.path+self.name+".mtz")
227         os.system("rm "+self.path+self.name+".csh")
228         self.unique=unique
229 #find the reflections which should be systematically absent
230         sys_abs=[]
231         if self.case=="P43212" or self.case=="P41212":
232             for l in xrange(0, int(max(l))):
233                 if l%4!=0:
234                     sys_abs.append((0,0,l))
235             for h in xrange(0, int(max(h))):
236                 if h%2!=0 :
237                     sys_abs.append((h,0,0))
238         elif self.case=="P212121":
239             for h in xrange(0, int(max(h))):
240                 if h%2!=0:
241                     sys_abs.append((h,0,0))
242             for k in xrange(0, int(max(k))):
243                 if k%2!=0 :
244                     sys_abs.append((0,k,0))
245             for l in xrange(0, int(max(l))):
246                 if l%2!=0 :
247                     sys_abs.append((0,0,l))
248         self.sys_abs=sys_abs
249         print " DEB There are %s theoretical systematic absences"  %str(len(sys_abs))
250         return (unique, sys_abs)
251
252 # make a dictionary with a unique, non-centric reflection as key and the following structure:
253 #  Bijvoet_Pairs{key:[[Bijvoet-Positives],[Bijvoet-Negatives]}; both Bijvoet-Positives and Negatives have
254 #the structure: [[I1, sigma1, resolutin1, image number1], [I2, sigma2, resolutin2, image number1]...]
255     @timecall
256     def Bijvoet_Pairs(self) :
```

```
257         print "--- \n DEB Bijvoet_Pairs started"
258         self.unique()
259         self.make_HKL_dic()
260         self.spacegroup()
261         Bijvoets={}
262         centric_real={}
263         i=0
264         sym_eqs={}
265         c=0
266 #    take unique keys and generate symmetry equivalents according to the space group (remember: h,k,l are symbolic variables!),
    differentiate between Bijvoet-Positive and Bijvoet-Negative
267         for el in self.unique.keys():
268             sym_eq=[]
269             _sym_eq=[]
270             centric_theo=[]
271             if el[0]!=0 and el[1]!=0 and el[2]!=0 and el[0]!=el[1] and self.case=="P43212" or self.case=="P41212":
272                     (h,k,l)=self.sym_eq[0]
273                     for key in self.sym_eq:
274                         i1=key[0]
275                         i2=key[1]
276                         i3=key[2]
277                         i1=i1.evalf(subs={h:el[0],k:el[1],l:el[2]})
278                         i2=i2.evalf(subs={h:el[0],k:el[1],l:el[2]})
279                         i3=i3.evalf(subs={h:el[0],k:el[1],l:el[2]})
280                         sym_eq.append((int(i1),int(i2),int(i3)))
281                     for key in self._sym_eq:
282                         i1=key[0]
283                         i2=key[1]
284                         i3=key[2]
285                         i1=i1.evalf(subs={h:el[0],k:el[1],l:el[2]})
286                         i2=i2.evalf(subs={h:el[0],k:el[1],l:el[2]})
287                         i3=i3.evalf(subs={h:el[0],k:el[1],l:el[2]})
288                         _sym_eq.append((int(i1),int(i2),int(i3)))
289             if el[0]!=0 and el[1]!=0 and el[2]!=0 and self.case=="P21212":
290                     (h,k,l)=self.sym_eq[0]
291                     for key in self.sym_eq:
292                         i1=key[0]
293                         i2=key[1]
294                         i3=key[2]
295                         i1=i1.evalf(subs={h:el[0],k:el[1],l:el[2]})
296                         i2=i2.evalf(subs={h:el[0],k:el[1],l:el[2]})
297                         i3=i3.evalf(subs={h:el[0],k:el[1],l:el[2]})
298                         sym_eq.append((int(i1),int(i2),int(i3)))
299                     for key in self._sym_eq:
300                         i1=key[0]
301                         i2=key[1]
302                         i3=key[2]
303                         i1=i1.evalf(subs={h:el[0],k:el[1],l:el[2]})
304                         i2=i2.evalf(subs={h:el[0],k:el[1],l:el[2]})
305                         i3=i3.evalf(subs={h:el[0],k:el[1],l:el[2]})
306                         _sym_eq.append((int(i1),int(i2),int(i3)))
307             if el[0]!=0 and el[1]!=0 and el[2]!=0 and (el[0]+el[1]+el[2])%2==0 and self.case=="I213":
308                     (h,k,l)=self.sym_eq[0]
309                     for key in self.sym_eq:
310                         i1=key[0]
311                         i2=key[1]
312                         i3=key[2]
313                         i1=i1.evalf(subs={h:el[0],k:el[1],l:el[2]})
314                         i2=i2.evalf(subs={h:el[0],k:el[1],l:el[2]})
315                         i3=i3.evalf(subs={h:el[0],k:el[1],l:el[2]})
316                         sym_eq.append((int(i1),int(i2),int(i3)))
317                     for key in self._sym_eq:
318                         i1=key[0]
319                         i2=key[1]
320                         i3=key[2]
321                         i1=i1.evalf(subs={h:el[0],k:el[1],l:el[2]})
322                         i2=i2.evalf(subs={h:el[0],k:el[1],l:el[2]})
323                         i3=i3.evalf(subs={h:el[0],k:el[1],l:el[2]})
324                         _sym_eq.append((int(i1),int(i2),int(i3)))
325
326 # takes reflections which are not systematically absent and not centric; attention!: space group dependent; can be checked with
    ecalc (CCP4)
327 # or the international tables of crystallography
328  #reflections of type h0l
329             elif (el[0]!=0 and el[1]==0 and el[2]!=0):
330                 if self.case=="P43212" or self.case=="P41212":
331                     centric_theo=centric_theo+[el, (-1*el[0], 0, -1*el[2]), (0, el[0], el[2]), (0, -1*el[0], el[2]), (-1*el[0], 0,
    el[2]), (el[0], 0, -1*el[2]), (0, -1*el[0], -1*el[2]), (0, el[0], -1*el[2])]
332                 elif self.case=="P212121" or self.case=="I213":
333                     centric_theo=centric_theo+[el, (-1*el[0], 0, -1*el[2]), (el[0], 0, -1*el[2]), (-1*el[0], 0, el[2])]
334 # reflections of type hk0
335             elif el[0]!=0 and el[1]!=0 and el[2]==0:
336                 if self.case=="P43212" or self.case=="P41212":
337                     centric_theo=centric_theo+[el, (-1*el[0], el[1], 0), (el[0], -1*el[1], 0), (-1*el[0], -1*el[1], 0), (el[1],
    el[0], 0), (-1*el[1], el[0], 0), (el[1], -1*el[0], 0), (-1*el[1], -1*el[0], 0)]
338                 elif self.case=="P212121" or self.case=="I213":
339                     centric_theo=centric_theo+[el, (-1*el[0], el[1], 0), (-1*el[0], -1*el[1], 0), (el[0], -1*el[1], 0)]
340 # reflection type hhl and h-hl
```

input_data

```
341            elif el[0]==el[1] and el[0]!=0 and el[2]!=0:
342                if self.case=="P43212" or self.case=="P41212":
343                    centric_theo=centric_theo+[el, (-1*el[0], el[0], el[2]), (el[0], -1*el[0], el[2]), (-1*el[0], -1*el[0], el[2]),
    (-1*el[0], el[0], -1*el[2]), (el[0], -1*el[0], -1*el[2]), (-1*el[0], -1*el[0], -1*el[2]), (el[0], el[0], -1*el[2])]
344# reflection type 0kl
345            elif el[0]==0 and el[1]!=0 and el[2]!=0:
346                if self.case=="P212121" or self.case=="I213" or self.case=="P41212" or self.case=="P43212":
347                    centric_theo=centric_theo+[el, (0, -1*el[1], el[2]), (0, -1*el[1], -1*el[2]), (0, el[1], -1*el[2])]
348# consider now reflections of type 00l which are not systematically absent
349            elif el[0]==0 and el[1]==0 and el[2]!=0:
350                centric_theo=centric_theo+[el, (0,0,-1*el[2])]
351# consider now reflections of type h00 (and 0k0) which are not systematically absent
352            elif el[0]!=0 and el[1]==0 and el[2]==0:
353                centric_theo=centric_theo+[el, (-1*el[0],0,0), (0,-1*el[0],0), (0,el[0],0)]
354
355            if sym_eq!=[]:
356                s=sym_eq+_sym_eq
357                sym_eqs.update({sym_eq[0]:s})
358# look for actual Bijvoet-positive/negative in dictionarys
359            if sym_eq!=[]:
360                Bi_p=[]
361                Bi_n=[]
362                for dic in self.dlist:
363                    for ref in sym_eq:
364                        if dic.has_key(ref) and not dic[ref] in Bi_p:
365                            Bi_p.append(dic[ref])
366                    for _ref in _sym_eq:
367                        if dic.has_key(_ref) and not dic[_ref] in Bi_n:
368                            Bi_n.append(dic[_ref])
369                    if Bi_p!=[] or Bi_n!=[]:
370                        Bijvoets.update({sym_eq[0]:(Bi_p, Bi_n)})
371# look for centric reflections in dictionaries
372            if centric_theo!=[]:
373                centric=[]
374        #        print centric_theo[0]
375                for dic in self.dlist:
376                    for ref in centric_theo:
377                        if dic.has_key(ref) and not dic[ref] in centric:
378                            centric.append(dic[ref])
379                if centric!=[]:
380                    centric_real.update({centric_theo[0]:centric})
381# look for systematic absent reflections
382        sys_abs={}
383        for el in self.sys_abs:
384            abs=[]
385            s_abs=[]
386            if el[0]==0 and el[1]==0 and el[2]!=0:
387                abs.append(el)
388                abs.append((0,0,-1*el[2]))
389            elif el[0]!=0 and el[1]==0 and el[2]==0:
390                abs.append(el)
391                abs.append((-1*el[0], 0, 0))
392                abs.append((0, el[0], 0))
393                abs.append((0, -1*el[0], 0))
394            for dic in self.dlist:
395                    for ref in abs:
396                        if dic.has_key(ref):
397                            s_abs.append(dic[ref])
398            if s_abs!=[]:
399                sys_abs.update({abs[0]: s_abs})
400# get number of measured systematic absence
401        sa=0
402        for key in sys_abs:
403            sa=sa+len(sys_abs[key])
404# calculate number of unique reflections
405        un=0
406        for key in Bijvoets:
407            if Bijvoets[key][0]!=[]:
408                un=un+1
409            if Bijvoets[key][1]!=[]:
410                un=un+1
411        uni=len(sys_abs)+len(centric_real)+un
412        ########################################
413        print " \n DEB %s theoretically unique systematic absent reflections were measured." %str(len(sys_abs))
414        print " \n DEB %s unique centric reflections were measured." % str(len(centric_real))
415        print " \n DEB %s Bijvoets were found and written to dictionary Bijvoets." % str(len(Bijvoets))
416        print " \n DEB %s unique reflections were measured." % str(uni)
417# store Bijvoets, centric reflections and systematic absent reflections in dictionaries
418        if not os.path.exists(self.path+self.name):
419            os.mkdir(self.path+self.name)
420        with open(self.path+self.name+'/Bijvoets.pic', 'w') as f:
421            pickle.dump(Bijvoets, f)
422        with open(self.path+self.name+'/centric.pic', 'w') as f:
423            pickle.dump(centric_real, f)
424        with open(self.path+'/sym_eq.pic', 'w') as f:
425            pickle.dump(sym_eqs, f)
426        return (Bijvoets, centric_real, sys_abs, sym_eqs)
```

anomalous_differences

```
import numpy, os, time
import copy, string
import pickle
import operator
import matplotlib
from matplotlib import rc
from collections import Counter
from input_data import *
from profilehooks import timecall
from interval import interval
from matplotlib import pyplot as plt
from matplotlib import colors as color, cm as cm
import pylab
import pprofile
from matplotlib import rc


class anomalous_differences():

    def __init__(self, case, path, name, r_m):
        self.case=case
        self.path=path
        self.name=name
        self.r_m=r_m
        # loading dictionary Bijvoets, if existing, otherwise start to produce dictionary
        print self.path+self.name+"/Bijvoets.pic"
        if os.path.exists(self.path+self.name+"/Bijvoets.pic"):
            with open(self.path+self.name+"/Bijvoets.pic", 'r') as f:
                print " DEB Bijvoets are loaded now"
                self.Bijvoets=pickle.load(f)
        else:
            print "Bijvoets need to be calculated"
            self.input=input_data(self.case, self.path, self.name, self.r_m)
            self.Bijvoets=self.input.Bijvoet_Pairs()[0]

    #build average of Bijvoet positives (<I+>) and negatives (<I->) to calculate average signed anomalous differences, considering only positive intensities
    #the image and the resolution interval of the reflections can be chosen
    #returns a dictionary with a unique reflection as key and the average signed anomalous difference, the corresponding standard deviation, the average Bijvoet intensities including standard deviations and the resolution
    def Ano_Dif1(self, im_int, resolution, _res):
        print " DEB Ano_Dif1(%s, %s, %s) started\n" %(self.name, str(im_int), str(resolution))
        ano_dif1={}
        for key in self.Bijvoets.keys():
            I_p=0
            I_n=0
            sig_p=0
            sig_n=0
            Bi_pos=self.Bijvoets[key][0]
            Bi_neg=self.Bijvoets[key][1]
            if len(Bi_pos)==0:
                I_p_av=0
                sig_p_av=0
            else:
                counter=0
                # checks whether the Bijvoet positive reflections are in the desired image and resolution interval and whether the intensity is > 0,
                # calculates average intensity and average I/sigma value for Bijvoet Positives
                for el in Bi_pos:
                    if el[3] in interval(im_int) and el[0]>0 and el[2] in interval(resolution):
                        I_p=I_p+el[0]
                        sig_p=sig_p+el[1]
                        counter=counter+1
                # builds average of the intensities fulfilling the above mentioned conditions
                if counter!=0:
                    I_p_av=I_p/counter
                    sig_p_av=sig_p/counter
                else:
                    I_p_av=0
                    sig_p_av=0
            if len(Bi_neg)==0:
                I_n_av=0
                sig_n_av=0
            else:
                counter=0
                # checks whether the Bijvoet positive reflections are in the desired image and resolution interval and whether the intensity is > 0,
                # calculates average intensity and average I/sigma value for Bijvoet Negatives
                for el in Bi_neg:
                    if el[3] in interval(im_int) and el[0]>0 and el[2] in interval(resolution):
                        I_n=I_n+el[0]
                        sig_n=sig_n+el[1]
                        counter=counter+1
                if counter!=0:
                    I_n_av=I_n/counter
                    sig_n_av=sig_n/counter
                else:
                    I_n_av=0
```

```
85                  sig_n_av=0
86              # calculate actual average anomalous difference and the average sigma(I)
87              if I_p_av >0 and I_n_av >0:
88                  ad=numpy.sqrt(I_p_av)-numpy.sqrt(I_n_av)
89                  sig=numpy.sqrt(sig_p_av)+numpy.sqrt(sig_n_av)
90                  res=self.Bijvoets[key][0][0][2]
91                  ano_dif1.update({key:[ad, sig, I_p_av, sig_p_av, I_n_av, sig_n_av, res]})
92          print " DEB Anomalous Differences calculated by averaging all %s Bijvoet positives and negatives." %str(len(ano_dif1))
93          with open(self.path+'/ano_dif1_'+_res+'.pic', 'w') as f:
94              pickle.dump(ano_dif1, f)
95          return ano_dif1
96
97      # make a dictionary with a unique reflection as key which has at least one Bijvoet positive reflection and three Bijvoet
    negative reflections.
98      # The structure of the returned dictionary is: reflection [[Bijvoet positive reflections [I, sigma, resolution, image number]],
    [Bijvoet negative reflections [I, sigma, resolution, image number]]].
99      # The Bijvoet symmetry equivalents are sorted by the image number. Remark: can be used for pseudo-symmetry, but needs to be
    extended (more Bijvoet positives)
100     def Ano_Dif2(self):
101         print " DEB Ano_Dif2() started\n"
102         Bi_select={}
103         for key in self.Bijvoets.keys():
104             if len(self.Bijvoets[key][0])>=1 and len(self.Bijvoets[key][1])>=3:
105                 pn=[]
106                 p=[]
107                 n=[]
108                 for el in self.Bijvoets[key][0]:
109                     p.append(el[1])
110                 for el in self.Bijvoets[key][1]:
111                     n.append(el[1])
112                 p=sorted(p, key=lambda tup: tup[3])
113                 n=sorted(n, key=lambda tup: tup[3])
114                 pn.append(p)
115                 pn.append(n)
116             Bi_select.update({key:pn})
117         # calculate now the anomalous difference between the first Bijvoet positive reflection in the list and all Bijvoet
    negatives
118         I_p=0
119         I_n=0
120         sig_p=0
121         sig_n=0
122         ano_dif2={}
123         for key in Bi_select:
124             ano_dif=[]
125             I_p=Bi_select[key][0][0][0]
126             sig_p=Bi_select[key][0][0][1]
127             res=Bi_select[key][0][0][2]
128             i_no=Bi_select[key][0][0][3]
129             for el in Bi_select[key][1]:
130                 if I_p>0 and el[0]>0:
131                     ad=numpy.sqrt(I_p)-numpy.sqrt(el[0])
132                     ds=numpy.sqrt(sig_p**2+el[1]**2)
133                     dif=i_no-el[3]
134                 ano_dif.append((ad, ds, dif))
135             ano_dif2.update({(key, res, i_no): ano_dif})
136         print " DEB Ano_Dif2 has been created."
137         with open(self.path+self.name+'/ano_dif2.pic', 'w') as f:
138             pickle.dump(ano_dif2, f)
139         return ano_dif2
140
141     # calculating anomalous differences pairwise, if they are in a certain image interval [I, sigma, resolution, image number]]
142     # returns dictionary with a reflection as a key and a list of anomalous differences including standard deviation, resolution
    and difference in image number
143     def Ano_Dif3(self, im_int):
144         print " DEB Ano_Dif3(%s) started\n" %str(im_int)
145         ano_dif3={}
146         for key in self.Bijvoets.keys():
147             Bi_pos=self.Bijvoets[key][0]
148             Bi_neg=self.Bijvoets[key][1]
149             m=len(Bi_pos)
150             n=len(Bi_neg)
151             ano=[]
152             for i in xrange(m):
153                 for j in xrange(n):
154                     print Bi_pos[i]
155                     dif=abs(int(Bi_pos[i][3])-int(Bi_neg[j][3]))
156                     if dif in interval(im_int):
157                         I_p=Bi_pos[i][0]
158                         sig_p=Bi_pos[i][1]
159                         I_n=Bi_neg[j][0]
160                         sig_n=Bi_neg[j][1]
161                         if I_p>0 and I_n >0:
162                             df=numpy.sqrt(I_p)-numpy.sqrt(I_n)
163                             ds=numpy.sqrt(sig_p**2+sig_n**2)
164                             ano.append([df, ds, dif])
165             if ano!=[]:
166                 ano_dif3.update({key:ano})
167         with open(self.path+self.name+'/ano_dif3.pic', 'w') as f:
```

```
168             pickle.dump(ano_dif3, f)
169         return ano_dif3
170 
171     # calculates subsequent anomalous differences, i.e. (Delta F_n - Delta F_n+1)/Delta F_n
172     def Ano_Dif4(self):
173         print " DEB Ano_Dif4 started"
174         ano_dif=[]
175         z=[]
176         x=[]
177         for key in self.Bijvoets.keys():
178             Bi_pos=self.Bijvoets[key][0]
179             Bi_neg=self.Bijvoets[key][1]
180             if len(Bi_pos)> 0 and len(Bi_neg)>0:
181                 I_pos=[]
182                 z_pos=[]
183                 I_neg=[]
184                 z_neg=[]
185                 # sort Bijvoet positive and negative by image number
186                 Bi_pos_s=sorted(Bi_pos, key=lambda tup: tup[3])
187                 Bi_neg_s=sorted(Bi_neg, key=lambda tup: tup[3])
188                 # select Bijvoets with intensities >0 and keep the corresponding image numbers
189                 for i in xrange(len(Bi_pos_s)):
190                     I_p=Bi_pos_s[i][0]
191                     if I_p > 0 :
192                         I_pos.append(I_p)
193                         z_pos.append(Bi_pos_s[i][3])
194                 for i in xrange(len(Bi_neg_s)):
195                     I_n=Bi_neg_s[i][0]
196                     if I_n >0:
197                         I_neg.append(I_n)
198                         z_neg.append(Bi_neg_s[i][3])
199                 # calculate the subsequent Bijvoets
200                 if len(I_pos)>1 and I_neg!=[]:
201                     if len(I_pos)>len(I_neg):
202                         for i in xrange(len(I_neg)):
203                             ano_n=numpy.sqrt(I_pos[i+1])-numpy.sqrt(I_neg[i])
204                             ano=numpy.sqrt(I_pos[i])-numpy.sqrt(I_neg[i])
205                             dif=(ano_n-ano)/ano
206                             if not numpy.isinf(dif):
207                                 ano_dif.append(dif)
208                                 # keep image number of I+ and the difference of the image number of the inbound reflections
209                                 x.append(z_pos[i+1])
210                                 z.append(abs(z_pos[i+1]-z_neg[i]))
211                     elif len(I_pos)==len(I_neg):
212                         for i in xrange(len(I_neg)-1):
213                             ano_n=numpy.sqrt(I_pos[i+1])-numpy.sqrt(I_neg[i])
214                             ano=numpy.sqrt(I_pos[i])-numpy.sqrt(I_neg[i])
215                             dif=(ano_n-ano)/ano
216                             if not numpy.isinf(dif):
217                                 ano_dif.append(dif)
218                                 z.append(abs(z_pos[i+1]-z_neg[i]))
219                                 x.append(z_pos[i+1])
220                     elif len(I_pos)<len(I_neg):
221                         for i in xrange(len(I_pos)-1):
222                             ano_n=numpy.sqrt(I_pos[i+1])-numpy.sqrt(I_neg[i])
223                             ano=numpy.sqrt(I_pos[i])-numpy.sqrt(I_neg[i])
224                             dif=(ano_n-ano)/ano
225                             if not numpy.isinf(dif):
226                                 ano_dif.append(dif)
227                                 x.append(z_pos[i]+1)
228                                 z.append(abs(z_pos[i+1]-z_neg[i]))
229         with open(self.path+self.name+'/ano_dif4.pic', 'w') as f:
230             pickle.dump((x,z,ano_dif), f)
231         print " DEB (dF_n+1 - dF_n )/dF_n: "
232         ano_dif=numpy.asarray(sorted(ano_dif))
233         z=numpy.asarray(sorted(z))
234         x=numpy.asarray(sorted(x))
235         return (x, z, ano_dif)
236 
237     # calculate Delta F_n - Delta F_n+1 /< Delta F > for a certain image interval with values in the interval change
238     # Ano_Dif1 calculates <Delta F>, seq_int should not be larger than im_int
239     # returns ano_dif =  Delta F_n - Delta F_n+1 /< Delta F > as a list and ano_dif5 returns a list with the reflections,
    intensities and corresponding image numbers involved in calculating ano_dif
240     @timecall
241     def Ano_Dif5(self, seq_int, im_int, resolution, change):
242         #loading average anomalous differences which are serving as scaling factor
243         print "--- \n DEB Ano_Dif5 started"
244         if os.path.exists(self.path+self.name+"/ano_dif1.pic"):
245             with open(self.path+self.name+"/ano_dif1.pic", 'r') as f:
246                 ano_dif1=pickle.load(f)
247         print "DEB anoDif1 loaded"
248         else:
249                print " DEB ano_dif1 is created now"
250         ano_dif1=self.Ano_Dif1(im_int, resolution)
251         ano_dif=[]
252         ano_all=[]
253         ano_n_all=[]
254         z=[]
```

```
255         ano_dif5b={}
256         lost={}
257         for key in self.Bijvoets.keys():
258            if key in ano_dif1.keys():
259            Bi_pos=self.Bijvoets[key][0]
260                Bi_neg=self.Bijvoets[key][1]
261                if len(Bi_pos)> 0 and len(Bi_neg)>0:
262                  res=Bi_pos[0][2]
263                  if res in interval(resolution):
264                     I_pos=[]
265                     z_pos=[]
266                     I_neg=[]
267                     z_neg=[]
268                     # sort Bojvoet positives and negatives by image number
269                     Bi_pos_s=sorted(Bi_pos, key=lambda tup: tup[3])
270                     Bi_neg_s=sorted(Bi_neg, key=lambda tup: tup[3])
271                     for i in xrange(len(Bi_pos_s)):
272                         I_p=Bi_pos_s[i][0]
273                         z_p=Bi_pos_s[i][3]
274                         if I_p > 0 and z_p in interval(im_int):
275                             I_pos.append(I_p)
276                             z_pos.append(z_p)
277                     for i in xrange(len(Bi_neg_s)):
278                         I_n=Bi_neg_s[i][0]
279                         z_n=Bi_neg_s[i][3]
280                         if I_n >0 and z_n in interval(im_int):
281                             I_neg.append(I_n)
282                             z_neg.append(z_n)
283                     # in case that subsequent anomalous differences cannot be calculated, store the key in the dictionary "lost"
284                     if I_pos==[] or I_neg==[]:
285                         lost.update({key:[I_pos, I_neg]})
286                     elif len(I_pos)==1 and len(I_neg)==1:
287                         lost.update({key:[I_pos, I_neg]})
288                     # start calculation of the subsequent anomalous differences
289                     elif (len(I_pos)>=1 and len(I_neg)>1) or (len(I_pos)>1 and len(I_neg)>=1):
290                         if len(I_pos)>len(I_neg):
291                             for i in xrange(len(I_neg)-1):
292                                 if len(I_neg) >1 and z_neg[i+1] < z_pos[i+1]:
293                                     ano_n=abs(numpy.sqrt(I_pos[i])-numpy.sqrt(I_neg[i+1]))
294                                     ano=abs(numpy.sqrt(I_pos[i])-numpy.sqrt(I_neg[i]))
295                                     dif=(ano-ano_n)/ano_dif1[key][0]
296                                     I_z_a=[(I_pos[i], z_pos[i]), (I_neg[i], z_neg[i]), (I_neg[i+1], z_neg[i+1])]
297                                     z_min=map(min, zip(*I_z_a))[1]
298                                     z_max=map(max, zip(*I_z_a))[1]
299                                 else:
300                                     ano_n=abs(numpy.sqrt(I_pos[i+1])-numpy.sqrt(I_neg[i]))
301                                     ano=abs(numpy.sqrt(I_pos[i])-numpy.sqrt(I_neg[i]))
302                                     dif=(ano-ano_n)/ano_dif1[key][0]
303                                     I_z_b=[(I_pos[i], z_pos[i]), (I_neg[i], z_neg[i]), (I_pos[i+1], z_pos[i+1])]
304                                     z_min=map(min, zip(*I_z_b))[1]
305                                     z_max=map(max, zip(*I_z_b))[1]
306                                 if abs(z_max-z_min) in interval(seq_int) and dif in interval(change):
307                                     ano_dif.append(dif)
308                                     ano_all.append(ano)
309                                     ano_n_all.append(ano_n)
310                                     z.append(z_max)
311                         elif len(I_pos)==len(I_neg) and len(I_neg)!=1:
312                             for i in xrange(len(I_neg)-1):
313                                 if z_neg[i+1] < z_pos[i+1]:
314                                     ano_n=abs(numpy.sqrt(I_pos[i])-numpy.sqrt(I_neg[i+1]))
315                                     ano=abs(numpy.sqrt(I_pos[i])-numpy.sqrt(I_neg[i]))
316                                     dif=(ano-ano_n)/ano_dif1[key][0]
317                                     I_z_a=[(I_pos[i], z_pos[i]), (I_neg[i], z_neg[i]), (I_neg[i+1], z_neg[i+1])]
318                                     z_min=map(min, zip(*I_z_a))[1]
319                                     z_max=map(max, zip(*I_z_a))[1]
320                                 else:
321                                     ano_n=abs(numpy.sqrt(I_pos[i+1])-numpy.sqrt(I_neg[i]))
322                                     ano=abs(numpy.sqrt(I_pos[i])-numpy.sqrt(I_neg[i]))
323                                     dif=(ano-ano_n)/ano_dif1[key][0]
324                                     I_z_b=[(I_pos[i], z_pos[i]), (I_neg[i], z_neg[i]), (I_pos[i+1], z_pos[i+1])]
325                                     z_min=map(min, zip(*I_z_b))[1]
326                                     z_max=map(max, zip(*I_z_b))[1]
327                                 if abs(z_max-z_min) in interval(seq_int) and dif in interval(change):
328                                     ano_dif.append(dif)
329                                     ano_all.append(ano)
330                                     ano_n_all.append(ano_n)
331                                     z.append(z_max)
332                         elif len(I_pos)<len(I_neg):
333                             for i in xrange(len(I_pos)-1):
334                                 if len(I_pos)>1 and z_pos[i+1] < z_neg[i+1]:
335                                     ano_n=abs(numpy.sqrt(I_pos[i+1])-numpy.sqrt(I_neg[i]))
336                                     ano=abs(numpy.sqrt(I_pos[i])-numpy.sqrt(I_neg[i]))
337                                     dif=((ano-ano_n))/ano_dif1[key][0]
338                                     I_z_a=[(I_pos[i], z_pos[i]), (I_neg[i], z_neg[i]), (I_pos[i+1], z_pos[i+1])]
339                                     z_min=map(min, zip(*I_z_a))[1]
340                                     z_max=map(max, zip(*I_z_a))[1]
341                                 else:
342                                     ano_n=abs(numpy.sqrt(I_pos[i])-numpy.sqrt(I_neg[i+1]))
```

```
343                                     ano=abs(numpy.sqrt(I_pos[i])-numpy.sqrt(I_neg[i]))
344                                     dif=((ano-ano_n))/ano_dif1[key][0]
345                                     I_z_b=[(I_pos[i], z_pos[i]), (I_neg[i], z_neg[i]), (I_neg[i+1], z_neg[i+1])]
346                                     z_min=map(min, zip(*I_z_b))[1]
347                                     z_max=map(max, zip(*I_z_b))[1]
348                                 if abs(z_max-z_min) in interval(seq_int) and dif in interval(change):
349                                     ano_dif.append(dif)
350                                     ano_all.append(ano)
351                                     ano_n_all.append(ano_n)
352                                     z.append(z_max)
353         # store subsequent Bijvoets, the anomalous differences Delta F and Delta F_n with the image numbers z_max and "lost" keys
    in dictionaries
354         with open(self.path+'/ano_dif5_'+str(change[1])+'.pic', 'w') as f:
355             pickle.dump((ano_dif, ano_all, ano_n_all, z), f)
356         with open(self.path+'/lost.pic', 'w') as f:
357             pickle.dump(lost, f)
358         print "There are %s keys not treated in ano_dif5" %len(lost)
359         print " DEB %s data points have been found" %str(len(ano_dif))
360         return (ano_dif, ano_all, ano_n_all, z)
361
362     # plot Delta Fn vs Delta F_n+1 and print the correlation coefficient
363     def CC_ano_subsequent(self, path, im_int, resolution):
364         print "--- \n DEB Ano_Dif6 started"
365         self.input=input_data(self.case, path)
366         self.Bijvoets=self.input.Bijvoet_Pairs()[0]
367         ano_dif=[]
368         ano_dif_n=[]
369         for key in self.Bijvoets.keys():
370             Bi_pos=self.Bijvoets[key][0]
371             Bi_neg=self.Bijvoets[key][1]
372             if len(Bi_pos)> 0 and len(Bi_neg)>0:
373                 I_pos=[]
374                 z_pos=[]
375                 I_neg=[]
376                 z_neg=[]
377                 Bi_pos_s=sorted(Bi_pos, key=lambda tup: tup[3])
378                 Bi_neg_s=sorted(Bi_neg, key=lambda tup: tup[3])
379                 print len(Bi_pos), len(Bi_neg)
380                 for i in xrange(len(Bi_pos_s)):
381                     I_p=Bi_pos_s[i][0]
382                     if I_p > 0 :
383                         I_pos.append(I_p)
384                         z_pos.append(Bi_pos_s[i][3])
385                 for i in xrange(len(Bi_neg_s)):
386                     I_n=Bi_neg_s[i][0]
387                     if I_n >0:
388                         I_neg.append(I_n)
389                         z_neg.append(Bi_neg_s[i][3])
390                         res=Bi_neg_s[i][2]
391                 if len(I_pos)>1 and I_neg!=[]:
392                     if len(I_pos)>len(I_neg):
393                         for i in xrange(len(I_neg)):
394                             if z_pos[i+1] in interval(im_int) and res in interval(im_int):
395                                 ano_dif_n.append(numpy.sqrt(I_pos[i+1])-numpy.sqrt(I_neg[i]))
396                                 ano_dif.append(numpy.sqrt(I_pos[i])-numpy.sqrt(I_neg[i]))
397                     elif len(I_pos)==len(I_neg):
398                         for i in xrange(len(I_neg)-1):
399                             if z_pos[i+1] in interval(im_int) and res in interval(im_int):
400                                 ano_dif_n.append(numpy.sqrt(I_pos[i+1])-numpy.sqrt(I_neg[i]))
401                                 ano_dif.append(numpy.sqrt(I_pos[i])-numpy.sqrt(I_neg[i]))
402                     elif len(I_pos)<len(I_neg):
403                         for i in xrange(len(I_pos)-1):
404                             if z_pos[i+1] in interval(im_int) and res in interval(im_int):
405                                     ano_dif_n.append(numpy.sqrt(I_pos[i+1])-numpy.sqrt(I_neg[i]))
406                                     ano_dif.append(numpy.sqrt(I_pos[i])-numpy.sqrt(I_neg[i]))
407             print ano_dif_n
408             print " DEB dF_n "
409             print ano_dif
410             print len(ano_dif)
411             pylab.clf()
412             pylab.plot(ano_dif, ano_dif_n,'r^')
413             pylab.xlabel(r'$\Delta F_n$', fontsize=14)
414             pylab.ylabel(r"$\Delta F_{n+1}$", fontsize=14)
415             CC=100*numpy.corrcoef(ano_dif, ano_dif_n)[0][1]
416             print CC
417             CC=round(CC, ndigits=2)
418             pylab.annotate('CC=%s %%' %CC, xy=(0.05, 0.95), xycoords='axes fraction', horizontalalignment='left',
    verticalalignment='top', fontsize=14)
419             pylab.title(r'$\Delta F_{n+1}$ vs $\Delta F_n$ in an image interval of '+str(interval)+ ' for reflections within a
    resolution range of ' + str(resolution) + ' $\AA$', fontsize=14)
420             pylab.savefig("/Users/storm/Desktop/test_cc.png")
421
422
423     # calculate the CC of reflections within a sliding window within a certain image interval within a given resolution
424     # code needs to be improved!
425     def CC_ano_subsequent_sliding_window(self, path, im_int, window, resolution):
426             print "--- \n DEB Ano_Dif7 started"
427             ano_dif=[]
```

```
428             ano_dif_n=[]
429             CCs=[]
430             z=[]
431             n=im_int[1]/window
432             print n
433             arr=numpy.zeros(shape=(2, 2*n))
434             for key in self.Bijvoets.keys():
435                 for j in range(n):
436                     Bi_pos=[]
437                     Bi_neg=[]
438                     for i in self.Bijvoets[key][0]:
439                         if i[3] in interval([j*window, (j+1)*window]):
440                             Bi_pos.append(i)
441                     for i in self.Bijvoets[key][1]:
442                         if i[3] in interval([j*window, (j+1)*window]):
443                             Bi_neg.append(i)
444                     if len(Bi_pos)>= 2 and len(Bi_neg)>1:
445                         I_pos=[]
446                         z_pos=[]
447                         I_neg=[]
448                         z_neg=[]
449                         for i in xrange(len(Bi_pos)):
450                             I_p=Bi_pos[i][0]
451                             if I_p > 0 and Bi_pos[i][3] in interval([j*window, (j+1)*window]):
452                                 I_pos.append(I_p)
453                         for i in xrange(len(Bi_neg)):
454                             I_n=Bi_neg[i][0]
455                             if I_n >0 and Bi_neg[i][3] in interval([j*window, (j+1)*window]):
456                                 I_neg.append(I_n)
457                                 res=Bi_neg[i][2]
458                         if len(I_pos)>1 and I_neg!=[]:
459                             if len(I_pos)>len(I_neg):
460                                 for i in xrange(len(I_neg)):
461                                     if res in interval(resolution):
462                                         list=[0]*2*n
463                                         list[2*j]=numpy.sqrt(I_pos[i+1])-numpy.sqrt(I_neg[i])
464                                         list[(2*j)+1]= numpy.sqrt(I_pos[i])-numpy.sqrt(I_neg[i])
465                                         arr=numpy.append(arr, [list], axis=0)
466                             elif len(I_pos)==len(I_neg):
467                                 for i in xrange(len(I_neg)-1):
468                                     if res in interval(resolution):
469                                         list=[0]*2*n
470                                         list[2*j]=numpy.sqrt(I_pos[i+1])-numpy.sqrt(I_neg[i])
471                                         list[(2*j)+1]= numpy.sqrt(I_pos[i])-numpy.sqrt(I_neg[i])
472                                         arr=numpy.append(arr, [list], axis=0)
473                             elif len(I_pos)<len(I_neg):
474                                 for i in xrange(len(I_pos)-1):
475                                     if res in interval(resolution):
476                                         list=[0]*2*n
477                                         list[2*j]=numpy.sqrt(I_pos[i+1])-numpy.sqrt(I_neg[i])
478                                         list[(2*j)+1]= numpy.sqrt(I_pos[i])-numpy.sqrt(I_neg[i])
479                                         arr=numpy.append(arr, [list], axis=0)
480             for j in range(n):
481                 z.append((j+1)*window)
482             arr=numpy.transpose(arr)
483             print arr.shape
484             q=0
485             while q < 2*n:
486                 CC=numpy.corrcoef(arr[q], arr[q+1])[0][1]#
487                 CC=100*CC
488                 print CC
489                 CCs.append(round(CC, ndigits=2))
490                 q=q+2
491             pylab.clf()
492             pylab.plot(z, CCs, 'r^-')
493             pylab.xlabel(r'image number', fontsize=12)
494             pylab.ylabel(r"CC($\Delta F_n, \Delta F_{n+1}$)/%", fontsize=12)
495             pylab.ylim(min(CCs)-2, max(CCs)+2)
496             pylab.title(r'CC in image intervals of '+str(window)+ ' for reflections within a resolution range of ' +
    str(resolution) + '$\AA$', fontsize=12, y=1.1)
497             pylab.savefig("/Users/storm/Documents/Programming/workspace/Anomalous_Differences/20141203/logfile_006.png",
    bbox_inches='tight', dpi=300)
498
499         # calculating the CC of subsequent Delta Fn, Delta F_n+1 in different resolution shells
500         def CC_subsequent_ano_res(self, path, im_int, res_shells):
501             print "--- \n DEB Ano_Dif8 started"
502             ano_dif=[]
503             ano_dif_n=[]
504             CCs=[]
505             i=0
506             z=[]
507             n=len(res_shells)
508             arr=numpy.zeros(shape=(2, 2*n))
509             for key in self.Bijvoets.keys():
510                 Bi_pos=self.Bijvoets[key][0]
511                 Bi_neg=self.Bijvoets[key][1]
512                 if len(Bi_pos)>= 2 and len(Bi_neg)>1:
513                         I_pos=[]
```

```
514                         z_pos=[]
515                         I_neg=[]
516                         z_neg=[]
517                         for i in xrange(len(Bi_pos)):
518                             I_p=Bi_pos[i][0]
519                             if I_p > 0 :
520                                 I_pos.append(I_p)
521                         for i in xrange(len(Bi_neg)):
522                             I_n=Bi_neg[i][0]
523                             if I_n >0 :
524                                 I_neg.append(I_n)
525                             res=Bi_neg[i][2]
526                         for j in range(n-1):
527                             if j in interval([res_shells[j], res_shells[j+1]]):
528                                 return j
529                         if len(I_pos)>1 and I_neg!=[]:
530                             if len(I_pos)>len(I_neg):
531                                 for i in xrange(len(I_neg)):
532                                     list=[0]*2*n
533                                     list[2*j]=numpy.sqrt(I_pos[i+1])-numpy.sqrt(I_neg[i])
534                                     list[(2*j)+1]= numpy.sqrt(I_pos[i])-numpy.sqrt(I_neg[i])
535                                     arr=numpy.append(arr, [list], axis=0)
536                             elif len(I_pos)==len(I_neg):
537                                 for i in xrange(len(I_neg)-1):
538                                     list=[0]*2*n
539                                     list[2*j]=numpy.sqrt(I_pos[i+1])-numpy.sqrt(I_neg[i])
540                                     list[(2*j)+1]= numpy.sqrt(I_pos[i])-numpy.sqrt(I_neg[i])
541                                     arr=numpy.append(arr, [list], axis=0)
542                             elif len(I_pos)<len(I_neg):
543                                 for i in xrange(len(I_pos)-1):
544                                     list=[0]*2*n
545                                     list[2*j]=numpy.sqrt(I_pos[i+1])-numpy.sqrt(I_neg[i])
546                                     list[(2*j)+1]= numpy.sqrt(I_pos[i])-numpy.sqrt(I_neg[i])
547                                     arr=numpy.append(arr, [list], axis=0)
548
549             for j in range(n-1):
550                 z.append(res_shells[j]+res_shells[j+1]/2)
551             print "here we are"
552             arr=numpy.transpose(arr)
553             print arr.shape
554             q=0
555             while q < 2*n:
556                 CC=numpy.corrcoef(arr[q], arr[q+1])[0][1]#
557                 CC=100*CC
558                 print CC
559                 CCs.append(round(CC, ndigits=2))
560                 q=q+2
561             pylab.clf()
562             pylab.plot(z, CCs, 'r^-')
563             pylab.xlabel(r'image number', fontsize=12)
564             pylab.ylabel(r"CC($\Delta F_n, \Delta F_{n+1}$)/%", fontsize=12)
565             pylab.ylim(min(CCs)-2, max(CCs)+2)
566             pylab.title(r'CC of $\Delta F_n$ and $\Delta F_{n+1} in different resolution shells$ ', fontsize=12, y=1.05)
567             pylab.savefig("/Users/storm/Documents/Programming/workspace/Anomalous_Differences/20141203/logfile_007.png",
    bbox_inches='tight', dpi=300)
568             with open(self.path+'arr.pic', 'w') as f:
569                 pickle.dump(arr, f)
570
571     # write anomalous difference from ideal data generated with phenix
572     def DF_from_ideal_data(self, reference, out):
573         # read hkl output
574         file=open(reference, "r")
575         lines=file.readlines()
576         data=lines[88:-6]
577         _data=[]
578         for line in data:
579             if "?" in line:
580                 continue
581             else:
582                 _data.append(line)
583         h,k,l,F_p,F_n=numpy.loadtxt(_data, usecols=(0,1,2,3,4), unpack=True)
584       # h,k,l,I,sig=numpy.loadtxt(_data, usecols=(0,1,2,3,4), unpack=True)
585         file.close()
586         # write reflections and anomalous difference to dictionary
587         dic={}
588         print len(_data)
589         for i in xrange(0,(len(_data)-1)):
590                 dic.update({(int(h[i]), int(k[i]), int(l[i])): [F_p[i]-F_n[i], F_p[i]**2, F_p[i], F_n[i]**2, F_n[i]]})
591             #   dic.update({(int(k[i]), int(h[i]), int(l[i])): [I[i], sig[i]]})
592         # write dic to file
593         if os.path.exists(out):
594             with open(out+"ano_dif1_phenix.pic", "w") as q:
595                 pickle.dump(dic, q)
596         print "file written to "+out+"ano_dif1_phenix.pic"
597
598
599
```

analysing

```
import matplotlib.mlab as mlab
from collections import Counter
from profilehooks import timecall
from interval import interval
from scipy import math
import scipy.misc
from scipy.stats import norm
from matplotlib.gridspec import GridSpec
from matplotlib.backends.backend_pdf import PdfPages
import numpy, scipy
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec
from matplotlib import rc
# import methods from other classes
from input_data import *
from verifying import *
from anomalous_differences import *
from plotting import *
from helping_routines import *



class analysing():
    def __init__(self, case, path, name, r_m):
        self.case = case
        self.path = path
        self.name = name
        self.r_m = r_m

# analyse the average anomalous differences resulting from a Bijvoet pair, including minimal, maximal and mean anomalous signal and the corresponding histogram
    @timecall
    def analyse_Ano_Dif1(self, im_int, stepsize, resolution, mode, _res, limit, _type):
        ano_dif1s=[]
        normed_ano_dif1s=[]
        lengths=[]
        im_s=im_int[0]
        im_r=stepsize
        x=[]
        y=[]
        err=[]
        parameters=[]
        # reading average anomalous differences from file
        while im_r<=im_int[1]:
            new_name="xds_tha_1_02_"+str(im_r/360)+"_1/tha1/"
            print self.path+new_name+'ano_dif1_'+_res+'.pic'
            if os.path.exists(self.path+new_name+"ano_dif1_"+_res+".pic"):
                with open(self.path+new_name+'ano_dif1_'+_res+'.pic', 'r') as f:
                    ano_dif1 = pickle.load(f)
                    print len(ano_dif1)
            else:
                ano_dif1= self.ano.Ano_Dif1(im_int, resolution, r_m,  _res)
            # convert anomalous differences to array and perform statistics
            a = numpy.ndarray(shape=(len(ano_dif1), 1), dtype=float)
            j = 0
            if mode=='I':
                for key in ano_dif1:
                    ano = ano_dif1[key][0]
                    a[j] = ano
                    j = j + 1
            elif mode=='I_sig':
                for key in ano_dif1:
                    ano = ano_dif1[key][1]
                    a[j] = ano
                    j = j + 1
            a_min = numpy.min(a)
            a_max = numpy.max(a)
            a_mean = numpy.mean(a)
            a_st = numpy.std(a)
            print " DEB The minimal average anomalous difference is %s, the maximal anomalaous difference is %s." % (a_min, a_max)
            print " DEB The mean average anomalous difference is %s, the corresponding standard deviation is %s." % (a_mean, a_st)
            # calculate histrogram of average anomalous differences and fit it with a gaussian of type A*exp(-0.5*(x/sigma)^2)
            if _type=="hist":
                bins = numpy.linspace(-1*limit, limit, num=101)
                _y, _x = numpy.histogram(a, bins=bins)
                _x = bins[:-1] + (bins[1] - bins[0]) / 2
                # trying to fit with a gaussioan distribution of type A*exp(-0.5*(x-x0)/sigma)^2)
                fitfunc= lambda p, x: p[0]*numpy.exp(-0.5*((x-p[1])/p[2])**2)
                errfunc= lambda p, x, y: fitfunc(p, x) - y
                # initial fit parameters
                p=[max(_y), a_mean, a_st]
                p1, pcov, infodict, errmsg, success = scipy.optimize.leastsq(errfunc, p, args=(_x,_y), full_output=1)
                parameters.append(p1)
                ano_dif1s.append(a)
                im_r=im_r+stepsize
            # norm the histogram to 1 and fit it with a normal distribution
            elif _type=="norm_fit":
                bins = numpy.linspace(-1*limit, limit, num=101)
```

```
88                _y, _x = numpy.histogram(a, bins=bins)
89                _x = bins[:-1] + (bins[1] - bins[0]) / 2
90                # norming the histogram to 1, alternative: pylab.hist(a, bins, normed=1, facecolor='green'); fitting with a normal
  distribution
91                _y = 1.0 * _y /  _y.sum() / (bins[1] - bins[0])
92                fitfunc = lambda p, x:  1/(numpy.sqrt(2*numpy.pi)*p[0]) * numpy.exp(-0.5 * ((x - p[1]) / p[0]) ** 2)
93                errfunc = lambda p, x, y: fitfunc(p, x) - y
94                p=[a_st, a_mean]
95                p1, pcov, infodict, errmsg, success = scipy.optimize.leastsq(errfunc, p, args=(_x,_y), full_output=1)
96                # calculate the fitting error of sigma according to http://stackoverflow.com/questions/14581358/getting-standard-
  errors-on-fitted-parameters-using-the-optimize-leastsq-method-i
97                # by multiplying the square root of the corresponding covariance matrix entry (_all[1]) with the residual s__sq =
  sum[(f(x)-y)^2]/(N-n), where N is number of data points and n is the number of fitting parameters
98                if (len(_y) > len(p)) and pcov is not None:
99                    s_sq = (errfunc(p1, _x, _y)**2).sum()/(len(_y)-len(p))
100                   pcov = pcov * s_sq
101               error=[]
102               for i in range(len(p1)):
103                   try:
104                       error.append( numpy.absolute(pcov[i][i])**0.5)
105                   except:
106                       error.append( 0.00 )
107               parameters.append(p1)
108               err.append(error)
109               normed_ano_dif1s.append(_y)
110               im_r=im_r+stepsize
111               lengths.append(len(a))
112       # store the average signed anomalous differences, fits and errors
113       if "hist" in _type:
114           print parameters
115           with open(self.path+self.name+"_"+_res+'_ano_dif1_hist.pic', "w") as g:
116               pickle.dump((len(a), ano_dif1s, parameters), g)
117           return (ano_dif1s, parameters)
118       elif "norm_fit" in _type:
119           print parameters
120           with open(self.path+self.name+"_"+_res+'_ano_dif1_norm_fit.pic', "w") as n:
121               pickle.dump((lengths, normed_ano_dif1s, parameters, err), n)
122           return(normed_ano_dif1s, parameters, err)
123
124   # calculate mean square deviation of fit and histogram
125   def compare_fit_with_histogram(self, _res, limit):
126       print self.path+self.name+"_"+_res+'_ano_dif1_norm_fit.pic'
127       if os.path.exists(self.path+self.name+"_"+_res+'_ano_dif1_norm_fit.pic'):
128           with open(self.path+self.name+"_"+_res+'_ano_dif1_norm_fit.pic', "r") as n:
129               lengths, normed_ano_dif1s, parameters, err=pickle.load(n)
130       def norm_dist(x, p, p_1):
131            return 1/(numpy.sqrt(2*numpy.pi)*p) * numpy.exp(-0.5 * ((x - p_1) / p) ** 2)
132       for i in xrange(len(normed_ano_dif1s)):
133           bins = numpy.linspace(-1*limit, limit, num=101)
134           x = bins[:-1] + (bins[1] - bins[0]) / 2
135           y=norm_dist(x, parameters[i][0], parameters[i][1])
136           sum=0
137           for k in xrange(0, len(y)):
138               sum=sum+(normed_ano_dif1s[i][k]-y[k])**2
139           sum=sum/len(y)
140
141
142
143# statistics of Bijvoet Positives and Negatives, considering intensities >0 only
144   @timecall
145   def _Bijvoets(self, name, im_int, resolution):
146       print " --- \n DEB analysing_old Bijvoets statistics"
147       if os.path.exists(self.path +"/Bijvoets.pic"):
148           with open(self.path + '/Bijvoets.pic', 'r') as f:
149               Bijvoets = pickle.load(f)
150       else:
151           self.input = input_data(self.case, self.path, name)
152           Bijvoets = self.input.Bijvoet_Pairs()[0]
153
154       a = numpy.ndarray(shape=(len(Bijvoets), 2), dtype=int)
155       b = numpy.ndarray(shape=(len(Bijvoets), 2), dtype=int)
156       ii = 0
157       c=0
158       for key in Bijvoets.keys():
159           I_p = []
160           I_n = []
161           Bi_pos = Bijvoets[key][0]
162           Bi_neg = Bijvoets[key][1]
163           if len(Bi_pos) != 0:
164               if Bi_pos[0][2] in interval(resolution):
165                   for j in xrange(len(Bi_pos)):
166                       I = Bi_pos[j][0]
167                       if I > 0 and Bi_pos[j][3] in interval(im_int):
168                           I_p.append(I)
169                   if I_p != []:
170                       I_p_m = max(I_p)
171                   else:
172                       I_p_m = 0
```

```
173             if len(Bi_neg) != 0:
174                 if Bi_neg[0][2] in interval(resolution):
175                     for n in xrange(len(Bi_neg)):
176                         _I = Bi_neg[n][0]
177                         if _I > 0 and Bi_neg[n][3] in interval(im_int):
178                             I_n.append(_I)
179                     if I_n != []:
180                         I_n_m = max(I_n)
181                     else: I_n_m = 0
182             if len(I_p)!=0 and len(I_n)!=0:
183                 a[ii] = [len(I_p), len(I_n)]
184                 b[ii] = [I_p_m, I_n_m]
185                 ii = ii + 1
186                 c=c+1
187         print c
188         Bi_pos_m = numpy.mean(a[:, 0])
189         Bi_pos_std = numpy.std(a[:, 0])
190         Bi_neg_m = numpy.mean(a[:, 1])
191         Bi_neg_std = numpy.std(a[:, 1])
192         print " DEB The mean multiplicity for the Bijvoet positives is %s, the corresponding standard deviation is %s." %
    (str(Bi_pos_m), str(Bi_pos_std))
193         print " DEB The mean multiplicity for the Bijvoet negatives is %s, the corresponding standard deviation is %s." %
    (str(Bi_neg_m), str(Bi_neg_std))
194         Bi_pos_I_max = numpy.max(b[:, 0])
195         Bi_pos_I_m = numpy.mean(b[:, 0])
196         Bi_pos_I_std = numpy.std(b[:, 0])
197         Bi_neg_I_max = numpy.max(b[:, 1])
198         Bi_neg_I_m = numpy.mean(b[:, 1])
199         Bi_neg_I_std = numpy.std(b[:, 1])
200         print " DEB The highest intensity for all Bijvoet positives is %s, the mean intensity is %s with a standard deviation of
    %s." % (str(Bi_pos_I_max), str(Bi_pos_I_m), str(Bi_pos_I_std))
201         print " DEB The highest intensity for all Bijvoet negatives is %s, the mean intensity is %s with a standard deviation of
    %s." % (str(Bi_neg_I_max), str(Bi_neg_I_m), str(Bi_neg_I_std))
202
203     # plot change of average signed anomalous differences for one reflection for wedges and accumulated data
204     def reflection_analysis(self, im_int, stepsize, key):
205         im_s=im_int[0]
206         im_r=stepsize
207         ano_dif_key=[]
208         ano_dif_key_2=[]
209         I_p=[]
210         I_n=[]
211         x=numpy.linspace(1, im_int[1]/360, num=im_int[1]/360)
212         z=numpy.linspace(0,0, num=im_int[1]/360)
213         # load dictiionaries of average signed anomalous differences for wedges and accumulated data
214         while im_r<= im_int[1]:
215            new_name= "xds_tha1_02_"+str(im_r/360)+"_1/"
216            if os.path.exists(self.path + new_name+"/ano_dif1_all.pic"):
217                with open(self.path  +new_name+"/ano_dif1_all.pic", 'r') as f:
218                    ano_dif1 = pickle.load(f)
219            else:
220                print "ano_dif1 not found"
221            if os.path.exists(self.path + new2+"/ano_dif1_all.pic"):
222                with open(self.path  +new2+"/ano_dif1_all.pic", 'r') as f:
223                  ano_dif2 = pickle.load(f)
224            else:
225                print "ano_dif2 not found"
226            # look for reflection and store average signed anomalous difference
227            if ano_dif1.has_key(key):
228                     print ano_dif1[key][0], ano_dif2[key][0]
229                     ano_dif_key.append(ano_dif1[key][0])
230                     ano_dif_key_2.append(ano_dif2[key][0])
231            print im_r
232            im_r=im_r+stepsize
233         print len(x), len(ano_dif_key), len(ano_dif_key_2)
234         fig1 = pylab.figure(figsize=(10,6), dpi = 300)
235         ax=fig1.add_axes([0.1, 0.1, 0.8, 0.8])
236         fig1.text(0.9, 0.06, r'x360$^\circ$', fontsize=20)
237         ax.plot(x, ano_dif_key, '^b-', label="av. anomalous difference calculated from wedges")
238         ax.plot(x, ano_dif_key_2, '^g-', label="av. anomalous difference calculated from accumulated data")
239         ax.plot(x, z, color='0.5', linestyle='--')
240         ax.set_xlabel(r"oscillation range [$^\circ$]")
241         ax.set_ylabel(r"$\Delta F$")
242         ax.set_xlim(0.5, max(x)+0.5)
243         ax.set_ylim(-6, 6)
244         ax.set_title(r"$\Delta$ F of the (17,10,39) reflection")
245         pylab.legend(loc=1)
246         pylab.savefig("/Users/storm/Desktop/tha6/17_10_39.pdf", dpi=300)
247         pylab.show()
248
249
```

plotting

```
import numpy, scipy
import matplotlib
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec
from matplotlib import rc
from operator import truediv
import sympy
from matplotlib import pylab
from input_data import *
from verifying import *
from anomalous_differences import *
from analysing import *
from helping_routines import *
from matplotlib.ticker import MultipleLocator


# setting font and fontsizes
rc('font', weight='bold', **{'family':'serif','serif':['Computer Modern Roman']})
rcParams['lines.linewidth'] = 1
rc('text', usetex=True)
rcParams['figure.figsize'] = 10, 6
title_font = {'fontname':'Arial', 'size':'24', 'color':'black', 'weight':'normal',
              'verticalalignment':'bottom'}
axis_font = {'fontname':'Arial', 'size':'24'}
annotate_font = {'fontname':'Arial', 'size':'20'}
tick_size='24'
tick_weight='bold'
_dpi=300

class plotting():

    def __init__(self, case, path, name, r_m):
        self.case=case
        self.path=path
        self.name=name
        self.r_m=r_m
        self.ano=anomalous_differences(self.case, self.path, self.name, self.r_m)
        self.analysing=analysing(self.case, self.path, self.name, self.r_m)
        self.hr=helping_routines(self.case, self.path, self.name, self.r_m)

     # plot the average intensities of the Bijvoet positive and Bijvoet negative reflections
    def plot_I_p_vs_I_n(self, _res):
        if os.path.exists(self.path+'./ano_dif1_'+_res+'.pic'):
            with open(self.path+'./ano_dif1_'+_res+'.pic', "r") as f:
                ano_dif=pickle.load(f)
        pair=[]
        for key in ano_dif:
            pair.append((ano_dif[key][2], ano_dif[key][3]))
        p,n=zip(*pair)
        fig = pylab.figure(figsize=(10,6), dpi = _dpi)
        ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])
        if "acc" in self.name:
                    ax.plot(p, n,  "r.")
        else:
                    ax.plot(p, n,  "b.")
        ax.set_xlabel(r"averaged I of the Bijvoet Positives")
        ax.set_ylabel(r"averaged I of the Bijvoet Negatives")
        ax.set_xlim(0, 60)
        ax.set_ylim(0, 1200)
        ax.set_title("Averaged Bijvoet Intensities for the accumulated data of wedge 1 and 2")
        pylab.savefig("/Users/storm/Desktop/tha6/I_p_vs_I_n_"+_res+"_acc_1-2.pdf", dpi=_dpi)

    # scatterplot, e.g. to show the agreement of the <Delta F> values from two wedges
    def scatterplot(self, _res, args):
        with open(args[0], "r") as f:
                f1=pickle.load(f)
        with open(args[1], "r") as g:
                f2=pickle.load(g)
        tit1="wedge 1"
        tit2='wedge 2'
        t=[]
        common_keys = f1.viewkeys() & f2.viewkeys()
        for key in common_keys:
            print f1[key][-1]
            t.append((f1[key][0], f2[key][0]))
        x, y= zip(*t)
        _x=numpy.linspace(-10, 10, num=21)
        m=numpy.polyfit(x, y, deg=1)[0]
        m=round(m, ndigits=2)
        b=numpy.polyfit(x, y, deg=1)[1]
        fit=numpy.polyval([m, b], _x)
        CC=100*numpy.corrcoef(x, y)[0][1]
        CC=round(CC, ndigits=2)
        fig = pylab.figure(figsize=(10,8), dpi = _dpi)
        ax = fig.add_axes([0.12, 0.1, 0.8, 0.8])
```

plotting

```
89          ax.plot(_x, _x, color='0.5', linestyle='--')
90          ax.plot(_x, fit, 'r-')
91          ax.plot(x, y, 'b.')
92          ax.legend(loc=2)
93          ax.set_xlabel(r'$\Delta $F [a.u.] ', **axis_font)
94          ax.set_ylabel(r'$\Delta$ F [a.u.] ', **axis_font)
95          ax.annotate(r'CC=%s %%' %CC, xy=(0.05, 0.95), xycoords='axes fraction', horizontalalignment='left',
  verticalalignment='top', **annotate_font)
96          ax.annotate(r'm=%s' %m, xy=(0.05, 0.9), xycoords='axes fraction', horizontalalignment='left', verticalalignment='top',
  **annotate_font)
97          ax.annotate(r'number of data points=%s' %len(x), xy=(0.05, 0.85), xycoords='axes fraction', horizontalalignment='left',
  verticalalignment='top', **annotate_font)
98          ax.set_xlim(-10, 10)
99          ax.set_ylim(-10, 10)
100         ax.legend(loc=2, fontsize=20)
101         for item in (ax.get_xticklabels() + ax.get_yticklabels()):
102                 item.set_fontsize(tick_size)
103                 item.set_weight(tick_weight)
104         pylab.savefig("/Users/storm/git/Thesis/Thesis/scatter.pdf")
105
106     # plotting_old I/sigma versus resolution
107     def plot_isa_vs_res(self, res_shells):
108         ano=self.ano.Ano_Dif1(self.path)
109         tit1=self.path.split("/")[-2]
110         pair=[]
111         x=[]
112         y=[]
113         for key in ano:
114             pair.append((float(ano[key][2]),float(ano[key][0]), float(ano[key][1])))
115         s_pair=sorted(pair, reverse=True)
116         res_shells.append(s_pair[0][0])
117         res_shells=sorted(res_shells, reverse=True)
118         pairarr = numpy.asarray(s_pair)
119         I=[]
120         sig=[]
121         res=[]
122         for ii in xrange(len(res_shells)-1):
123             cc = numpy.where(numpy.where(pairarr[:,0] <= res_shells[ii], 1, 0) * numpy.where(pairarr[:,0] >= res_shells[ii+1], 1,
  0) == 1)[0]
124             sum_I = numpy.sum(pairarr[cc,1])
125             sum_sig=numpy.sum(pairarr[cc,2])
126             I.append(sum_I/(len(cc)))
127             sig.append(sum_sig/(len(cc)))
128         I=numpy.asarray(I)
129         sig=numpy.asarray(sig)
130         y=I/sig
131         x=res_shells[1:]
132         x=numpy.asarray(x)
133         y=numpy.asarray(y)
134         pylab.plot(x, y, 'b>-')
135         pylab.xlim(max(x)+1, (min(x)-1))
136         pylab.xlabel(r'resolution/ $ \AA$', fontsize=17)
137         pylab.ylabel(r"<I/$\sigma$>", fontsize=17)
138         pylab.title("I/sigma vs. Resolution of "+tit1)
139         pylab.savefig(self.path+"Anomalous_Signal_vs_res")
140         pylab.show()
141         print "DEB The anomalous differences of %s have been plotted vs. resolution." %tit1
142
143
144     #plotting_old the correlation coefficient of the anomalous differences versus resolution
145     def CC_vs_res(self, res_shells, args):
146         list=[]
147         shelxc_results=[68.7, 95.7, 86.1, 87.4, 84.1, 79.6, 74.1, 63.5, 49.8, 34.6, 18.7]
148         tit1="wedge 1"
149         tit2="wedge 2"
150         with open(args[0], "r") as f:
151                 f1=pickle.load(f)
152         with open(args[1], "r") as g:
153                 f2=pickle.load(g)
154         common_keys = f1.viewkeys() & f2.viewkeys()
155         for key in common_keys:
156             ano_1=f1[key][0]
157             res=f1[key][5]
158             print res
159             ano_2=f2[key][0]
160             list.append((ano_1, ano_2, res))
161         list=sorted(list, key=lambda tup: tup[2])
162         res_shells=sorted(res_shells, reverse=True)
163         print res_shells
164         shells=[]
165         # sorting the anomalous differences by resolution shell
166         for ii in xrange(0,len(res_shells)-1):
167             s=[]
168             print res_shells[ii], res_shells[ii+1]
169             for el in list:
170                 if el[2] < res_shells[ii] and el[2] > res_shells[ii+1]:
171                     s.append(el)
172             print len(s)
```

```
173             shells.append(s)
174         print len(shells)
175         res=[]
176         CC=[]
177         for shell in shells:
178             print len(shell)
179             shell=numpy.asarray(shell)
180             x,y,r=zip(*shell)
181             res.append(numpy.mean(r))
182             cc=100*numpy.corrcoef(x,y)[0][1]
183             print cc
184             CC.append(cc)
185         fig1 = pylab.figure(figsize=(10,8), dpi = _dpi)
186         fig=fig1.add_axes([0.12, 0.12, 0.8, 0.8])
187         fig.plot(res, CC,'r^-', label=r"self calculated")
188         fig.plot(res, shelxc_results,'b>-', label=r"calculated by SHELXC")
189         fig.set_xlabel(r'resolution/$\r{A}$', **axis_font)
190         fig.set_ylabel(r"CC/\%", **axis_font)
191         fig.set_xlim(max(res)+1, (min(res)-1))
192         fig.legend(loc=3, prop={'size':20})
193         for item in (fig.get_xticklabels() + fig.get_yticklabels()):
194                 item.set_fontsize(tick_size)
195                 item.set_weight(tick_weight)
196         fig.title(r'CC of the signed anomalous differences of in different resolution shells', fontsize=14)
197         pylab.savefig("/Users/storm/git/Thesis/Thesis/CC_vs_res.pdf")
198 
199     # plot the average anomalous signal within a certain image interval
200     def plot_ano_vs_frameInt(self, images, fr_dif):
201         y=[]
202         x=[]
203         for i in xrange(images/fr_dif):
204             dF=[]
205             interval=[i*fr_dif, (i+1)*fr_dif]
206             ano_dif=self.ano.Ano_Dif4(self.path, interval)
207             for key in ano_dif:
208                 dF.append(ano_dif[key][2])
209             av_dif=numpy.sum(dF)/len(dF)
210             y.append(av_dif)
211             x.append((i+(i+1))/2)
212         pylab.plot(x, y,'r^-')
213         pylab.xlabel(r'image interval', fontsize=14)
214         pylab.ylabel("Anomalous Signal", fontsize=14)
215         pylab.title(r'Average anomalous signal within an interval of '+str(fr_dif)+ 'images', fontsize=14)
216         pylab.savefig(self.path+"_ano_vs_frame_Int.png")
217         pylab.show()
218 
219 # plot <Delta F> versus resolution
220     def plot_ano_dif1_vs_res(self, _res):
221         if os.path.exists(self.path+'./ano_dif1_'+_res+'.pic'):
222             with open(self.path+'./ano_dif1_'+_res+'.pic', "r") as f:
223                 ano_dif=pickle.load(f)
224         pair=[]
225         for key in ano_dif:
226             pair.append((ano_dif[key][-1], ano_dif[key][0]))
227         x,y=zip(*pair)
228         fig = pylab.figure(figsize=(10,6), dpi = _dpi)
229         ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])
230         if "acc" in self.name:
231                     ax.plot(x, y,  "r.", label=self.name)
232         else:
233                     ax.plot(x, y,  "b.", label=self.name)
234         ax.set_xlabel(r"resolution [$\r{A}]$")
235         ax.set_ylabel(r"average signed [$\Delta$ F]")
236         ax.set_ylim(-8, 8)
237         ax.set_title(r"Average signed anomalous differences vs. resolution for the first three accumulated turns")
238         pylab.savefig("/Users/storm/Desktop/tha6/AnoDif_vs_res_acc_1-3_"+_res+".pdf", dpi=_dpi)
239 
240 # plotting_old the histogram of the average anomalous differences, optionally with a gaussian fit
241     def plot_AnoDif1_hist(self, im_int, stepsize, resolution, mode, _res, limit, _type, out):
242         print self.path+self.name+"_"+_res+'_ano_dif1_hist_180.pic'
243         if os.path.exists(self.path+self.name+"_"+_res+'_ano_dif1_hist_180.pic'):
244             print "DEB Histograms are loaded"
245             with open(self.path+self.name+"_"+_res+'_ano_dif1_hist_180.pic', "r") as g:
246                 no, arrays, params=pickle.load(g)
247         else:   arrays, params= self.analysing.analyse_Ano_Dif1(im_int, stepsize, resolution, mode, _res, limit, "hist")
248         fig1 = pylab.figure(figsize=(10,6), dpi = _dpi)
249         fig=fig1.add_axes([0.12, 0.12, 0.8, 0.8])
250         for i in xrange(0,len(arrays)):
251             print " DEB %s array(s) are loaded. "%str(len(arrays))
252             bins = numpy.linspace(-1*limit, limit, num=101)
253             _y, _x = numpy.histogram(arrays[i], bins=bins)
254             _x = bins[:-1] + (bins[1] - bins[0]) / 2
255             if  _type=="hist":
256                 fig.bar(_x, _y, width = (_x[1]- _x[0]), color='g', align = 'center')
257                 fig.set_xlabel(r"$<\Delta F>$", **axis_font)
258                 fig.set_ylabel(r"frequencies", **axis_font)
259                 # fig.annotate(r'%s average signed anomalous differences' %str(len(arrays[i])), xy=(0.6, 0.95), xycoords='axes
    fraction', horizontalalignment='left', verticalalignment='top', **annotate_font)
```

plotting

```
260                 fig.set_title("Histogram of the average anomalous differences", y=1.05, **title_font)
261                 # fig.annotate(r'%s average signed anomalous differences' %str(len(anodif)), xy=(0.85, 0.95), xycoords='axes
    fraction', horizontalalignment='left', verticalalignment='top', **annotate_font)
262             if "hist+fit" in _type:
263                 fitfunc= lambda p, x: p[0]*numpy.exp(-0.5*((x-p[1])/p[2])**2)
264                 errfunc= lambda p, x, y: fitfunc(p, x) - y
265                 fit_params=params[i].tolist()
266                 xnew=numpy.linspace(min(_x), max(_x), num=10000)
267                 if 'acc' in self.name:
268                     fig.bar(_x, _y, width = (_x[1]- _x[0]), color='r', align = 'center')
269                 else:
270                     fig.bar(_x, _y, width = (_x[1]- _x[0]), color='b', align = 'center')
271                     fig.plot(xnew, fitfunc(fit_params, xnew), 'r--', linewidth=2)
272                 fig.plot(xnew, fitfunc(fit_params, xnew), 'k--', linewidth=2)
273                 fig.set_xlabel(r"$<\Delta F>$", **axis_font)
274                 fig.set_ylabel(r"frequencies", **axis_font)
275                 fig.set_ylim(0, 2300)
276                 fig.annotate(r'%s av. signed anomalous differences' %str(len(arrays[i])), xy=(0.45, 0.96), xycoords='axes
    fraction', horizontalalignment='left', verticalalignment='top', **annotate_font)
277                 fig.annotate(r'Fit: A$\cdot e^{-0.5\cdot(\frac{x-x_0}{\sigma})^2}$', xy=(0.05, 0.96), xycoords='axes fraction',
    horizontalalignment='left', verticalalignment='top', **annotate_font)
278                 fig.annotate(r'A  = %s' % str(round(params[i][0], ndigits=3)), xy=(0.05, 0.87), xycoords='axes fraction',
    horizontalalignment='left', verticalalignment='top', **annotate_font)
279                 fig.annotate(r'$\sigma$  = %s' % str(round(params[i][2], ndigits=3)), xy=(0.05, 0.80), xycoords='axes fraction',
    horizontalalignment='left', verticalalignment='top', **annotate_font)
280                 fig.annotate(r'$x_0$ =  %s' % str(round(params[i][1], ndigits=3)), xy=(0.05, 0.73), xycoords='axes fraction',
    horizontalalignment='left', verticalalignment='top', **annotate_font)
281                 fig.set_ylim(0, 2200)
282             for item in (fig.get_xticklabels() + fig.get_yticklabels()):
283                 item.set_fontsize(tick_size)
284                 item.set_weight(tick_weight)
285             pylab.savefig(out[0]+self.name+"_"+_res+"_"+_type+"_"+str((i+1)*360)+"_180_"+out[1], dpi=_dpi)
286             pylab.clf()
287
288 # plotting_old the fits with normal distributions of the normed histograms of the average anomalous differences
289     def plot_AnoDif1_normed_fits(self, im_int, stepsize, resolution, limit, kind, dose, annotation, _res, out):
290         print self.path+self.name+'_'+_res+'_ano_dif1_norm_fit.pic'
291         if os.path.exists(self.path+self.name+'_'+_res+'_ano_dif1_norm_fit.pic'):
292             print " DEB Normalized arrays are loaded."
293             with open(self.path+self.name+'_'+_res+'_ano_dif1_norm_fit.pic', "r") as n:
294                 no, normed_ano_dif1s, params, err=pickle.load(n)
295                 print no
296                 print len(no)
297         else:
298              normed_ano_difs1s, params, err= self.analysing.analyse_Ano_Dif1(im_int, stepsize, resolution, 'I', _res, limit,
    "norm_fit")
299         print self.path+self.name+'_acc_'+_res+'_ano_dif1_norm_fit.pic'
300         if os.path.exists(self.path+self.name+'_acc_'+_res+'_ano_dif1_norm_fit.pic'):
301             print " DEB Normalized arrays are loaded."
302             with open(self.path+self.name+'_acc_'+_res+'_ano_dif1_norm_fit.pic', "r") as n:
303                   sno, snormed_ano_dif1s, sparams, serr=pickle.load(n)
304                   print sno
305         else:
306             snormed_ano_dif1s, sparams, serr= self.analysing.analyse_Ano_Dif1(im_int, stepsize, resolution, 'I', _res, limit,
    "norm_fit")
307         if len(params)>1:
308             sigma, x_0=zip(*params)
309             sig_err, x_0_err=zip(*err)
310             ssigma, sx_0=zip(*sparams)
311             ssig_err, sx_0_err=zip(*serr)
312         else:
313             sigma, x_0=params[0].tolist()
314             sig_err, x_0_err=err[0]
315             ssigma, sx_0=sparams[0].tolist()
316             ssig_err, sx_0_err=serr[0]
317         fitfunc = lambda p, x:  (1/(numpy.sqrt(2*numpy.pi)*p[0]))* numpy.exp(-0.5 * ((x - p[1]) / p[0]) ** 2)
318         errfunc = lambda p, x, y: fitfunc(p, x) - y
319         # plot the normalized fits
320         if '2d' in kind:
321             _x=numpy.linspace(-1*limit, limit, 1000)
322             colors = len(normed_ano_dif1s)
323             cm = plt.get_cmap('jet')
324             fig1 = pylab.figure(figsize=(10,8), dpi = _dpi)
325             ax = fig1.add_subplot(111)
326             ax.set_color_cycle([cm(1.*i/colors) for i in range(colors)])
327             pylab.subplots_adjust(left=0.125, right=1.1, bottom=0.1, top=0.9, wspace=0.2, hspace=0.5)
328             if "acc" in self.name:
329                 c=1
330                 ax.plot(_x, fitfunc(params[0], _x), linestyle='-', label=r"wedge "+str(1)+","+ " $x_0$="+str(round(params[0][1],
    ndigits=3))+", $\sigma$="+str(round(params[0][0], ndigits=2)))
331                 for param in params[1:]:
332                     print round(param[0], ndigits=2)
333                     if c%2 ==0:
334                         print max(fitfunc(param, _x))
335                         ax.plot(_x, fitfunc(param, _x), linestyle='-', label=r"wedge 1-"+str(c+1)+","+ "
    $x_0$="+str(round(param[1], ndigits=3))+", $\sigma$="+str(round(param[0], ndigits=2)))
336                     else:
337                         ax.plot(_x, fitfunc(param, _x), linestyle='--', label=r"wedge 1-"+str(c+1)+","+ "
```

```
    $x_0$="+str(round(param[1], ndigits=3))+", $\sigma$="+str(round(param[0], ndigits=2)))
338                     c=c+1
339             else:
340                 c=0
341                 for param in params[0:]:
342                     print round(param[0], ndigits=2)
343                     if c%2 ==0:
344                         ax.plot(_x, fitfunc(param, _x), linestyle='-', label=r"wedge "+str(c+1)+","+ " $x_0$="+str(round(param[1],
    ndigits=3))+", $\sigma$="+str(round(param[0], ndigits=2)))
345                     else:
346                         ax.plot(_x, fitfunc(param, _x), linestyle='--', label=r"wedge "+str(c+1)+","+ " $x_0$="+str(round(param[1],
    ndigits=3))+", $\sigma$="+str(round(param[0], ndigits=2)))
347                     c=c+1
348             ax.set_xlabel(r"$<\Delta F>$ / [a.u.] ", **axis_font)
349             ax.set_ylabel(r"fit of relative frequencies / [a.u.] ", **axis_font)
350             ax.set_xlim(-1*limit, limit)
351             ax.set_ylim(0, numpy.max(normed_ano_dif1s))
352             ax.legend(bbox_to_anchor=(1.01, 1.0), loc=2, borderaxespad=0., prop={'size':16})
353             for item in (ax.get_xticklabels() + ax.get_yticklabels()):
354                 item.set_fontsize(tick_size)
355                 item.set_weight('medium')
356             ax.set_ylim(0, 2.0)
357             fig1.text(0.97, 0.85, annotation, **annotate_font)
358             pylab.savefig(out[0]+self.name+"_"+kind+"_"+out[1], bbox_inches='tight', dpi=_dpi)
359         # plot the normalized histograms and the fitted normal distribution
360         if 'normed_hist+fit' in kind:
361             fig1 = pylab.figure(figsize=(10,6), dpi = _dpi)
362             fig=fig1.add_axes([0.1, 0.1, 0.75, 0.75])
363             for i, anodif in enumerate(normed_ano_dif1s):
364                 bins = numpy.linspace(-1*limit, limit, num=101)
365                 x = bins[:-1] + (bins[1] - bins[0]) / 2
366                 fig.bar(x, anodif, width = (x[1]- x[0]), color='g', align = 'center')
367                 fig.plot(x, fitfunc(params[i], x), linestyle='-', color='red')
368                 fig.set_xlabel(r"$<\Delta F>$",**axis_font)
369                 fig.set_ylabel(r"rel. frequencies", **axis_font)
370                 fig.annotate(r'Fit: $\frac{1}{\sqrt{2\pi}\sigma}\cdot e^{-0.5\cdot(\frac{x-x_0}{\sigma})^2}$', xy=(0.05, 0.95),
    xycoords='axes fraction', horizontalalignment='left', verticalalignment='top', **axis_font)
371                 fig.annotate(r'$\sigma$  = %s' % str(round(params[i][0], ndigits=3)), xy=(0.05, 0.85), xycoords='axes fraction',
    horizontalalignment='left', verticalalignment='top', **axis_font)
372                 fig.annotate(r'$x_0$ = %s' % str(round(params[i][1], ndigits=3)), xy=(0.05, 0.8), xycoords='axes fraction',
    horizontalalignment='left', verticalalignment='top', **axis_font)
373                 fig.set_title("Normalized histogram of the average anomalous differences fitted with a normal distribution",
    y=1.05, **title_font)
374                 pylab.savefig(out[0]+"_"+kind+"_"+out[1], bbox_inches='tight', dpi=_dpi)
375                 pylab.show()
376         # the 3d mode is meant for the distribution of the accumulated signed average anomalous differences to visualize the change
    when more data are added
377         if '3d' in kind:
378             fig = pylab.figure(figsize=(10,8), dpi = _dpi)
379             ax = fig.add_axes([0.0, 0.0, 0.9, 0.9], projection = '3d')
380             pylab.subplots_adjust(left=0.125, right=1.1, bottom=0.1, top=0.95, wspace=0.2, hspace=0.5)
381             cm = plt.get_cmap('jet')
382             ax.set_color_cycle([cm(1.*i/len(normed_ano_dif1s)) for i in range(len(normed_ano_dif1s))])
383             _x=numpy.linspace(-limit, limit, 1000)
384             z=[1,5,10,15,20]
385             x=[-3,-2,-1,0,1,2,3]
386             rounds=(im_int[1]/stepsize)+1
387             if "acc" in self.name:
388                 ax.plot(_x, fitfunc(params[i-1], _x), zs=i-1, zdir="z", linestyle='-', label=r"wedge "+str(i)+","+ "
    $x_0$="+str(round(params[i-1][1], ndigits=3))+", $\sigma$="+str(round(params[i-1][0], ndigits=2)))
389                 for i in xrange(2, rounds):
390                     ax.plot(_x, fitfunc(params[i-1], _x), zs=i-1, zdir="z", linestyle='-', label=r"wedge 1-"+str(i)+","+ "
    $x_0$="+str(round(params[i-1][1], ndigits=3))+", $\sigma$="+str(round(params[i-1][0], ndigits=2)))
391             else:
392                 for i in xrange(1, rounds):
393                     ax.plot(_x, fitfunc(params[i-1], _x), zs=i-1, zdir="z", linestyle='-', label=r"wedge "+str(i)+","+ "
    $x_0$="+str(round(params[i-1][1], ndigits=3))+", $\sigma$="+str(round(params[i-1][0], ndigits=2)))
394             a=ax.set_xlabel(r"$<\Delta F>$ / [a.u.] ", **axis_font)
395             b=ax.set_ylabel(r"fit of rel. frequencies / [a.u.] ", **axis_font)
396             c=ax.set_zticks(z)
397             ax.set_zlabel("oscillation [x360$^\circ$]", **axis_font)
398             ax.view_init(elev=-73, azim=90)
399             ax.set_zlim(1, max(z))
400             ax.xaxis._axinfo['label']['space_factor'] = 2.0
401             ax.yaxis._axinfo['label']['space_factor'] = 2.2
402             ax.zaxis._axinfo['label']['space_factor'] = 2.2
403             ax.set_xticklabels(x, verticalalignment='baseline', horizontalalignment='left')
404             ax.legend(bbox_to_anchor=(0.9, 0.9), loc=2, borderaxespad=0., prop={'size':16})
405             for item in (ax.get_xticklabels() + ax.get_yticklabels() +ax.get_zticklabels()):
406                 item.set_fontsize(tick_size)
407                 item.set_weight('medium')
408             pylab.savefig(out[0]+self.name+"_"+kind+"_"+out[1], bbox_inches='tight', dpi=_dpi)
409             pylab.show()
410         if 'sigma' in kind:
411             fig = pylab.figure(figsize=(10,6), dpi = _dpi)
412             ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])
413             x=numpy.arange(1, (im_int[1]/360)+1, 1)
414             # calculating the error
```

plotting

```
415             sys_err=[numpy.sqrt(i)/i for i in no]
416             ssys_err=[numpy.sqrt(k)/k for k in sno]
417             fit_err=map(truediv, sig_err, sigma)
418             sfit_err=map(truediv, ssig_err, ssigma)
419             t_err_p=[]
420             st_err_p=[]
421             for e, _e in zip(sys_err, fit_err):
422                 t_err_p.append(numpy.sqrt(e**2+_e**2))
423             for es, _es in zip(ssys_err, sfit_err):
424                 st_err_p.append(numpy.sqrt(es**2+_es**2))
425             t_err=[]
426             for e_, s in zip(sigma, t_err_p):
427                 t_err.append(e_*s)
428             st_err=[]
429             for se_, ss in zip(ssigma, st_err_p):
430                 st_err.append(se_*ss)
431             # start plotting_old
432             ax.errorbar(x, ssigma, yerr=st_err, marker="^", color="red", label="accumulated data")
433             print ssigma
434             ax.errorbar(x, sigma, yerr=t_err, marker="^", color="blue", label="data in wedges")
435             pylab.legend(loc=2, prop={'size':20})
436             print sigma
437             ax.set_xlim(0.5, im_int[1]/360+0.5)
438             ax.set_ylim(0, 1.6)
439             ax.set_xlabel(r"oscillation range [$^\circ$]", **axis_font)
440             fig.text(0.9, 0.055, r'x360$^\circ$', **axis_font)
441             ax2 = ax.twiny()
442             ax2.set_xlabel("dose [MGy]", labelpad=10, **axis_font)
443             _dose=numpy.linspace(dose, max(x)*dose, 5)
444             _dose=[round(d, ndigits=2) for d in _dose]
445             ax2.set_xlabel("dose [MGy]", labelpad=10, **axis_font)
446             ax2.set_xlim(0.5*dose, len(x)*dose+0.5*dose)
447             ax2.set_xticks(_dose)
448             ax.annotate(annotation, color='black', xy=(0.85, 0.95), xycoords='axes fraction', horizontalalignment='left',
    verticalalignment='top', **annotate_font)
449             ax.set_ylabel(r"$\sigma$ [a.u.]", **axis_font)
450             for item in (ax.get_xticklabels() + ax.get_yticklabels() +ax2.get_xticklabels()):
451                 item.set_fontsize(tick_size)
452                 item.set_weight('medium')
453             pylab.savefig(out[0]+self.name+"_"+kind+"_"+_res+"_"+out[1], bbox_inches='tight', dpi=_dpi)
454         # plotting_old the change of x_0 as a parameter of the normal distributions
455         if 'x_0' in kind:
456             fig = pylab.figure(figsize=(10,6), dpi = _dpi)
457             ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])
458             x=numpy.arange(1, (im_int[1]/360)+1, 1)
459             if "acc" in self.name:
460                 ax.errorbar(x, x_0, yerr=x_0_err, marker="^", color="red", label=self.name)
461             else:
462                 ax.errorbar(x, x_0, yerr=x_0_err, marker="^", color="blue", label=self.name)
463             ax2 = ax.twinx()
464             dlim=len(sigma)*dose+(0.5*dose)
465             print dlim
466             _dose=numpy.linspace(dose, len(sigma)*dose, num=5)
467             ax2.set_xlabel(r"dose [MGy]",  **axis_font)
468             ax2.set_xticks(_dose)
469             ax.set_xlim(0.5, max(x)+0.5)
470             ax2.set_xlim(0.5*dose, dlim)
471             ax.set_xlabel("oscillation range [$^\circ$]", **axis_font)
472             ax.set_ylabel("$x_0$ [a.u.]", **axis_font)
473             fig.text(0.9, 0.1, r'x360$^\circ$', fontsize=20)
474             for item in (ax.get_xticklabels() + ax.get_yticklabels() +ax2.get_xticklabels()):
475                 item.set_fontsize(20)
476                 item.set_weight('medium')
477             pylab.savefig(out[0]+self.name+"_"+kind+"_"+out[1], bbox_inches='tight', dpi=_dpi)
478         # plotting_old the change in the maxima of the normal distributions
479         if 'max' in kind:
480             _m=[]
481             _x=numpy.linspace(-1*limit, limit, 1000)
482             fig = pylab.figure(figsize=(10,6), dpi = _dpi)
483             ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])
484             x=numpy.arange(1, (im_int[1]/360)+1, 1)
485             for param in params[0:]:
486                _m.append(max(fitfunc(param, _x)))
487             print _m
488             if "acc" in self.name:
489                 ax.plot(x, _m, marker="^", color="red")
490             else:
491                 ax.plot(x, _m, marker="^", color="blue")
492             ax2 = ax.twiny()
493             dlim=len(sigma)*dose+(0.5*dose)
494             _dose=numpy.linspace(dose, len(sigma)*dose, num=5)
495             ax2.set_xlabel(r"dose [MGy]",  **axis_font)
496             ax2.set_xticks(_dose)
497             ax.set_xlim(0.5, max(x)+0.5)
498             ax2.set_xlim(0.5*dose, dlim)
499             ax.set_ylim(0, max(_m)+0.2)
500             ax.set_xlabel("oscillation range [$^\circ$]", **axis_font)
501             ax.set_ylabel("$x_0$ [a.u.]", **axis_font)
```

```
502             fig.text(0.9, 0.06, r'x360$^\circ$', fontsize=20)
503             for item in (ax.get_xticklabels() + ax.get_yticklabels() +ax2.get_xticklabels()):
504                 item.set_fontsize(20)
505                 item.set_weight('medium')
506             pylab.savefig(out[0]+self.name+"_isa_"+kind+"_"+out[1], bbox_inches='tight', dpi=_dpi)
507 
508     # plotting_old the sigma values of the normal distributions and fitting them (metric)!
509     def plot_sigma_fit(self, im_int, stepsize, resolution, limit, dose, annotation, _res, out):
510         print self.path+self.name+'_'+_res+'_ano_dif1_norm_fit.pic'
511         if os.path.exists(self.path+self.name+'_'+_res+'_ano_dif1_norm_fit.pic'):
512              print " DEB Normalized arrays are loaded."
513              with open(self.path+self.name+'_'+_res+'_ano_dif1_norm_fit.pic', "r") as n:
514                  no, normed_ano_dif1s, params, err=pickle.load(n)
515         else:
516             no, normed_ano_dif1s, params, err= self.analysing.analyse_Ano_Dif1(im_int, stepsize, resolution, 'I', _res, limit,
    "norm_fit")
517         print self.path+self.name+'_acc_'+_res+'_ano_dif1_norm_fit.pic'
518         if os.path.exists(self.path+self.name+'_acc_'+_res+'_ano_dif1_norm_fit.pic'):
519            print " DEB Normalized arrays are loaded."
520            with open(self.path+self.name+'_acc_'+_res+'_ano_dif1_norm_fit.pic', "r") as n:
521              sno, snormed_ano_dif1s, sparams, serr=pickle.load(n)
522         else:
523             sno, snormed_ano_dif1s, sparams, serr=self.analysing.analyse_Ano_Dif1(im_int, stepsize, resolution, 'I', _res, limit,
    "norm_fit")
524         if len(params)>1:
525             sigma, x_0=zip(*params)
526             sig_err, x_0_err=zip(*err)
527             ssigma, sx_0=zip(*sparams)
528             ssig_err, sx_0_err=zip(*serr)
529         else:
530             sigma, x_0=params[0].tolist()
531             sig_err, x_0_err=err[0]
532             ssigma, sx_0=sparams[0].tolist()
533             ssig_err, sx_0_err=serr[0]
534         # add systematic error according to sqrt(n) to fitting error
535         sys_err=[numpy.sqrt(i)/i for i in no]
536         ssys_err=[numpy.sqrt(k)/k for k in sno]
537         fit_err=map(truediv, sig_err, sigma)
538         sfit_err=map(truediv, ssig_err, ssigma)
539         t_err_p=[]
540         st_err_p=[]
541         for e, _e in zip(sys_err, fit_err):
542             t_err_p.append(numpy.sqrt(e**2+_e**2))
543         for es, _es in zip(ssys_err, sfit_err):
544             st_err_p.append(numpy.sqrt(es**2+_es**2))
545         t_err=[]
546         for e_, s in zip(sigma, t_err_p):
547             t_err.append(e_*s)
548         st_err=[]
549         for se_, ss in zip(ssigma, st_err_p):
550             st_err.append(se_*ss)
551         print t_err_p, st_err_p
552         # define points for different fits
553         y, y_err, z, z_err= [], [], [], []
554         x=numpy.arange(1, len(sigma)+1)
555         print len(x), len(sigma)
556         # get the sigma > 0 and write them to z
557         a=numpy.where(sigma>(sigma[0]))
558         print a[0]
559         for n in a[0]:
560             z.append(sigma[n])
561             z_err.append(t_err[n])
562         x2 = x[a[0]]
563         print "sigma > first sigma", a[0]
564         print "x2", x2
565         print "first sigma value", sigma[0]
566         # get the sigma < 0 and write them to y
567         b=numpy.where(sigma<=(sigma[0]))
568         for _n in b[0]:
569              y.append(sigma[_n])
570              y_err.append(t_err[_n])
571         x1=b[0]
572         print "x1", x1
573         print "sigma < first sigma", b[0]
574         print b
575         fig = pylab.figure(figsize=(10,6), dpi = _dpi)
576         ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])
577         #first linear fit for all sigmas < sigma[0]
578         def line(x, m, b):
579             return m*x+b
580         p0=[-0.005, 0.48]
581         p0=numpy.asarray(p0)
582         popt_1, pcov=scipy.optimize.curve_fit(line, x1, y, p0=p0, sigma=y_err)
583         print "results optimize linear fit 1", popt_1
584         # fit 2 with error bars
585         p1=[0.03, 0.34]
586         p1=numpy.asarray(p1)
587         popt_2, pcov=scipy.optimize.curve_fit(line, x2, z, p0=p1, sigma=z_err)
```

plotting

```
588         print "results optimize linear fit 2", popt_2
589         # parabel fit
590         # def parabel(x, a, b, c):
591         #     return a*x**2+b*x+c
592         #  p0=numpy.asarray([0.0015, 0.0013, 0.46])
593         #  popt_1, pcov=scipy.optimize.curve_fit(parabel, x, sigma, p0=p0, sigma=t_err)
594         #  print "parabolic fit with parameters ", popt_1
595         # polynomial fit
596         # def polynom_3(x, a, b, c, d):
597         #     return a*x**3+b*x**2+c*x+d
598         # p0=numpy.asarray([0.01, 0.0015, 0.0013, 0.46])
599         # popt_1, pcov=scipy.optimize.curve_fit(polynom_3, x, sigma, p0=p0, sigma=t_err)
600         # print "polynomial fit with parameters ", popt_1
601         # exponential fit
602         # def exponential(x, a, b, c):
603         # return a*b**x+c
604         # p0=[0.05, 1.1, 0.4]
605         # p0=numpy.asarray(p0)
606         # popt_1, pcov=scipy.optimize.curve_fit(exponential, x, sigma, p0=p0, sigma=t_err)
607         # print "exponential fit with parameters ", popt_1
608         # fit of the accumulated data
609         def f(x, c, d):
610             return numpy.sqrt(c/x)+d
611         p=[0.122, 0.127]
612         p=numpy.asarray(p)
613         popt_3, pcov=scipy.optimize.curve_fit(f, x, ssigma, p0=p, sigma=st_err)
614         print "fit3 parameter for the accumulated data", popt_3
615         ax2 = ax.twiny()
616         ax2.set_xlabel("dose [MGy]", labelpad=10, **axis_font)
617         _dose=numpy.linspace(dose, max(x)*dose, 5)
618         _dose=[round(d, ndigits=2) for d in _dose]
619         ax2.set_xlabel("dose [MGy]", labelpad=10, **axis_font)
620         ax2.set_xlim(0.5*dose, len(x)*dose+0.5*dose)
621         ax2.set_xticks(_dose)
622         a=ax.errorbar(x, sigma, yerr=t_err, marker="^", color="blue", label="data in wedges", zorder=-1)
623         b=ax.errorbar(x, ssigma, yerr=st_err, marker="^", color="red", label="accumulated data", zorder=-1)
624         pylab.legend(loc=1)
625         # fitting sigma
626     #     ax.plot(x, sigma[0], label="y_1=%.3f" %sigma[0], color='green',  linestyle="-", zorder=1)
627     #     ax.plot(x, parabel(x, popt_1[0], popt_1[1], popt_1[2]), label=r"$y_1$= %.4f x^2 + %.4f x+ %.4f" %(popt_1[0], popt_1[1],
    popt_1[2]), color='cyan')
628     #     ax.plot(x, polynom_3(x, popt_1[0], popt_1[1], popt_1[2], popt_1[3]), label=r"$y_1$= %4.f x^3 + %.4f x^2 + %.4f x+ %.4f" %
    (popt_1[0], popt_1[1], popt_1[2], popt_1[3]), color='black')
629     #     ax.plot(x, exponential(x, popt_1[0], popt_1[1], popt_1[2]), label=r"$y_1$= %.4f $\cdot$ %.4f^x -  %.4f " %(popt_1[0],
    popt_1[1], popt_1[2]), color='cyan')
630         ax.plot(x, line(x, popt_1[0], popt_1[1]), label=r"$y_1$= %.3f x + %.3f" %(popt_1[0], popt_1[1]), color='green',
    linestyle="-", zorder=1)
631         ax.plot(x, line(x, popt_2[0], popt_2[1]), label=r"$y_2$= %.3f x + %.3f" %(popt_2[0], popt_2[1]), color='cyan',
    linestyle="-", zorder=1)
632      #   ax.plot(x, fitfunc_lin(p2, x), label=r"$y_2$="+str(p1[0])+"x+"+str(p1[1]), color='red')
633      # fitting the accumulated data
634         e=ax.plot(x, f(x, popt_3[0], popt_3[1]), label=r"$y_3$= $\sqrt{\frac{%.3f}{x}}+%.3f$" %(popt_3[0], popt_3[1]),
    color='black', linestyle="-", zorder=1)
635     #   ax.plot(x, 0*x+ssigma[-1], label=r"$y_3$=%.3f" %(ssigma[-1]), color="green")
636         ax.set_xlim(0.5, im_int[1]/360+0.5)
637         ax.set_ylim(0, 1.5)
638         ax.set_xlabel(r"oscillation range [$^\circ$]", **axis_font)
639   #       ax.annotate(r'x=3.6 where $y_1=y_2$', color='black', xy=(0.68, 0.95), xycoords='axes fraction',
    horizontalalignment='left', verticalalignment='top', **annotate_font)
640         fig.text(0.9, 0.055, r'x360$^\circ$', **axis_font)
641         ax.annotate(annotation, color='black', xy=(0.05, 0.81), xycoords='axes fraction', horizontalalignment='left',
    verticalalignment='top', **annotate_font)
642         ax.set_ylabel(r"$\sigma$ [a.u.]", **axis_font)
643         ax.legend(loc=2)
644         for item in (ax.get_xticklabels() + ax.get_yticklabels() + ax2.get_xticklabels()):
645                 item.set_fontsize(tick_size)
646                 item.set_weight('medium')
647         pylab.savefig(out[0]+self.name+"_sigma_fit_"+_res+"_"+out[1], bbox_inches='tight', dpi=_dpi)
648
649     # "deviation" of the sigma plots
650     def plot_sigma_change(self, im_int, stepsize, resolution, limit, dose, annotation, _res, out):
651         if os.path.exists(self.path+self.name+'_'+_res+'_ano_dif1_norm_fit_90.pic'):
652             print self.path+self.name+'_'+_res+'_ano_dif1_norm_fit_90.pic'
653             print " DEB Normalized arrays are loaded."
654             with open(self.path+self.name+'_'+_res+'_ano_dif1_norm_fit_90.pic', "r") as n:
655                 no, normed_ano_dif1s, params, err=pickle.load(n)
656                 print no
657         else:
658           normed_ano_dif1s, params, err= self.analysing.analyse_Ano_Dif1(im_int, stepsize, resolution, 'I', _res, limit,
    "norm_fit")
659         if len(params)>1:
660             sigma, x_0=zip(*params)
661             sig_err, x_0_err=zip(*err)
662         change_sigma=[]
663         for i in xrange(0, len(sigma)-1):
664             change_sigma.append(sigma[i+1]-sigma[i])
665         x=numpy.arange(1, len(sigma))
666         x=[i+0.5 for i in x]
```

```
667         fig = pylab.figure(figsize=(10,6), dpi = 100)
668         ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])
669         ax.plot(x, change_sigma)
670         pylab.show()
671 
672     # plot the anomalous differences Ano_Dif2 (pseudosymmetry / subsequent) against the image number of the Bijvoet positive
    reflection
673     def plot_Ano_Dif2(self):
674         Ano = self.ano.Ano_Dif2(self.path)
675         lkeys = Ano.keys()
676         for el in lkeys:
677             x = []
678             y = []
679             if Ano[el][0][0] > 0:
680                 for l in Ano[el]:
681                     x.append(l[2])
682                     y.append(l[0])
683             #    if len(x)>len(Ano[el]):
684             #        break
685                 print len(x)
686                 pylab.plot(x, y, 'b^-')
687                 pylab.xlabel(r'image number', fontsize=14)
688                 pylab.ylabel(r'Anomalous Difference', fontsize=14)
689                 pylab.show()
690 
691     # plot the anomalous difference Ano_Dif3 (pairwise for certain resolution/ difference in image interval) against the image
    number difference
692     def plot_Ano_Dif3(self):
693         Ano = self.ano.Ano_Dif3(self.path, self.interval)
694         lkeys = Ano.keys()
695         x = []
696         y = []
697         for el in lkeys:
698             for i in xrange(len(Ano[el])):
699                 x.append(Ano[el][i][0])
700                 y.append(abs(Ano[el][i][1]))
701         pylab.plot(x, y, 'b^')
702         a = str(self.interval)
703         pylab.title('Anomalous Differences within an interval of ' + a + ' images')
704         pylab.xlabel(r'image number difference', fontsize=14)
705         pylab.ylabel(r'Anomalous Difference', fontsize=14)
706         pylab.show()
707 
708     # plot subsequent anomalous differences normed to Delta F_n as a histogram
709     def analyse_Ano_Dif4(self, name, r_m, im_int):
710         if os.path.exists(self.path+"ano_dif4.pic"):
711             with open(self.path+"ano_dif4.pic", "r") as g:
712                 ano4=pickle.load(g)
713         else:
714             ano4 = self.ano.Ano_Dif4(self.path, name, r_m, im_int)
715         print ano4[(2,1,3)]
716         anos=[]
717         for key in ano4:
718             for el in ano4[key]:
719                 anos.append(el[0])
720         a=numpy.asarray(anos)
721         bins = numpy.linspace(-10, 10, num=101)
722         _y, _x = numpy.histogram(a, bins=bins)
723         _x = bins[:-1] + (bins[1] - bins[0]) / 2
724         fig1 = pylab.figure(figsize=(10,6), dpi = 80)
725         fig1a=fig1.add_axes([0.1, 0.1, 0.75, 0.75])
726         fig1a.bar(_x, _y, width = (_x[1]- _x[0]), color='g', align = 'center')
727         fig1a.set_xlabel(r"$\Delta F$ / [a.u.]", **axis_font)
728         fig1a.set_ylabel("frequency", **axis_font)
729         pylab.title("Histogram for all possible anomalous differences calculated from 360$^\circ$unscaled data of
    thaumatin",**title_font)
730         pylab.savefig("/Users/storm/Documents/Experiments_2015/Anomalous_Diffraction/20150421/results/tha6/1.75_res_cut/
    thau6_360_unscaled/thau6_360_unscaled.png", bbox_inches='tight', dpi=300)
731         pylab.show()
732 
733     # plot subsequent anomalous differences normed to <Delta F>
734     def plot_AnoDif5(self, seq_int, im_int, resolution, change, out):
735         if os.path.exists(self.path+'./ano_dif5_'+str(change[1])+'.pic'):
736             with open(self.path+'./ano_dif5_'+str(change[1])+'.pic', "r") as f:
737                 ano_dif, ano_all, ano_n_all, z=pickle.load(f)[0:4]
738         else:
739              ano_dif, ano_all, ano_n_all, z=self.ano.AnoDif5b(self, seq_int, im_int, resolution, change)[0:4]
740         pylab.clf()
741         ano_allarr = numpy.asarray(ano_n_all)
742         col=pylab.get_cmap("Spectral")
743         mp = pylab.cm.ScalarMappable(cmap='Spectral')
744         mp.set_array(ano_allarr)
745         fig, ax = plt.subplots()
746         print " DEB %s data points have been found" %str(len(ano_dif))
747         plot=ax.scatter(z, ano_dif, c=col((ano_allarr-ano_allarr.min())/(ano_allarr.max()-ano_allarr.min())), marker="^", lw=0.1)
748         fig.colorbar(mp)
749         pylab.xlim(min(z)-50, max(z)+50)
750         pylab.ylim(min(ano_dif)-0.1, max(ano_dif)+0.6)
```

plotting

```
751         pylab.annotate('%s data points' %len(ano_dif), xy=(0.05, 0.95), xycoords='axes fraction', horizontalalignment='left',
    verticalalignment='top', **annotate_font)
752         pylab.annotate(r'$\mu (\frac{|\Delta F_n|- |\Delta F_{n+1}|}{<\Delta F>})$ = %s' % round(numpy.mean(ano_dif), ndigits=4),
    xy=(0.05, 0.9), xycoords='axes fraction', horizontalalignment='left', verticalalignment='top', **annotate_font)
753         pylab.annotate(r'$\sigma (\frac{|\Delta F_n|-|\Delta F_{n+1}|}{<\Delta F>})$ = %s' % round(numpy.std(ano_dif), ndigits=4),
    xy=(0.05, 0.85), xycoords='axes fraction', horizontalalignment='left', verticalalignment='top', **annotate_font)
754         pylab.annotate(r'$\mu (|\Delta F_n|$) = %s' % round(numpy.mean(ano_all), ndigits=4), xy=(0.9, 0.9), xycoords='axes
    fraction', horizontalalignment='right', verticalalignment='top', **annotate_font)
755         pylab.annotate(r'$\mu (|\Delta F_{n+1}|$) = %s' % round(numpy.mean(ano_n_all), ndigits=4), xy=(0.9, 0.85), xycoords='axes
    fraction', horizontalalignment='right', verticalalignment='top', **annotate_font)
756         pylab.xlabel(r'image number of $I_{n+1}^+$', **axis_font)
757         pylab.ylabel(r"$\frac{|\Delta F_n|-|\Delta F_{n+1}|}{<\Delta F>}$", **axis_font)
758         pylab.savefig(out)
759
```

helping_routines

```
from matplotlib import *
import matplotlib.mlab as mlab
from collections import Counter
from input_data import *
from profilehooks import timecall
from interval import interval
from anomalous_differences import *
from analysing import *
from plottin import *
from scipy import math
import scipy.misc
from scipy.stats import norm
from matplotlib import rc

class helping_routines():

    def __init__(self, case, path, name, r_m):
        self.case = case
        self.path = path
        self.name = name
        self.r_m= r_m

    # select reflections within a certain image interval from XDS_ASCII.HKL
    def rewrite_XDS_ASCII(self, im_int):
        fname = "XDS_ASCII.HKL"
        print " --- \n DEB make_HKL start. Reading from %s " % (fname)
        file=open(self.path+fname,'r')
        print im_int
        lines=file.readlines()
        print "lines are read from"+self.path+fname
        file.close()
        header=lines[0:47]
        data=lines[47:-1]
        end=lines[-1]
        line_counter=0
        new=self.path+self.name+"_"+str(im_int[1])
        if not os.path.exists(new):
            os.mkdir(new)
        os.chdir(new)
        print new
        new_file=open(new+"/XDS_ASCII.HKL",'w')
        for line in header:
            new_file.write(line)
        for line in data:
            z=float(line.split()[7])
            if z <= float(im_int[1]):
                line_counter=line_counter+1
                new_file.write(line)
            else: break
        new_file.write(end)
        new_file.close()
        print "%s reflections have been written to the new XDS_ASCII.HKL file" %str(line_counter)

    # compare the ano_dif1 based on common acentric keys based on which the signed average anomalous differences are calculated;
  the first ano_dif1 serves as reference
    def common_acentric(self, workList, im_int, resolution, _res):
        print workList[0]+'/ano_dif1_'+_res+'.pic'
        if os.path.exists(workList[0]+'ano_dif1_'+_res+'.pic'):
            with open(workList[0]+'ano_dif1_'+_res+'.pic', "r") as g:
                ano_dif1_ref=pickle.load(g)
        else:
            ano=anomalous_differences(self.case, self.path, self.name, self.r_m)
            ano_dif1_ref=self.ano.Ano_Dif1(im_int, resolution)
        os.system('cp '+workList[0]+'/ano_dif1.pic'+ workList[0]+'ano_dif1_common.pic')
        print "DEB The reference data set contains %s average signed anomalous differences" %str(len(ano_dif1_ref))
        for path in workList[1:]:
            print path+'ano_dif1_'+_res+'.pic'
            ano_dif1_common={}
            if os.path.exists(path+'ano_dif1_'+_res+'.pic'):
                with open(path+'ano_dif1_'+_res+'.pic', "r") as g:
                    ano_dif1=pickle.load(g)
            else:
                ano=anomalous_differences(self.case, self.path, self.name, self.r_m)
                ano_dif1=self.ano.Ano_Dif1(im_int, resolution)
            for key in ano_dif1_ref.keys():
                if not ano_dif1.has_key(key):
                    ano_dif1_common.update({key:ano_dif1[key]})
                    del ano_dif1_ref[key]
                else:
                    continue
        print "DEB There are %s common acentric reflections in all data sets" %str(len(ano_dif1_ref))
        for path in workList:
            ano_dif1_common={}
            with open(path+'/ano_dif1_'+_res+'.pic', "r") as g:
                ano_dif1=pickle.load(g)
            for key in ano_dif1_ref.keys():
                ano_dif1_common.update({key:ano_dif1[key]})
            print "DEB This data set has %s acentric reflections in common with the reference data set." %str(len(ano_dif1_common))
```

```
88            with open(path+'ano_dif1_Isig_common_'+_res+'.pic', "w") as d:
89                pickle.dump(ano_dif1_common, d)
90
91    # write a program to  include all results from a data set to one pdf via LaTeX
92    def results_to_pdf(self, output_folder, im_path, system, date, meas_param, rounds, cell_a, cell_c, Bi_ra, mosaicity, red,
  completeness, ano_completeness, SHELX_res, SHELX_CC, tot_res, arp_res, R_w, R_f, R_p):
93        if not os.path.exists(output_folder):
94            os.mkdir(output_folder)
95        else:
96            with open(output_folder+self.name+".tex", "w") as f:
97                f.write('\documentclass[headsepline,12pt,a4paper,bibliography=totoc, twoside]{article}\n')
98                f.write('\usepackage{booktabs}\n')
99                f.write('\usepackage{a4wide}\n')
100               f.write('\usepackage{graphicx}\n')
101               f.write('\usepackage{subfig}\n')
102               f.write('\usepackage{palatino}\n')
103               f.write('\usepackage[utf8]{inputenc}\n')
104               f.write('\usepackage[english]{babel}\n')
105               f.write('\usepackage{amsmath, amstext, amssymb, amsthm}\n')
106               f.write('\usepackage{color}\n')
107               f.write('\usepackage{rotating}\n')
108               f.write('\usepackage{setspace}\n')
109               f.write('\usepackage{lmodern}\n')
110               f.write('\usepackage{multirow}\n')
111               f.write('\usepackage[T1]{fontenc}\n')
112               f.write("\usepackage{textcomp} \n")
113               f.write("\usepackage{transparent}\n")
114               f.write("\n")
115               f.write("\def\coot{{\sc coot}} \n")
116               f.write("\def\\refmac{{\sc refmac}} \n")
117               f.write("\def\\raddose{{\sc raddose3d}} \n")
118               f.write("\def\\xds{{\sc xds}} \n")
119               f.write("\def\shelxd{{\sc SHELXD}} \n")
120               f.write("\def\sitcom{{\sc sitcom}} ")
121               # begin document with the title
122               f.write('\\begin{document}\n')
123               f.write('\\begin{center}\n')
124               f.write('\\begin{huge}{'+str(system.split()[0])+'}\\ \n')
125               f.write('\end{huge}\n')
126               f.write('\end{center}\n')
127               f.write("\n")
128               # plot an image of the crystal
129               f.write('\\begin{figure}[h!]\n')
130               f.write('\centering\n')
131               f.write('\includegraphics[scale=0.5]{'+im_path+'}\n')
132               f.write('\caption{'+system+' with $150~\mu$m aperture (blue circle).}\n')
133               f.write('\label{cc}\n')
134               f.write('\end{figure}\n\n')
135               f.write("\n")
136               # write table with measurement and refinement parameters
137               f.write("\\begin{table}[h!!!] \n")
138               f.write("\centering \n")
139               f.write("\\begin{tabular}{l l} \n")
140               f.write("\\toprule[0.1em] \n")
141               f.write("\\textbf{data collection} \\\ \midrule[0.05em] \n")
142               f.write("energy [keV]  & "+str(meas_param[0])+" \\\ \n")
143               f.write("flux [ph/s]  &  "+str(meas_param[3])+" \\\ \n")
144               f.write("aperture [$\mu m$] &"+str(meas_param[2])+ " \\\ \n")
145               f.write("exposure [s/image] & "+str(meas_param[4])+" \\\ \n")
146               f.write("oscillation range [$^\circ$] &"+str(meas_param[5])+" \\\ \n")
147               f.write("temperature [K] & 100 \\\ \n")
148               f.write("crystal size [$\mu m^3$] & "+str(meas_param[1]) +"\\\ \n")
149               f.write("space group & $"+str(meas_param[6])+"$ \\\ \n")
150               f.write("unit cell parameters [$\\r{A}$,$^\circ$] & a=b="+str(cell_a)+", c="+str(cell_c)+", $\\alpha$=$\\beta$=$
  \gamma$= 90.0 \\\ \n")
151               f.write("max. resolution [$\\r{A}$] &"+ str(meas_param[7])+" \\\ \n")
152               f.write("Bijvoet ratio [\%] &"+ str(Bi_ra)+ "\\\ \n")
153               f.write("solvent fraction &"+  str(meas_param[9]) +" \\\ \n")
154               f.write("solvent heavy atom conc. [M/l] & "+ str(meas_param[-2])+" \\\ \n")
155               f.write("average diffraction weighted dose [MGy/360$^\circ$] &"+ str(meas_param[-1])+ "\\\ \midrule[0.05em] \n")
156               f.write("\\textbf{refinement}  \\\ \midrule[0.05em] \n")
157               f.write("data [$^\circ$] & "+str(rounds)+"x360 \\\ \n")
158               f.write("mosaicity [$^\circ$] & "+str(mosaicity)+ "\\\ \n")
159               f.write("average anomalous multiplicity & "+str(red)+"\\\ \n")
160               f.write("overall anomalous completeness & "+str(ano_completeness)+"\\\ \n")
161               f.write("overall completeness [\%] & "+str(completeness)+" \\\ \n")
162               f.write("residues built (SHELXE) & "+str(SHELX_res) +" ("+str(tot_res)+") \\\ \n")
163               f.write("$CC_{partial}$ (SHELXE) [\%] &"+str(SHELX_CC)+ "\\\ \n")
164               f.write("residues built (arpwarp) &"+str(arp_res)+" ("+str(tot_res)+") \\\ \n")
165               f.write("$R_{work}$ [\%] & "+ str(R_w) +"\\\ \n")
166               f.write("$R_{free}$ [\%]  &"+ str(R_f)+ " \\\ \n")
167               f.write("$R_{pim}$ (all $I^+$ \& $I^-$) [\%] & "+ str(R_p) +"\\\ \n")
168               f.write("\\bottomrule[0.1em] \n")
169               f.write("\end{tabular} \n")
170               f.write("\caption{Parameters for thaumatin data collected with a top-hat profile at P14. $R_{work}$ and $R_{free}$
  have been obtained by refining 8 rounds with \coot~and \\refmac.} \n")
171               f.write("\label{tab} \n")
172               f.write("\end{table} \n")
```

```
173                 # XDS results (taking pdf output from plotting_results; other option: svg output + conversion mit inkscape,
    currently not working)
174                 f.write("\\begin{figure}[h!!!]\n")
175                 f.write("\centering\n")
176                 f.write("\subfloat[]{\includegraphics[scale=0.315]{"+output_folder+self.name+"_I.pdf}}  \hfill \n")
177                 f.write("\subfloat[]{\includegraphics[scale=0.315]{"+output_folder+self.name+"_C.pdf}} \\\ \n")
178                 f.write("\subfloat[]{\includegraphics[scale=0.315]{"+output_folder+self.name+"_an.pdf}} \hfill \n")
179                 f.write("\subfloat[]{\includegraphics[scale=0.315]{"+output_folder+self.name+"_R.pdf}} \n")
180                 f.write("\caption{Plots of a) the I/$\sigma$ values in the highest resolution shell, b) the $CC_{1/2}$ values in
    the highest resolution shell, c) the total anomalous signal and d) the total \
181                         $R_{meas}$ obtained by  processing subsequently collected $360^\circ$ of data with \\xds.} \n")
182                 f.write("\label{XDS}\n")
183                 f.write("\end{figure}\n")
184                 f.write("\n")
185                 # here: SHELX and sitcom results for accumulated and wedgewise processed data; wait until SHELXD version problem is
    solved
186                 f.write("\\begin{figure}[h!!!]\n")
187                 f.write("\centering\n")
188                 f.write("\subfloat[]{\includegraphics[scale=0.315]{"+output_folder+self.name+"_CF.pdf}}  \hfill \n")
189                 f.write("\subfloat[]{\includegraphics[scale=0.315]{"+output_folder+self.name+"_acc_CF.pdf}} \\\ \n")
190                 f.write("\subfloat[]{\includegraphics[scale=0.315]{"+output_folder+self.name+"_si.pdf}} \hfill \n")
191                 f.write("\subfloat[]{\includegraphics[scale=0.315]{"+output_folder+self.name+"_acc_si.pdf}} \\\ \n")
192                 f.write("\subfloat[]{\includegraphics[scale=0.29]{"+output_folder+self.name+"_ano_peak.pdf}} \hfill \n")
193                 f.write("\subfloat[]{\includegraphics[scale=0.29]{"+output_folder+self.name+"_acc_ano_peak.pdf}} \n")
194                 f.write("\caption{Plots of a) the CFOM obtained by \shelxd~via processing the data in 360$^\circ$ wedges, b) the
    CFOM obtained by \shelxd~via processing the accumulated data, c) the number of  \
195                         correct heavy atom sides and the corresponding rmsd to the reference structure obtained by \sitcom~via
    processing the data in wedges, d) the number of correct heavy atom sides and the  \
196                         corresponding rmsd to the reference structure obtained by \sitcom~via processing the accumulated unscaled
    data. \shelxd~is run with 10000 trials and a resolution cut-off of $2.8~\\r{A}$. Fig. \
197                         \\ref{SHELX} e) shows the anomalous peak heights of the sulfur atoms calculated with anode for the data
    processed in wedges and f) the same for the accumulated data.} \n")
198                 f.write("\label{SHELX}\n")
199                 f.write("\end{figure}\n")
200                 f.write("\n")
201                 # ano_dif1 for wedgewise processed data based on intensities only
202                 f.write("\\begin{figure}[h] \n")
203                 f.write("\centering \n")
204                 f.write("\subfloat[]{\includegraphics[scale=0.315]{"+output_folder+self.name+"_all_hist+fit_360_.pdf}} \hfill \n")
205                 f.write("\subfloat[]{\includegraphics[scale=0.315]{"+output_folder+self.name+"_acc_all_hist+fit_1800_.pdf}} \\\ ")
206                 f.write("\subfloat[]{\includegraphics[scale=0.315]{"+output_folder+self.name+"_all_hist+fit_5400_.pdf}} \hfill \n")
207                 f.write("\subfloat[]{\includegraphics[scale=0.315]{"+output_folder+self.name+"_acc_all_hist+fit_5400_.pdf}} \n")
208                 f.write("\caption{Analysis of the average signed anomalous differences, taking only the common acentric unique
    reflections into account. Fig. \\ref{ano_dif} a) represents the histogram of the wedgewise \
209                          processed data based on the first $360^\circ$ of data, b) shows the histogram of $5$x$360^\circ$
    accumulated data, c) depicts the histogram of the last $360^\circ$ of data (15th turn) and d) shows\
210                         the histogram based on the average signed anomalous differences accumulated in $15$ turns. All histograms
    are calculated with the same $100$ bins.}\n")
211                 f.write("\label{ano_dif}\n")
212                 f.write("\end{figure}\n")
213                 f.write("\n")
214                 # ano_dif1 fits for accumulated data based on intensities only
215                 f.write("\\begin{figure}[h] \n")
216                 f.write("\centering \n")
217                 f.write("\subfloat[]{\includegraphics[scale=0.5]{"+output_folder+self.name+"_2d_.pdf}} \\\ \n")
218                 f.write("\subfloat[]{\includegraphics[scale=0.6]{"+output_folder+self.name+"_acc_3d_.pdf}}\n")
219                 f.write("\caption{The normalized histograms of the signed averaged anomalous differences have been fitted with
    normal distributions for a) the wedgewise processed data and b) for the accumulated data.}\n")
220                 f.write("\label{fits}\n")
221                 f.write("\end{figure}\n")
222                 f.write("\n")
223                 # sigma_fits
224                 f.write("\\begin{figure}[h] \n")
225                 f.write("\centering \n")
226                 f.write("\subfloat[]{\includegraphics[scale=0.315]{"+output_folder+self.name+"_sigma_all_.pdf}} \hfill \n")
227                 f.write("\subfloat[]{\includegraphics[scale=0.315]{"+output_folder+self.name+"_sigma_hres_.pdf}} \\\ \n")
228                 f.write("\subfloat[]{\includegraphics[scale=0.315]{"+output_folder+self.name+"_sigma_mres_.pdf}}\hfill \n")
229                 f.write("\subfloat[]{\includegraphics[scale=0.315]{"+output_folder+self.name+"_sigma_lres_.pdf}} \n")
230                 f.write("\caption{Analysis of the fitting parameter $\sigma$ for the normal distributions fitting the normalized
    histograms of the average signed anomalous differences based on intensities only, depending \
231                         on the resolution of the common acentric reflections, i.e. a) for all wedgewise (blue) and all accumulated
    (red) data; b) for data with a resolution of $1.7-2.0~\\r{A}$, c) for data with a \
232                         resolution of $2.0-2.8~\\r{A}$ and d) for data with a resolution of >$2.8~\\r{A}$. The numbers of average
    signed anomalous differences correspond to the ones used for the last data point in the plot.}\n")
233                 f.write("\label{sigma} \n")
234                 f.write("\end{figure} \n")
235                 f.write("\n")
236                 f.write("\end{document}\n")
```

main

```
import numpy, scipy
import matplotlib
import sympy
from input_data import *
from anomalous_differences import *
from plotting import *
from analysing import *
from helping_routines import *
from profilehooks import timecall
import matplotlib
import matplotlib.pyplot as plt
from matplotlib.font_manager import path
import matplotlib.backends.backend_pdf

#import pprofile


@timecall
def __main__():

# define path to XDS_ASCII.HKL file, space group and name of the data set

# ***************************thaumatin*****************************************
    case="P41212"
    space_group=92
    name="tha_1"
    path="/Volumes/New_Volume/PETRA_data/20150421/PROCESSED_DATA/tha1/"
    resolution=[1.9, 999]
    r_m=resolution[0]
# *******************************************************************************

#######class input_data#################
# class to read XDS_ASCII.HKL or the scaled version, calculate the resolution, sort in unique,
# systematic absent and centric reflections as in Bijvoets requieres the program "unique" from ccp4
    # initialize class
    run=input_data(case, path, name, r_m)
    #--method call--
    run.spacegroup()                    # depending on the spacegroup, symmetry equivalents (Bijvoet positive and negative reflections) are defined in symbolic variables; reference sfall (CCP4)
    run.extraxt()                       # extracting cell constants from XDS_ASCII.HKL header
    run.resolution()                    # calculates resolution based on hkl-indices and cell constants (possible for all space groups)
    run.make_HKL_dic()                  # extracting all reflections with a sigma >0, including systematic absences,  and sorting them in dictionaries
    run.unique()                        # run unique (CCP4) and store unique reflections in a dictionary
    run.Bijvoet_Pairs()                 # find all symmetry equivalents for each unique reflection, determine systematic absences and centric reflections dependent on the space group and store Bijvoets, centric and systematic abesent reflections in dictionaries
########################################

#######class anomalous_differences#####
# class to calculate anomalous differences in different ways; requires Bijvoets
    # parameter
    seq_int=[0, 5400]
    im_int=[1, 5400]
    change=[-1, 1]
    window=90
    res_shells=[1.75, 2.0, 2.2, 2.4, 2.6, 3.0, 3.5, 4.0, 5.0, 6.0, 8.0, 999]
    reference=self.path+"tha1_phenix.hkl
    out=path
    # initialize class
    ano=anomalous_differences(case, _path, name, r_m)
    #--method calls--
    ano.Ano_Dif1(im_int, resolution, _res)                          # build average of Bijvoet positives and negatives to build signed anomalous differences, considering only positive intensities
    ano.Ano_Dif2()                                                  # calculate the anomalous difference between the first Bijvoet positive and all, but at least three Bijvoet negative
    ano.Ano_Dif3(im_int)                                            # calculating anomalous differences pairwise, while the difference between the corresponding image numbers is defined by im_int
    ano.Ano_Dif4()                                                  # calculate subsequent anomalous differences, i.e. (Delta F_n - Delta F_n+1)/Delta F_n
    ano.Ano_Dif5(seq_int, im_int, resolution, change)               # calculate Delta F_n - Delta F_n+1 /< Delta F > for a certain image interval with values in the interval change
    ano.CC_ano_subsequent(path, im_int, resolution)                 # calculate subsequent Bijvoet mates with given resolution in a window defined by seq_int with an overall image interval im_int with values within the interval change
    ano.CC_ano_subsequent_window(path, im_int, window, resolution)  # calculate the CC of dF_n, dF_n+1 within a certain image interval and a certain sliding window for a given resolution
    ano.CC_subsequent_ano_res(path, im_int, res_shells)             # calculate the CC of dF_n, dF_n+1 within a certain image interval in different resolution shells
    ano.DF_from_ideal_data(self, reference, out)                    # write anomalous difference from ideal data generated with phenix
########################################

#######class analysing#################
    # parameter
    stepsize=360
    im_int=[1, 5400]
    key=(17,10,39)
    _type="norm_fit"                        # either "norm_fit" or "hist"
```

main

```
76      limit=3                                 # limits for the histogram /the max. average signed anomalous difference in the
    histogram
77      # initialize class
78      analyse=analysing(case, path, name, r_m)
79      #--method calls--
80      analyse.analyse_Ano_Dif1(im_int, stepsize, resolution, mode, _res, limit, _type)  # depending on the output, a) a histogram
    with a gaussian fit (_type='histogram') and b) a normalized histogram fitted with a normal distribution
81      analyse.compare_fit_with_histogram(_res, limit)                                   # calculates the mean square difference
    between the histograms and the fit
82      analyse._Bijvoets(name, im_int, resolution)                                       # statistics of the Bijvoets regarding
    multiplicity, minimum and maxiumum intensities
83      analyse.reflection_analysis(self, im_int, stepsize, key)                          # change of average signed anomalous
    differences for one reflection for wedges and accumulated data
84  ######################################
85
86  #######class plotting###################
87      # parameter
88      res_shells=[1.75, 2.0, 2.2, 2.4, 2.6, 3.0, 3.5, 4.0, 5.0, 6.0, 8.0, 999]
89      args=[path+"thau6_1800/_thau6_1800/ano_dif1_"+_res+".pic", path+"thau6_1800/_thau6_1800/ano_dif1_"+_res+".pic"]
90      images-
91      fr_dif=
92      _type='hist+fit'            # alternative: hist
93      kind='sigma'                # alternative: x_0, 2d, 3d, max, sigma
94      out=["/Users/storm/Documents/Experiments_2015/Anomalous_Diffraction/20150421/results/tha1/", ".pdf"]
95      dose=0.65
96      change=[-1, 1]
97      _res='hres'                 # could also be mres, lres, all
98      annotation=r'1.9-2.0~\r{A}'
99      # initialize class
100     plot=plotting(case, path, name, r_m)
101     # method calls
102     plot.plot_I_p_vs_I_n(_res)                                                                  # plot the average
    intensities of the Bijvoet positive and Bijvoet negative reflections
103     plot.scatterplot(_res, args)                                                                # scatterplot for two
    arrays, e.g. the average signed anomalous difference of two wedges
104     plot.plot_isa_vs_res(res_shells)                                                            # plot I/sigma versus
    resolution
105     plot.CC_vs_res(res_shells, args)                                                            # plot the correlation
    coefficient of the anomalous differences versus resolution
106     plot.plot_ano_vs_frameInt(images, fr_dif)                                                   # plot the average
    anomalous signal within a certain image interval
107     plot.plot_ano_dif1_vs_res(_res)                                                             # plot <Delta F> versus
    resolution
108     plot.plot_AnoDif1_hist(im_int, stepsize, resolution, mode, _res, limit, _type, out)        # plot histogram of
    <Delta F>, optionally with a fit                                                                    # either a
    histogram only (_type=hist) or a histogram with a gaussian fit (_type=hist+fit)
109     plot.plot_AnoDif1_normed_fits(im_int, stepsize, resolution, limit, kind, dose, annotation, _res, out)   # plot normalized
    histograms, normal distributions, sigma, x_0 or maxima of the distribution
110     plot.plot_sigma_fit(im_int, stepsize, resolution, limit, dose, annotation, _res, out)       # fit the sigma values
    of the normal distributions
111     plot.plot_sigma_change(im_int, stepsize, resolution, limit, dose, annotation, _res, out)    # plot the change of
    subsequent sigma values
112     plot.plot_Ano_Dif2()                                                                        # plot the anomalous
    differences Ano_Dif2 against the image number of the Bijvoet positive reflection
113     plot.plot_Ano_Dif3()                                                                        # plot the anomalous
    difference Ano_Dif3 against the image number difference
114     plot.plot_Ano_Dif4(name, r_m, im_int)                                                       # plot subsequent
    anomalous differences normed to Delta F_n as a histogram
115     plot.plot_Ano_Dif5(seq_int, im_int, resolution, change, out)                                # plot subsequent
    anomalous differences normed to <Delta F>
116 ######################################
117
118 ########class helping_routines #######
119     # parameter
120     output_folder='/Users/storm/Documents/Experiments_2015/Anomalous_Diffraction/20151017/tha_1/results/'
121     system='Thaumatin 2'
122     date=20151017
123     meas_param=[8.01, '92 x 121 x 23', 150, '$2.8\cdot 10^{11} / 1.4\cdot 10^{12} $', 0.05, 1, 'P4_12_12', 1.7, 0.54, '1.5 Na',
    0.18 / 0.88]
124     im_path='/Users/storm/Documents/Experiments_2015/Anomalous_Diffraction/20151017/thau_2/thaumatin_2.pdf'
125     rounds=4
126     Bi_ra=1.23
127     cell_a=57.86
128     cell_c=150.42
129     mosaicity=0.06
130     red=22.1
131     completeness=99.9
132     SHELX_res= 195
133     SHELX_CC=43.95
134     tot_res=207
135     arp_res=205
136     R_w=15.93
137     R_f=18.49
138     R_p=1.87
139     workList=[path+"/thau1_"+str(i*360)+"/" for i in xrange(1,16)]
140     # initialize class
141     hr=helping_routines(case, path, name, r_m)
142     # method calls
```

main

```
143     hr.rewrite_XDS_ASCII(im_int)                                # rewrites XDS_ASCII.HKL file with the reflections in the desired
    image interval
144     hr.common_acentric(workList, im_int, resolution, _res)      # finds common acentric reflections in dictionaries
145     hr.results_to_pdf(output_folder, im_path, system, date, meas_param, rounds, cell_a, cell_c, Bi_ra, mosaicity, red,
    completeness, SHELX_res, SHELX_CC, tot_res, arp_res, R_w, R_f, R_p)
146                                                                 # writes results and plots to file
147
148 __main__()
149
```

# Bibliography


[1] http://intranet.embl-hamburg.de/facilities/external_facilities/mx/p13/index.html, 01/2016.

[2] http://intranet.embl-hamburg.de/facilities/external_facilities/mx/p14/index.html, 01/2016.

[3] http://strucbio.biologie.uni-konstanz.de/xdswiki/index.php/isa, 02/2016.

[4] P. D. Adams, P. V. Afonine, G. Bunkóczi, V. B. Chen, I. W. Davis, N. Echols, J. J. Headd, L.-W. Hung, G. J. Kapral, R. W. Grosse-Kunstleve, et al. PHENIX: a comprehensive Python-based system for macromolecular structure solution. *Acta Crystallographica Section D: Biological Crystallography*, 66(2):213–221, 2010.

[5] J. Als-Nielsen and D. McMorrow. *Elements of Modern X-ray Physics*. John Wiley & Sons, 2011.

[6] K. Balewski, W. Brefeld, W. Decking, H. Franz, R. Röhlsberger, and E. Weckert. *PETRA III: a Low Emittance Synchrotron Radiation Source: Technical Design Report*, volume 35. DESY, 2004.

[7] N. Ban, P. Nissen, J. Hansen, P. B. Moore, and T. A. Steitz. The complete atomic structure of the large ribosomal subunit at 2.4 Å resolution. *Science*, 289(5481):905–920, 2000.

[8] S. Banumathi, P. H. Zwart, U. A. Ramagopal, M. Dauter, and Z. Dauter. Structural effects of radiation damage and its potential for phasing. *Acta Crystallographica Section D: Biological Crystallography*, 60(6):1085–1093, 2004.

[9] C. Blake and D. Phillips. Effects of X-irradiation on single crystals of myoglobin. 1962.

[10] D. Blow. *Outline of Crystallography for Biologists*. Oxford University Press, 2002.

[11] D. Blow and F. Crick. The treatment of errors in the isomorphous replacement method. *Acta Crystallographica*, 12(10):794–802, 1959.

[12] G. Bourenkov. Private communication.

[13] G. P. Bourenkov and A. N. Popov. Optimization of data collection taking radiation damage into account. *Acta Crystallographica Section D: Biological Crystallography*, 66(4):409–419, 2010.

[14] W. H. Bragg and W. L. Bragg. The reflection of x-rays by crystals. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 88(605):428–438, 1913.

[15] A. T. Brünger. Free R value: a novel statistical quantity for assessing the accuracy of crystal structures. *Nature*, 355:472–475, 1992.

[16] G. Bunkóczi, A. J. McCoy, N. Echols, R. W. Grosse-Kunstleve, P. D. Adams, J. M. Holton, R. J. Read, and T. C. Terwilliger. Macromolecular X-ray structure determination using weak, single-wavelength anomalous data. *Nature Methods*, 2014.

[17] M. Cianci, M. R. Groves, D. Barford, and T. Schneider. Data collection with a tailored X-ray beam size at 2.69 Å wavelength (4.6 keV): sulfur SAD phasing of Cdc23$^{Nterm}$. *Acta Crystallographica Section D Structural Biology*, 72(3) : 403 – 412, 2016.

[18] C. P. Collaborative et al. The CCP4 suite: programs for protein crystallography. *Acta Crystallographica Section D, Biological Crystallography*, 50(Pt 5):760, 1994.

[19] K. Cowtan. Recent developments in classical density modification. *Acta Crystallographica Section D: Biological Crystallography*, 66(4):470–478, 2010.

[20] F. Dall'Antonia and T. Schneider. SITCOM: a program for comparing sites in macromolecular substructures. *Journal of Applied Crystallography*, 39(4):618–619, 2006.

[21] Z. Dauter and D. A. Adamiak. Anomalous signal of phosphorus used for phasing DNA oligomer: importance of data redundancy. *Acta Crystallographica Section D: Biological Crystallography*, 57(7):990–995, 2001.

[22] Z. Dauter, M. Dauter, E. de La Fortelle, G. Bricogne, and G. M. Sheldrick. Can anomalous signal of sulfur become a tool for solving protein crystal structures? *Journal of Molecular Biology*, 289(1):83–92, 1999.

[23] Z. Dauter, M. Dauter, and E. Dodson. Jolly SAD. *Acta Crystallographica Section D: Biological Crystallography*, 58(3):494–506, 2002.

[24] P. Debye. Zerstreuung von Röntgenstrahlen. *Annalen der Physik*, 351(6):809–823, 1915.

[25] K. Diederichs. Some aspects of quantitative analysis and correction of radiation damage. *Acta Crystallographica Section D: Biological Crystallography*, 62(1):96–101, 2006.

[26] K. Diederichs. Quantifying instrument errors in macromolecular X-ray data sets. *Acta Crystallographica Section D: Biological Crystallography*, 66(6):733–740, 2010.

[27] K. Diederichs and P. A. Karplus. Better models by discarding data? *Acta Crystallographica Section D: Biological Crystallography*, 69(7):1215–1222, 2013.

[28] K. Djinović Carugo, J. R. Helliwell, H. Stuhrmann, and M. S. Weiss. Softer and soft X-rays in macromolecular crystallography. *Journal of Synchrotron Radiation*, 12(4):410–419, 2005.

[29] E. Dodson. Is it jolly SAD? *Acta Crystallographica Section D: Biological Crystallography*, 59(11):1958–1965, 2003.

[30] P. Emsley, B. Lohkamp, W. G. Scott, and K. Cowtan. Features and development of coot. *Acta Crystallographica Section D: Biological Crystallography*, 66(4):486–501, 2010.

[31] G. Evans. *The Method of Multiple Wavelength Anomalous Diffraction Using Synchrotron Radiation at Optimal X-ray Energies: Application to Protein Crystallography*. PhD thesis, 1994.

[32] P. Evans. Scaling and assessment of data quality. *Acta Crystallographica Section D: Biological Crystallography*, 62(1):72–82, 2006.

[33] A. Faust, S. Panjikar, U. Mueller, V. Parthasarathy, A. Schmidt, V. S. Lamzin, and M. S. Weiss. A tutorial for learning and teaching macromolecular crystallography. *Journal of Applied Crystallography*, 41(6):1161–1172, 2008.

[34] A. D. Finke, E. Panepucci, C. Vonrhein, M. Wang, G. Bricogne, and V. Oliéric. Advanced crystallographic data collection protocols for experimental phasing. *Nucleic Acid Crystallography: Methods and Protocols*, pages 175–191, 2016.

[35] S. French and K. Wilson. On the treatment of negative intensity observations. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 34(4):517–525, 1978.

[36] Z.-Q. Fu, J. P. Rose, and B.-C. Wang. Monitoring the anomalous scattering signal and noise levels in X-ray diffraction of crystals. *Acta Crystallographica Section D: Biological Crystallography*, 60(3):499–506, 2004.

[37] E. F. Garman. Radiation damage in macromolecular crystallography: what is it and why should we care? *Acta Crystallographica Section D: Biological Crystallography*, 66(4):339–351, 2010.

[38] E. F. Garman and R. L. Owen. Cryocooling and radiation damage in macromolecular crystallography. *Acta Crystallographica Section D: Biological Crystallography*, 62(1):32–47, 2006.

[39] E. F. Garman and T. R. Schneider. Macromolecular cryocrystallography. *Journal of Applied Crystallography*, 30(3):211–237, 1997.

[40] E. F. Garman and M. Weik. Radiation damage to biological macromolecules: some answers and more questions. *Journal of Synchrotron Radiation*, 20(1):1–6, 2012.

[41] E. F. Garman and M. Weik. Radiation damage to macromolecules: kill or cure? *Journal of Synchrotron Radiation*, 22(2):195–200, 2015.

[42] C. Gati, G. Bourenkov, M. Klinge, D. Rehders, F. Stellato, D. Oberthür, O. Yefanov, B. P. Sommer, S. Mogk, M. Duszenko, et al. Serial crystallography on in vivo grown microcrystals using synchrotron radiation. *IUCrJ*, 1(2):87–94, 2014.

[43] C. Giacovazzo, H. Monaco, G. Artioli, D. Viterbo, M. Milanesio, G. Gilli, P. Gilli, G. Zanotti, and G. Ferraris. *Fundamentals of Crystallography (International Union of Crystallography Monographs on Crystallography)*. Oxford University Press, New York, 2011.

[44] D. Harker. The determination of the phases of the structure factors of non-centrosymmetric crystals by the method of double isomorphous replacement. *Acta Crystallographica*, 9(1):1–9, 1956.

[45] J. R. Helliwell and E. P. Mitchell. Synchrotron radiation macromolecular crystallography: science and spin-offs. *IUCrJ*, 2(2):0–0, 2015.

[46] W. Hendrickson and E. Lattman. Representation of phase probability distributions for simplified combination of independent phase information. *Acta Crystallographica Section B: Structural Crystallography and Crystal Chemistry*, 26(2):136–143, 1970.

[47] W. A. Hendrickson. Determination of macromolecular structures from anomalous diffraction of synchrotron radiation. *Science*, 254(5028):51–58, 1991.

[48] W. A. Hendrickson. Synchrotron crystallography. *Trends in biochemical sciences*, 25(12):637–643, 2000.

[49] W. A. Hendrickson. Anomalous diffraction in crystallographic phase evaluation. *Quarterly Reviews of Biophysics*, 47(01):49–93, 2014.

[50] W. A. Hendrickson, J. R. Horton, and D. M. LeMaster. Selenomethionyl proteins produced for analysis by multiwavelength anomalous diffraction (MAD): a vehicle for direct determination of three-dimensional structure. *The EMBO journal*, 9(5):1665, 1990.

[51] W. A. Hendrickson and C. M. Ogata. Phase determination from multiwavelength anomalous diffraction measurements. *Methods in Enzymology*, 276:494–523, 1997.

[52] W. A. Hendrickson, J. L. Smith, and S. Sheriff. Direct phase determination based on anomalous scattering. *Methods in Enzymology*, 115:41–55, 1985.

[53] W. A. Hendrickson and M. M. Teeter. Structure of the hydrophobic protein crambin determined directly from the anomalous scattering of sulphur. 1981.

[54] F. Hirshfeld and D. Rabinovich. Treating weak reflexions in least-squares calculations. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 29(5):510–513, 1973.

[55] A. Hodel, S.-H. Kim, and A. T. Brünger. Model bias in macromolecular crystal structures. *Acta Crystallographica Section A: Foundations of Crystallography*, 48(6):851–858, 1992.

[56] J. M. Holton. A beginner's guide to radiation damage. *Journal of Synchrotron Radiation*, 16(2):133–142, 2009.

[57] W. Kabsch. Integration, scaling, space-group assignment and post-refinement. *Acta Crystallographica Section D: Biological Crystallography*, 66(2):133–144, 2010.

[58] W. Kabsch. XDS. *Acta Crystallographica Section D: Biological Crystallography*, 66(2):125–132, 2010.

[59] J. Karle. Partial structural information combined with the tangent formula for noncentrosymmetric crystals. *Acta Crystallographica Section B: Structural Crystallography and Crystal Chemistry*, 24(2):182–186, 1968.

[60] J. Karle. Some developments in anomalous dispersion for the structural investigation of macromolecular systems in biology. *International Journal of Quantum Chemistry*, 18(S7):357–367, 1980.

[61] P. A. Karplus and K. Diederichs. Linking crystallographic model and data quality. *Science*, 336(6084):1030–1033, 2012.

[62] P. A. Karplus and K. Diederichs. Assessing and maximizing data quality in macromolecular crystallography. *Current Opinion in Structural Biology*, 34:60–68, 2015.

[63] J. C. Kendrew, G. Bodo, H. M. Dintzis, R. Parrish, H. Wyckoff, and D. C. Phillips. A three-dimensional model of the myoglobin molecule obtained by x-ray analysis. *Nature*, 181(4610):662–666, 1958.

[64] J. Kmetko, N. S. Husseini, M. Naides, Y. Kalinin, and R. E. Thorne. Quantifying X-ray radiation damage in protein crystals at cryogenic temperatures. *Acta Crystallographica Section D: Biological Crystallography*, 62(9):1030–1038, 2006.

[65] V. S. Lamzin, A. Perrakis, and K. S. Wilson. ARP/WARP– automated model building and refinement. *International Tables of Crystallography*, F:525–528, 2012.

[66] H.-K. Leiros, S. a. M. McSweeney, and A. O. Smalås. Atomic resolution structures of trypsin provide insight into structural radiation damage. *Acta Crystallographica Section D: Biological Crystallography*, 57(4):488–497, 2001.

[67] D. Liebschner, G. Rosenbaum, M. Dauter, and Z. Dauter. Radiation decay of thaumatin crystals at three X-ray energies. *Acta Crystallographica Section D: Biological Crystallography*, 71(4):772–778, 2015.

[68] Q. Liu, Y. Guo, Y. Chang, Z. Cai, Z. Assur, F. Mancia, M. I. Greene, and W. A. Hendrickson. Multi-crystal native SAD analysis at 6 keV. *Acta Crystallographica Section D: Biological Crystallography*, 70(10):2544–2557, 2014.

[69] Q. Liu and W. A. Hendrickson. Crystallographic phasing from weak anomalous signals. *Current Opinion in Structural Biology*, 34:99–107, 2015.

[70] K. Moffat and Z. Ren. Synchrotron radiation applications to macromolecular crystallography. *Current Opinion in Structural Biology*, 7(5):689–696, 1997.

[71] M. Mueller, M. Wang, and C. Schulze-Briese. Optimal fine $\varphi$-slicing for single-photon-counting pixel detectors. *Acta Crystallographica Section D: Biological Crystallography*, 68(1):42–56, 2012.

[72] J. Murray and E. Garman. Investigation of possible free-radical scavengers and metrics for radiation damage in protein cryocrystallography. *Journal of Synchrotron Radiation*, 9(6):347–354, 2002.

[73] M. H. Nanao, G. M. Sheldrick, and R. B. Ravelli. Improving radiation-damage substructures for RIP. *Acta Crystallographica Section D: Biological Crystallography*, 61(9):1227–1237, 2005.

[74] C. Nave, G. Sutton, G. Evans, R. Owen, C. Rau, I. Robinson, and D. Stuart. Imperfection and radiation damage in protein crystals studied with coherent radiation. *Journal of Synchrotron Radiation*, 23(1), 2016.

[75] V. Olieric, T. Weinert, A. D. Finke, C. Anders, D. Li, N. Olieric, C. N. Borca, M. O. Steinmetz, M. Caffrey, M. Jinek, et al. Data-collection strategy for challenging native SAD phasing. *Acta Crystallographica Section D: Structural Biology*, 72(3):421–429, 2016.

[76] N. S. Pannu, G. N. Murshudov, E. J. Dodson, and R. J. Read. Incorporation of prior phase information strengthens maximum-likelihood structure refinement. *Acta Crystallographica Section D: Biological Crystallography*, 54(6):1285–1294, 1998.

[77] T. Pape and T. R. Schneider. HKL2MAP: a graphical user interface for macromolecular phasing with SHELX programs. *Journal of applied crystallography*, 37(5):843–844, 2004.

[78] T. Petrova, S. Ginell, A. Mitschler, Y. Kim, V. Y. Lunin, G. Joachimiak, A. Cousido-Siah, I. Hazemann, A. Podjarny, K. Lazarski, et al. X-ray-induced deterioration of disulfide bridges at atomic resolution. *Acta Crystallographica Section D: Biological Crystallography*, 66(10):1075–1091, 2010.

[79] J. Pflugrath. The finer things in X-ray diffraction data collection. *Acta Crystallographica Section D: Biological Crystallography*, 55(10):1718–1725, 1999.

[80] U. A. Ramagopal, M. Dauter, and Z. Dauter. Phasing on anomalous signal of sulfurs: what is the limit? *Acta Crystallographica Section D: Biological Crystallography*, 59(6):1020–1027, 2003.

[81] S. Ramaseshan and N. Venkatesan. Use of anomalous scattering without phase change in crystal structure analysis. *Current Science*, 26:352, 1957.

[82] R. B. Ravelli and E. F. Garman. Radiation damage in macromolecular cryocrystallography. *Current Opinion in Structural Biology*, 16(5):624–629, 2006.

[83] R. B. Ravelli and S. M. McSweeney. The 'fingerprint' that X-rays can leave on structures. *Structure*, 8(3):315–328, 2000.

[84] R. B. Ravelli, P. Theveneau, S. McSweeney, and M. Caffrey. Unit-cell volume change as a metric of radiation damage in crystals of macromolecules. *Journal of Synchrotron Radiation*, 9(6):355–360, 2002.

[85] M. G. Rossmann and D. M. Blow. The detection of sub-units within the crystallographic asymmetric unit. *Acta Crystallographica*, 15(1):24–31, 1962.

[86] B. Rupp. *Biomolecular Crystallography*. Garland Science New York, 2010.

[87] G. Scapin. Molecular replacement then and now. *Acta Crystallographica Section D: Biological Crystallography*, 69(11):2266–2275, 2013.

[88] T. R. Schneider and G. M. Sheldrick. Substructure solution with SHELXD. *Acta Crystallographica Section D: Biological Crystallography*, 58(10):1772–1779, 2002.

[89] G. M. Sheldrick. A short history of SHELX. *Acta Crystallographica Section A: Foundations of Crystallography*, 64(1):112–122, 2007.

[90] G. M. Sheldrick. Experimental phasing with SHELXC/D/E: combining chain tracing with density modification. *Acta Crystallographica Section D: Biological Crystallography*, 66(4):479–485, 2010.

[91] G. M. Sheldrick, C. J. Gilmore, H. A. Hauptman, C. M. Weeks, R. Miller, and I. Usón. Direct methods. *International Tables of Crystallography*, F:413–432, 2012.

[92] J. L. Smith. Determination of three-dimensional structure by multiwavelength anomalous diffraction. *Current Opinion in Structural Biology*, 1(6):1002–1011, 1991.

[93] S. Storm, M. Ogurreck, D. Laipple, C. Krywka, M. Burghammer, E. Di Cola, and M. Mueller. On radiation damage in FIB-prepared softwood samples measured by scanning X-ray diffraction. *Journal of Synchrotron Radiation*, 22(2):267–272, 2015.

[94] K. A. Sutton, P. J. Black, K. R. Mercer, E. F. Garman, R. L. Owen, E. H. Snell, and W. A. Bernhard. Insights into the mechanism of X-ray-induced disulfide-bond cleavage in lysozyme crystals based on EPR, optical absorption and X-ray diffraction studies. *Acta Crystallographica Section D: Biological Crystallography*, 69(12):2381–2394, 2013.

[95] C. Tanford and J. Reynolds. *Nature's Robots: a History of Proteins*. Oxford University Press, 2001.

[96] G. Taylor. The phase problem. *Acta Crystallographica Section D: Biological Crystallography*, 59(11):1881–1890, 2003.

[97] T. y. Teng and K. Moffat. Primary radiation damage of protein crystals by an intense synchrotron X-ray beam. *Journal of Synchrotron Radiation*, 7(5):313–317, 2000.

[98] T. C. Terwilliger. MAD phasing: Bayesian estimates of $F_A$. *Acta Crystallographica Section D: Biological Crystallography*, 50(1):11–16, 1994.

[99] A. Thorn and G. M. Sheldrick. ANODE: anomalous and heavy-atom density calculation. *Journal of Applied Crystallography*, 44(6):1285–1287, 2011.

[100] A. Thorn and G. M. Sheldrick. Extending molecular-replacement solutions with SHELXE. *Acta Crystallographica Section D: Biological Crystallography*, 69(11):2251–2256, 2013.

[101] T. Ursby and D. Bourgeois. Improved estimation of structure-factor difference amplitudes from poorly accurate data. *Acta Crystallographica Section A: Foundations of Crystallography*, 53(5):564–575, 1997.

[102] A. A. Vagin, R. A. Steiner, A. A. Lebedev, L. Potterton, S. McNicholas, F. Long, and G. N. Murshudov. REFMAC5dictionary: organization of prior chemical knowledge and guidelines for its use. *Acta Crystallographica Section D: Biological Crystallography*, 60(12):2184–2195, 2004.

[103] A. Wagner, R. Duman, K. Henderson, and V. Mykhaylyk. In-vacuum long-wavelength macromolecular crystallography. *Acta Crystallographica Section D: Structural Biology*, 72(3):430–439, 2016.

[104] B.-C. Wang. Resolution of phase ambiguity in macromolecular crystallography. *Methods in Enzymology*, 115:90–112, 1985.

[105] T. Weinert, V. Olieric, S. Waltersperger, E. Panepucci, L. Chen, H. Zhang, D. Zhou, J. Rose, A. Ebihara, S. Kuramitsu, et al. Fast native-SAD phasing for routine macromolecular structure determination. *Nature Methods*, 12(2):131–133, 2015.

[106] M. S. Weiss. Global indicators of X-ray data quality. *Journal of Applied Crystallography*, 34(2):130–135, 2001.

[107] A. Wilson. Largest likely values for the reliability index. *Acta Crystallographica*, 3(5):397–398, 1950.

[108] M. D. Winn, C. C. Ballard, K. D. Cowtan, E. J. Dodson, P. Emsley, P. R. Evans, R. M. Keegan, E. B. Krissinel, A. G. Leslie, A. McCoy, et al. Overview of the CCP4 suite and current developments. *Acta Crystallographica Section D: Biological Crystallography*, 67(4):235–242, 2011.

[109] M. D. Winn, G. N. Murshudov, and M. Z. Papiz. Macromolecular TLS refinement in REFMAC5at moderate resolutions. *Methods in Enzymology*, 374:300–321, 2003.

[110] A. Wlodawer, W. Minor, Z. Dauter, and M. Jaskolski. Protein crystallography for non-crystallographers, or how to get the best (but not more) from published macromolecular structures. *FEBS Journal*, 275(1):1–21, 2008.

[111] J. Xu, Z. Chen, A.-T. Le, and C. Lin. Self-imaging of molecules from diffraction spectra by laser-induced rescattering electrons. *Physical Review A*, 82(3):033403, 2010.

[112] C. Yang, J. Pflugrath, D. Courville, C. Stence, and J. D. Ferrara. Away from the edge: SAD phasing from the sulfur anomalous signal measured in-house with chromium radiation. *Acta Crystallographica Section D: Biological Crystallography*, 59(11):1943–1957, 2003.

[113] O. B. Zeldin, S. Brockhauser, J. Bremridge, J. M. Holton, and E. F. Garman. Predicting the X-ray lifetime of protein crystals. *Proceedings of the National Academy of Sciences*, 110(51):20551–20556, 2013.

[114] O. B. Zeldin, M. Gerstel, and E. F. Garman. Optimizing the spatial distribution of dose in X-ray macromolecular crystallography. *Journal of Synchrotron Radiation*, 20(1):49–57, 2012.

[115] O. B. Zeldin, M. Gerstel, and E. F. Garman. RADDOSE-3D: time-and space-resolved modelling of dose in macromolecular crystallography. *Applied Crystallography*, 46(4):1225–1230, 2013.

[116] P. Zwart and V. Lamzin. The influence of positional errors on the Debye effects. *Acta Crystallographica Section D: Biological Crystallography*, 60(2):220–226, 2004.

[117] P. H. Zwart, S. Banumathi, M. Dauter, and Z. Dauter. Radiation-damage-induced phasing with anomalous scattering: substructure solution and phasing. *Acta Crystallographica Section D: Biological Crystallography*, 60(11):1958–1963, 2004.

# Acknowledgements

First of all, my thanks go to Thomas R. Schneider for letting me dive into the fascinating world of protein crystallography in all its different aspects. Coming from solid state physics, it was a great opportunity to learn how to work in the wetlab, how to grow (or not grow) protein crystals, how to measure them at the beamline and also to improve my programming skills. I am especially grateful for the freedom given to pursue old and new projects, the chance to go to many conferences and workshops and for always constructive discussions.
For patient discussions regarding spacegroups, symmetry and helping me to understand and run the variety of programs used within this thesis I would like to thank Fabio Dall'Antonia. He was not only nearly always available, but also did a great job in proof-reading this thesis.
Gleb Bourenkov taught me how to use his beamline, fixed it also on Sundays when something did not work and always had an explanation when it came to strangely behaving programs, data and radiation damage. Furthermore, he was very helpful when it came to the planning and realisation of new experiments. Thank you very much!
Unfortunately, not all projects I worked on could be finished and are therefore not part of this thesis. Nevertheless, I would like to thank the people involved: Matthew Dunne, Sandra Kozak, Claudia Hackenberg, Sophie Zimmermann and Abul Tarafder for teaching me how to perform protein expression, purification, characterization and crystallization; Guillaume Pompidor, Johanna Kallio and Michele Cianci for helping me out at the beamline and teaching me MAD phasing, Cy Jeffries and Melissa Graewert for performing and analysing SAXS experiments and Christina Krywka and Daniel Laipple (both Helmholtz-Zentrum Geesthacht) for enabling the nano-diffraction experiments.
I am grateful to the European Molecular Biology Laboratory for financial support. I would also like to thank Rosemary Wilson for proof-reading the thesis and hope that native speakers do not suffer too much when reading it now. For the help with the installation of programs and python modules I would like to express my gratitude to the IT group and Ivars Karpics.

Furthermore, I would like to thank the other members of my thesis advisory committee, Rob Meijers and Carsten Sachse (EMBL Hamburg). Special thanks go to Martin Müller, who was not only part of the thesis advisory committee, but also agreed to supervise me once more, despite the fact that cellulose and protein crystals do not have much in common.
Last but not least I would like to thank my boyfriend Malte Ogurreck. He supported me in all stages of this work, convinced me again and again that Python was not invented to drive me crazy, but that it is actually very useful, and ensured that this thesis is also understandable for physicists.

# Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit unter Einhaltung der Regeln zur guten wissenschaftlichen Praxis der Deutschen Forschungsgemeinschaft selbstständig angefertigt habe. Abgesehen von Beratungen mit meinem Betreuer wurden nur die im Literaturverzeichnis angegebenen Hilfsmittel verwendet. Alle wörtlich und sinngemäßen Zitate sind als solche gekennzeichnet.

Die hier vorliegende Arbeit wurde weder ganz noch in Teilen an anderer Stelle im Rahmen eines Prüfunsverfahrens vorgelegt.