UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI BIOLOGIA

DOTTORATO DI RICERCA IN FISIOLOGIA MOLECOLARE E BIOLOGIA STRUTTURALE

XVIII CICLO

PH.D. THESIS

SAXS STUDY OF THE QUATERNARY STRUCTURE OF OLIGOMERIC PROTEINS

TUTORS: PROF. BENEDETTO SALVATO
DR. FRANCESCO SPINOZZI

PH.D. STUDENT: IVAN MIČETIĆ

DECEMBER 31ST, 2005

To my parents

Summary

Small angle X-ray scattering (SAXS) is a powerful tool for quaternary structure determination of proteins in solutions. Even if data acquisition with this technique is relatively simple from the experimental point of view, the data interpretation is not so straightforward. Several methods and programs have been described for the *ab initio* determination of quaternary structure from SAXS measurements, even at relatively high resolution. However, none of these programs utilize some known structural characteristics of the proteins used in this study, hemocyanins. These characteristics are the presence of D_n type symmetries and the geometry of the structural subunits that form oligomeric proteins.

This work describes a new method for the quaternary structure determination of oligomeric proteins from experimental SAXS curves of such whole proteins and simplified models of structural subunits. Structural subunit models are generated from biochemical data, electron microscopy or high resolution structures. These models, composed of spheres with varying radii, were used to reconstruct the quaternary structure of whole proteins by fitting their orientation and position to the experimental SAXS data. The fits were obtained with custom made programs and can be used to model any oligomeric protein with similar symmetry constraints.

The method has been verified with arthropod hemocyanins, where it succeeded to reconstruct the quaternary structure of whole molecules. Consequently, the same method has been applied to much more complex molluscan hemocyanins. In this case, only the positions of the structural subunits could be obtained at a satisfactory level while it was impossible to determine their spatial orientations. The reason is the almost spherical form of structural subunit monomers used for the fits. The method predicts the creation of higher resolution monomer models, but it would require a lot more computing time.

Riassunto

La diffusione a piccoli angoli di raggi X (SAXS) è una tecnica molto efficace nello studio della struttura quaternaria delle proteine in soluzione. Anche se l'acquisizione di tali misure è relativamente facile dal punto di vista sperimentale, la loro interpretazione non è immediata. In letteratura sono descritti diversi programmi con i quali si possono ottenere modelli strutturali direttamente dalle curve SAXS, anche a risoluzione relativamente elevata. Per contro nessun programma tiene conto di alcune caratteristiche note delle proteine oggetto di questo lavoro (emocianine), quali per esempio la presenza delle simmetrie D_n e la struttura delle subunità strutturali che le costituiscono.

Questa tesi riguarda lo sviluppo di un metodo di determinazione della struttura quaternaria delle proteine oligomeriche a partire da curve SAXS di molecole intere e modelli semplificati di subunità strutturali. I modelli delle subunità strutturali sono costruiti partendo da informazioni derivanti da tecniche biochimiche, microscopia elettronica o strutture ad alta risoluzione. Questi modelli, composti da sfere a raggio diverso, sono stati usati per ricostruire la struttura delle molecole oligomeriche, *fittando* i dati su curve SAXS tramite programmi realizzati *ad hoc*. Tali programmi possono essere utilizzati per modellare qualsiasi proteina con caratteristiche di simmetria simili.

Il metodo è stato verificato sulle emocianine di artropodi dove è riuscito a ricostruire in maniera soddisfacente la struttura quaternaria delle molecole intere. A questo punto, l'analisi è stata estesa anche sulle emocianine di molluschi, proteine con struttura molto più complessa. In questo caso, si è riusciti ad ottenere, in maniera soddisfacente, solamente le posizioni delle subunità strutturali che compongono la proteina intera, mentre non era possibile determinarne il corretto orientamento nello spazio. Questo risultato sarebbe da attribuire alla forma pressoché sferica dei monomeri strutturali usati nei *fit*. La generazione di monomeri ad una risoluzione più elevata è prevista dal metodo

ma comporterebbe un tempo di calcolo maggiore.

Contents

Abbreviations

2D	two-dimensional
3D	three-dimensional
AFM	atomic force microscopy
CF	cost function
EM	electron microscopy
EPR	electron paramagnetic resonance
ESRF	European Synchrotron Radiation Facility
FU	functional unit
Hb	hemoglobin
Hc	hemocyanin
Hr	hemerythrin
MM	molecular mass
PDB	Protein Data Bank
SA	simulated annealing
SAS	small angle scattering
SAXS	small angle X-ray scattering
SS	structural subunit

1 Introduction

The scope of this study is to describe a method for determining the quaternary structure of a class of oxygen binding proteins, hemocyanins, with the use of small angle X-ray scattering technique.

Hemocyanins are oligomeric oxygen transport and storage proteins found primarily in the hemolymph of several invertebrate species belonging to the phyla of Mollusca and Arthropoda. Their role is to reversibly bind molecular oxygen by means of two copper ions directly bound to the protein matrix. Hemocyanins are very large molecules with a complex quaternary structure and therefore they are very difficult to analyze with high resolution methods like nuclear magnetic resonance or X-ray protein crystallography. On the other hand, their large size, together with a simple purification and handling, make it an excellent model protein for structural investigations using electron microscopy and small angle scattering techniques.

There are many different methods available for *ab initio* determination of protein structure from small angle X-ray scattering data. However in the case of hemocyanins, many low resolution structural details, ranging from biochemical studies to cryo-electron microscopy of whole proteins, are known. High resolution information has also been obtained from parts of monomers by protein crystallography.

In this study, a method is described that uses this prior knowledge together with small angle X-ray scattering data and custom made programs to build a higher resolution model of the whole hemocyanin molecule. The simulated annealing algorithm was used to search for the best structural conformation that fitted the X-ray scattering data. Protein models composed of spheres of constant and varying radii were used in the process.

This method was built and validated with proteins of known structure (arthropod hemocyanins) and finally, it was applied to a more complex mol-

luscan hemocyanin molecule.

Since SAXS modelling is strictly influenced by constraints such as size, shape and symmetries of the proteins under study, several biochemical and biophysical methods were used including electron microscopy, atomic force microscopy and X-ray crystallography.

2 Small angle X-ray scattering

When an electromagnetic wave impinges to a target, the interaction of the incoming wave with the sample can give rise principally to two phenomena: absorption and scattering. The latter can be coherent, in the case of the elastic or Rayleigh scattering and in diffraction phenomena, or incoherent (inelastic Compton scattering). Many techniques based on these phenomena are applied in the study of condensed state and structural biology. Among them, small angle X-ray scattering (SAXS) was used in the present study.

Small angle scattering (SAS) technique is based on the phenomenon of elastic scattering of an electromagnetic beam and the sample under study. The most important feature in the SAS technique is its potential for the analysis of disordered systems. The radiation implied for SAS mainly consists in X-rays (with a wavelength generally of 0.5–2 Å) and thermal neutrons (wavelength 1–10 Å). This range of incident energies allows the investigation of inhomogeneities of the order of 10–1000 Å. Among many other fields, this technique finds its application in the study of biological systems. In this case, it allows investigating the structure of biological macromolecules in solution. A protein or lipid solution can be seen as a homogeneous sample (the solvent) presenting inhomogeneities of the order of 10–100 Å (the proteins or lipids themselves). In the case of proteins, this technique can give low resolution information on the tertiary and/or quaternary organization in solution.

In the following, a brief description of the fundamentals of the technique is given, applied for X-rays. The same principles are valid for neutron scattering, with the neutron/nucleus interaction instead of X-ray/electron (Vachette & Svergun, 2000).

2.1 Scattering by matter

The incident beam can be described as a plane wave. From the interaction with matter a diffused wave will be emitted. In SAXS technique, only the elastic scattering phenomena are considered, in which there is no change in energy between the beam and the electrons in the sample.

In the case of a sample composed of N atoms, the amplitude of the diffuse wave will be proportional to the scattering length of the electron-photon interaction. In practice, since the number of atoms composing the sample is very high, and the resolution of the technique is not at the atomic level, the sum of the scattering lengths in the sample is described using a continuous density distribution.

If $\rho(\mathbf{r})$ is the electron density of the sample at point $\mathbf{r}$, and $\mathbf{s}_0$ the characteristic vector of the incident beam, at distances much greater than the size of the sample the expression for the amplitude of the scattered radiation is

$$F(\mathbf{s}) = \int_{V_\mathbf{r}} \rho(\mathbf{r}) e^{-2i\pi \mathbf{rs}} \, \mathrm{d}V_\mathbf{r} \tag{2.1}$$

where $\mathbf{s} = \mathbf{s}_1 - \mathbf{s}_0$ and $V_\mathbf{r}$ is the volume of the sample; $\mathbf{s}_1$ is the vector of the scattered wave and $\mathbf{s}$ is the scattering vector. $F(\mathbf{s})$ is the Fourier transform of the electron density distribution $\rho(\mathbf{r})$ and

$$s = |\mathbf{s}| = \frac{|\mathbf{q}|}{2\pi} = \frac{2\sin\theta}{\lambda} \cong \frac{2\theta}{\lambda} \tag{2.2}$$

is the modulus of the scattering vector (fig. 2.1).

The last approximation holds true for small values of θ. In solution scattering experiments, 2θ is generally $< 5^\circ$ (85 mrad) for $\lambda = 1.5$ Å.

The scattering intensity will be the square of the amplitude and is given by the product of $F(\mathbf{s})$ by its complex conjugate:

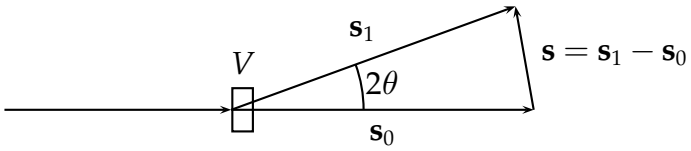
$$I(\mathbf{s}) = F(\mathbf{s}) \cdot F^*(\mathbf{s}) = \int_{V_\mathbf{r}} \int_{V_{\mathbf{r}'}} \rho(\mathbf{r}) \rho(\mathbf{r}') e^{-2i\pi(\mathbf{r}-\mathbf{r}')\mathbf{s}} \, \mathrm{d}V_\mathbf{r} \, \mathrm{d}V_{\mathbf{r}'} \tag{2.3}$$

If $V_\mathbf{r}\gamma(\mathbf{R})$ is the autocorrelation function of the electron density $\rho(\mathbf{r})$ of the sample then

$$\gamma(\mathbf{R}) = \frac{1}{V_\mathbf{r}} \int_{V_\mathbf{r}} \rho(\mathbf{r}) \rho(\mathbf{r}+\mathbf{R}) \, \mathrm{d}V_\mathbf{r} \tag{2.4}$$

and the scattering intensity becomes

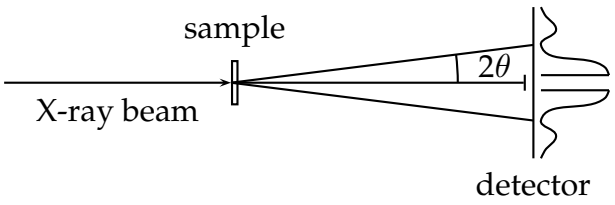
$$I(\mathbf{s}) = \int_{V_\mathbf{r}} V_\mathbf{r} \gamma(\mathbf{R}) e^{-2i\pi \mathbf{R}\cdot\mathbf{s}} \, \mathrm{d}V_\mathbf{R} \qquad \text{with} \qquad \mathbf{R} = \mathbf{r} - \mathbf{r}'. \tag{2.5}$$



$$|\mathbf{s}_0| = |\mathbf{s}_1| = \frac{1}{\lambda}$$

$$|\mathbf{s}| = \frac{2\sin\theta}{\lambda} \cong \frac{2\theta}{\lambda} \qquad 2\theta < 5^\circ = 85 \text{ mrad}$$

(a)



(b)

Figure 2.1: (a) Vectorial representation of scattering by a volume V; (b) schematic representation of a scattering experiment.

2.2 Scattering of proteins in solution

In the case of a sample consisting of proteins in solution, the total scattering density will be given by the difference between the scattering density of the protein $\rho(\mathbf{r})$ and of the solvent ρ_0 which generally can be considered as constant.

If we define the contrast factor as follows:

$$\Delta\rho(\mathbf{r}) = \rho(\mathbf{r}) - \rho_0$$

then the total scattering intensity can be written as

$$F(\mathbf{s}) = \int_{V_\mathbf{r}} \Delta\rho(\mathbf{r}) e^{-2i\pi\mathbf{R}\cdot\mathbf{s}} \, \mathrm{d}V_\mathbf{r} \tag{2.6}$$

Since a protein in solution is a statistically isotropic sample (it presents no preferential orientation in space), only the spherical average of the total intensity can be detected in a SAXS experiment:

$$I(s) = \langle I(\mathbf{s}) \rangle_\Omega$$

where $\langle\rangle_\Omega$ signifies a spherical average and Ω is the solid angle in reciprocal space. After averaging over all orientations with respect to the beam, expressions (2.4) and (2.5) become

$$I(s) = 4\pi \int_0^\infty p(R) \frac{\sin 2\pi Rs}{2\pi Rs} \, \mathrm{d}R \tag{2.7}$$

and

$$p(R) = R^2 V \gamma(R) = \frac{1}{\pi} \int_0^\infty Rs I(s) \sin 2\pi Rs \, \mathrm{d}s \quad \text{with} \quad \gamma(R) = \langle \gamma(\mathbf{R}) \rangle_\Omega \tag{2.8}$$

$p(\mathbf{R})$ is called pair distribution function and for homogeneous particles ($\rho(\mathbf{R})$ is constant), $p(\mathbf{R})$ is the histogram of distances between all pairs of points (volume elements) of the sample. $\gamma(\mathbf{R})$, called the characteristic function of the particle, represents the probability of finding a point within the particle at a distance R from a given point. Integrating this probability over the surface of the sphere of radius R and over the volume V yields the distance distribution function, as expressed by the equation (2.8).

2.3 Structural parameters

If we can assume the protein sample under investigation to be ideal and monodisperse, we can obtain some structural parameters from the analysis of the scattering curve. In our case ideal means that the protein particles in solution are not interacting. Theoretically, this is only true at infinite dilution. In practice, measurements are performed at different concentrations, and an extrapolation of the curve to infinite dilution is performed. The monodispersity of the solution can be very difficult to obtain. Generally, preparative chromatographies are performed in protein samples just before the measurements in order to avoid eventual association/dissociation processes in the sample.

2.4 Data analysis

2.4.1 The Guinier approximation

At small angle values $\left(sR_g = 0.1\right)$, the scattered intensity can be approximated by the Guinier formula (Guinier & Fournet, 1955):

$$I(s) \cong I(0) \exp\left(-\frac{4\pi^2}{3} R_g^2 s^2\right) \tag{2.9}$$

where $I(0)$ is the zero angle intensity and R_g is the radius of gyration of the particle defined as

$$R_g^2 = \frac{\int_{V_\mathbf{r}} \Delta\rho(\mathbf{r}) r^2 \, \mathrm{d}V_\mathbf{r}}{\int_{V_\mathbf{r}} \Delta\rho(\mathbf{r}) \, \mathrm{d}V_\mathbf{r}} \tag{2.10}$$

The zero angle intensity is proportional to the molecular mass of the protein and is given by

$$I(0) = N \iint_V \Delta\rho(\mathbf{r}) \Delta\rho(\mathbf{r}') \, \mathrm{d}V_\mathbf{r} \, \mathrm{d}V_{\mathbf{r}'} = N\left(m^2 - m_0^2\right) = Nm^2(1 - \rho_0\psi)^2 \tag{2.11}$$

where N is the number of particles in the sample, m is the number of electrons in each particle, m_0 is the number of electrons of the solvent displaced by the molecule and ψ is the electronic partial specific volume of the protein. If c is the concentration of the protein (g/L) $c = \frac{N\mu m}{N_A}$, it follows that

$$\frac{I(0)}{c} = \frac{N_A M}{\mu^2}(1 - \rho_0\psi)^2 \tag{2.12}$$

where N_A is Avogadro's number, $\mu = \frac{M}{m}$ is the ratio of the molecular mass to the number of electrons, which only depends of the chemical composition of the sample (for a protein, 1.87 is a good estimate). In this way, from the analysis of the Guinier plot ($\ln[I(s)]$ vs s^2), both the molecular mass and the radius of gyration of the protein can be obtained. Experimentally, the scattered zero angle intensity cannot be measured directly, since it sums with the direct beam. Also it is very difficult to perform an absolute calibration of the intensity of the beam, which would allow to exactly determine the molecular mass of the protein.

2.4.2 Scattering of particles of simple shapes

An approximation of a particle by simple geometrical bodies (spheres, ellipsoids, cylinders) is frequently employed as a first step on the scattering pat-

tern interpretation. In the case of biological molecules in solution, this approximation provides a rough estimate of their shape and size. The advantage of this approximation is that it is possible, for a number of simple geometrical bodies, to express the small angle intensity as a function of their parameters.

For a uniform sphere of radius r, the scattering function was first calculated by Rayleigh (1914):

$$\phi^2(q) = \left(3\frac{\sin(qr) - qr\cos(qr)}{(qr)^3} \right)^2 \tag{2.13}$$

The scattering function of a uniform cylinder of a radius r, height $2h$ has been shown by Fournet (1951) to be given by:

$$\phi(q) = \int_0^{\frac{\pi}{2}} \frac{\sin^2(qh\cos\vartheta)}{(qh\cos\vartheta)^2} \frac{4J_1^2(qr\sin\vartheta)}{(qr\sin\vartheta)^2} \sin\vartheta \, \mathrm{d}\vartheta \tag{2.14}$$

where J_1 is the Bessel function of order 1.

For a hollow cylinder of external radius r_1, internal radius r_2, height $2h$, the scattering function will be (Feigin & Svergun, 1987)

$$\begin{gathered} \phi(q) = \int_0^{\frac{\pi}{2}} S^2(qh) \left(\frac{1}{1-\gamma} \right) \frac{4J_1^2(A)}{A^2} - \gamma^2 \frac{4J_1^2(B)}{B^2} \sin\vartheta \, \mathrm{d}\vartheta \\ \text{where } \gamma = \frac{r_2}{r_1},\ S^2(qh) = \frac{\sin^2\left(qh\cos\frac{\vartheta}{2}\right)}{\left(qh\cos\frac{\vartheta}{2}\right)^2},\ A = qr_1\sin\vartheta,\ B = qr_2\sin\vartheta \end{gathered} \tag{2.15}$$

2.4.3 The Debye method

Many molecules under investigation cannot be approximated satisfactorily by simple triaxial models. This is especially true for proteins in solution that have in general more complex shapes and, sometimes, asymmetric structures. In these cases, models composed of many triaxial bodies can be taken in consideration. The scattering amplitudes of the individual constituent subunits are calculated analytically and added according to the particular orientation in the model. The total amplitude is squared and averaged over all orientations.

According to the Debye method (or sphere method) the whole particle is considered as to be composed of N spheres of equal size. If the center of mass of each sphere is designated by

$$\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N$$

and their amplitudes (with respect to each center) are

$$\phi_1, \phi_2, \ldots, \phi_N$$

the total scattering intensity is given by (Debye, 1915):

$$I(q) = I'(q)\left[N + 2\left\langle \sum_{j\neq k}\sum \frac{\sin\left(qd_{jk}\right)}{qd_{jk}}\right\rangle\right] \tag{2.16}$$

where $d_{jk} = |\mathbf{r}_j - \mathbf{r}_k|$ is the distance between the centers of the j-th and k-th subunit and $I'(q)$ is the intensity scattered by a single sphere, which can be calculated analytically from equation 2.13.

For the calculation of the $p(r)$ distribution function, it is possible to define an equation analogous to the Debye formula (eq. 2.16) in real space as follows:

$$p(r) = \sum_{i=1}^{n} \rho_i^2 p_0\left(r, R_i\right) + 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \rho_i\rho_j \mathbf{p}\left(r, d_{ij}, R_i, R_j\right) \tag{2.17}$$

where ρ_i is the mean electron density of the spherical subunits, $p_0\left(r, R_i\right)$ is the distance distribution function of a sphere with radius R_i, $\mathbf{p}\left(r, d_{ik}, R_i, R_j\right)$ is the cross term distance distribution between the ith and jth sphere separated by $d_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$. The analytical expressions for p_0 and $\mathbf{p}$ can be found in (Glatter, 1980).

2.4.4 The indirect transform method

A very useful tool in data analysis is the indirect transform approach introduced by Glatter (1977). According to this approach, a study of the $p(r)$ function is performed. The advantage is that, unlike the scattering intensity, the $p(r)$ is defined in a limited region of space. For a monodisperse system, $p(r)$ is 0 for $r > D_{max}$, where D_{max} is the maximal dimension of the particle.

This analysis is implemented in the program GNOM (Svergun *et al.*, 1988; Svergun, 1992). The program GNOM allows one to calculate the $p(r)$ function with the D_{max} of the particle being determined by a trial and error procedure and performs then the calculation of the radius of gyration in real space. As a further task it performs the back-transformation of the $p(r)$ function. The scattered intensity thus obtained is compared with the experimental pattern. The radius of gyration of this curve and the zero angle intensity are calculated as well.

2.4.5 Multipole expansion

One powerful procedure for *ab initio* shape determination from experimental solution scattering curves is the multipole expansion method, originally proposed by Stuhrmann (1970a,b) and later implemented by Svergun & Stuhrmann (1991); Svergun (1997); Svergun *et al.* (1996, 1997)

The structure of a particle is described by a series of multipole components, i.e. by the additive composition of a term with spherical harmonic symmetry (monopole) and corresponding terms with dipole, quadrupole etc. symmetry. The excess scattering density of the particle with respect to the solvent is defined as:

$$\rho(r) = \begin{cases} 1 & 0 \leq r \leq F(\omega) \\ 0 & \text{otherwise} \end{cases} \tag{2.18}$$

The particle envelope $F(\omega)$ expressed in spherical coordinates (r, ω), with $\omega = (\theta, \phi)$, is parameterized as

$$F(\omega) \approx F_L(\omega) = \sum_{l=0}^{L} \sum_{m=-l}^{l} f_{lm} Y_{lm}(\omega) \tag{2.19}$$

where $Y_{lm}(\omega)$ are spherical harmonics and f_{lm} are complex numbers. The number of parameters and the spatial resolution of this representation, defined by the truncation parameter L, are equal to $N_p = (L+1)^2$ and $\rho_r = \frac{\sqrt{5}\pi R_g}{\sqrt{3}(L+1)}$ respectively.

The scattering intensity is:

$$I(q) = 2\pi^2 \sum_{l=0}^{\infty} \sum_{m=-l}^{l} |A_{lm}(q)|^2 \tag{2.20}$$

where the partial amplitudes $A_{lm}(q)$ are computed from the shape coefficients f_{lm}. The coefficients f_{lm} are determined by a non-linear optimization procedure that minimizes the R factor between the calculated and the experimental curve.

2.4.6 Utilization of crystallographic data

The interpretation of solution scattering data can take advantage of the prior knowledge of the atomic structure of the entire or part of the macromolecule under study. For this purpose, the program CRYSOL (Svergun *et al.*, 1995) calculates the scattering curves from the atomic model, taking into account the scattering from the solvation shell. CRYSOL evaluates not only the

scattering intensities but also the multipole components of the scattering amplitudes of the macromolecule. This opens the possibility of fast computation of the scattering patterns from assemblies of several macromolecules (or oligomers of one single asymmetric unit). In particular, programs by the same author, taking as input the multipole-calculated curves of two partners in a complex, obtained with CRYSOL, allows to determine the relative position (roto-translations) of one molecule with respect to the other one. The movements are interactively defined using a graphics interface. Upon request, the program performs the calculation of the scattering curve of the new model and the comparison with the experimental one. Since the calculations use the envelope function, they can make use of crystal structures using CRYSOL, or of a direct preliminary calculation of the envelope from the scattering pattern of the isolated partners.

3

Dioxygen carriers and hemocyanins

3.1 Dioxygen carriers

It is conceivable that the appearance and development of proteins specialized in the transport/storage of molecular dioxygen represent an improvement of the power availability for the aerobic organisms. Since the diffusion of dioxygen in solution is quite inefficient as compared with the diffusion in the gaseous state (1.98 $mm^2\ s^{-1}$ versus 0.00025 $mm^2\ s^{-1}$ at 20 °C) and the solubility of this gas in water is rather low (about 0.3 mM at a temperature of 293 K and at p_{O_2} of 0.2 atmospheres), when the oxygen supply to an organism depends only from free diffusion processes in solution, the size of the organism itself is limited to a few millimeters, in order to allow local oxygen partial pressure compatible with the metabolic rate.

Normally, the increase in the dimensions of aerobic organisms requires the development of adequate structures/mechanisms for oxygen distribution to the tissues, and to increase the oxygen capacity of body fluids, unless specific adaptations emerge. As an example, the blood of some Antarctic fishes, belonging to Nototenioideae, is totally missing the erythrocytes containing hemoglobin, as specific adaptation to low temperatures ($T \leq 2$ °C). At this temperature, the viscosity of blood containing the same quantity of red cells as that of fishes living in temperate environments is unacceptably high for blood circulation. In these animals the suitable oxygen supply is realized by a four-five folds increase of heart volume as well as of circulation rate, taking also advantages from the definitely higher oxygen solubility at low temperatures.

The origin of the oxygen transport proteins is currently estimated to be occurred about 600–800 million years ago (van Holde & Miller, 1995).

The most interesting and intriguing issue concerning the oxygen carriers is that the organisms developed for this purpose several classes of proteins, which differ not only in size and structure, but also in active site characteristics with specific molecular mechanisms of oxygen binding (summarized in table 3.1).

	Fe^{2+} heme	Fe^{2+} non heme	Cu^{2+}
Endocellular	Hemoglobins Mioglobins	Hemerythrins Miohemerythrins	
Extracellular	Chlorocruorins invert. Hemoglobins		Hemocyanins

Table 3.1: A summary of respiratory pigments found in vertebrates and invertebrates.

At present, three different oxygen storage/transport proteins are known: hemoglobins (Hbs), hemerythrins (Hrs) and hemocyanins (Hcs). Figure 3.1 shows the phylogenetic distribution of these respiratory pigments of the Metazoa.

Hbs and Hrs are iron containing proteins and can be considered to constitute the most ancient and efficient class of oxygen carriers.

The class of Hbs is heterogeneous in molecular structure and organization. The proteins of this class contain different prosthetic groups, namely an iron-bound protoporphyrin IX macro-cycle (heme group) or, in some invertebrates, an iron-chloroheme group. Hbs are present in almost all vertebrates and in some invertebrates being the most widely distributed class.

Hrs are found mainly in the phyla of Priapulida and Sipunculida. The active site of this class of proteins consists of a Fe(II) binuclear complex, with the metal directly bound to the protein matrix. The crystallographic structure has been determined and extensive X-ray absorption spectroscopy studies have been carried out in these proteins in different chemical forms (Riggs-Gelasco *et al.*, 1995).

Hcs, the subject of this work, represent the most complex member of the oxygen transport pigments. They constitute a family of copper proteins found freely dissolved in the hemolymph of certain invertebrate species belonging to the phyla of Arthropoda and Mollusca. The most striking difference with the

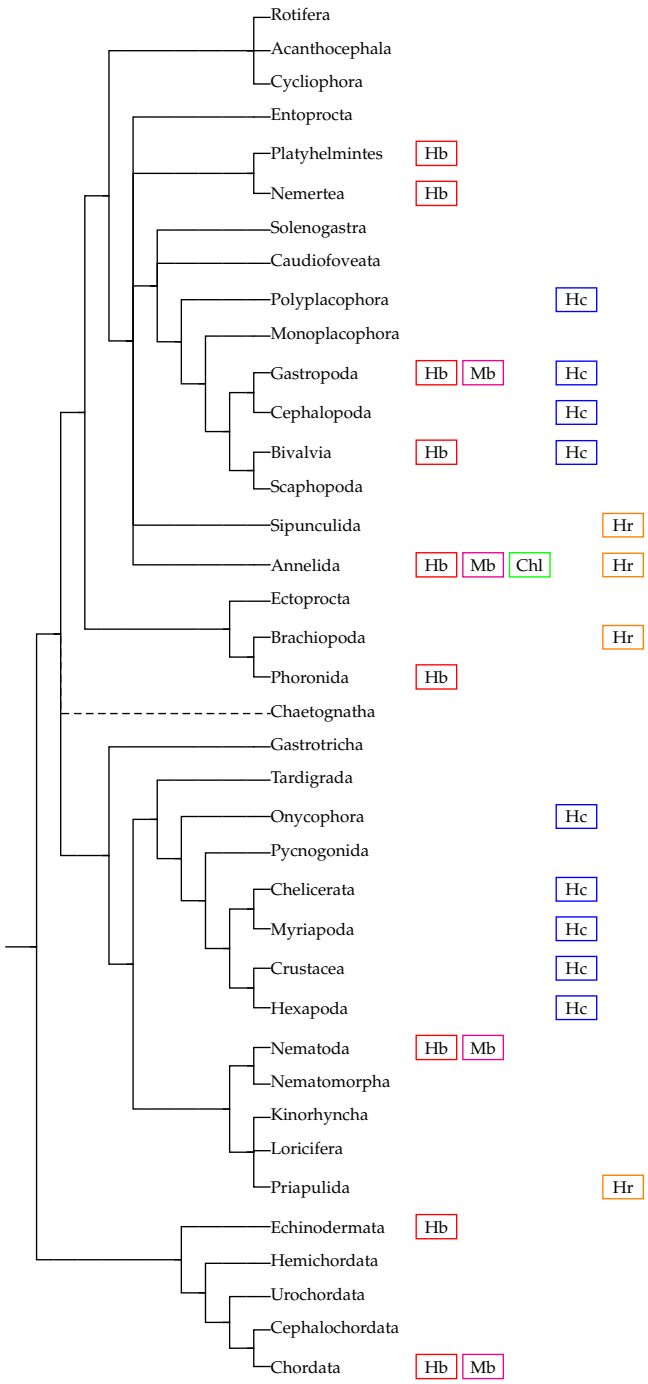


Figure 3.1: Distribution of oxygen carrying proteins. Phylogenetic tree elaborated following Lecointre & Le Guyader (2001); Hwang *et al.* (2001). Mb=mioglobin, Hb=hemoglobin, Chl=chlorocruorin, Hc=hemocyanin, Hr=hemerithrin.

other respiratory pigments is that the active site does not contain iron, neither in the heme group nor directly bound but it contains a couple of copper ions directly bound to imidazole residues provided by the protein matrix.

It is interesting to note that among all the oxygen carriers, only Hcs do not have a mio-hemocyanin muscle counterpart for helping oxygen diffusion or storage. The distribution of oxygen carriers in invertebrate species seems quite erratic. Not all phyla contain respiratory pigments while others have many different types. Furthermore, some species belonging to a given phylum may have more than one pigment or even none. The difficulties to interpret the above distribution peculiarities are clearly linked to the still unsolved problem of the origin and evolution of these proteins.

Some attempt to solve this problem have been proposed by van Holde & Miller (1995). According to their theory of necessity, the development of a specific protein along an evolutionary line is connected to a specific request by the organism. Moreover, it must be taken in consideration that if a protein has an accessory function, its genic expression can be repressed without altering the fitness of the entire organism. Thus, a possible explanation for the wide variety of oxygen carriers would be based on the assumption that, from a common genetic material, various proteins evolved following different patterns being expressed and repressed different times in the various organisms. Alternatively they evolved independently and converged to the same function following completely different evolutionary patterns.

A mechanism for the evolution of arthropod and molluscan Hcs has been proposed (van Holde *et al.*, 1992; Decker & Terwilliger, 2000). In particular, the arthropod copper binding domain could have originated from a simple duplication of a primordial mononuclear "B" type copper binding site whereas the molluscan copper binding domain resulted from a fusion between two genes, one carrying an "A" site and the other a "B" site. The "A" and "B" sites are the names of the copper binding sites as found in present day Hcs: that nearer the N terminus is called the "A" site and that nearer the C terminus is called the "B" site. However, primary sequences analysis of various type 3 copper proteins (Durstewitz & Terwilliger, 1997; Burmester & Scheller, 1996) (including arthropod and molluscan Hcs, catechol oxidase, tyrosinases and phenol oxidases) lead to the idea that all type 3 copper proteins evolved from a common

binuclear predecessor, which itself arose from the duplication of a "B"-like copper site (van Holde *et al.*, 2001). From this data it was also possible to estimate the time of the arthropod Hc-prophenol oxydase split to 550–600 millions of years ago and the molluscan Hc-tyrosinase split to 700–800 millions of years ago (Voit *et al.*, 2000; Burmester, 2001; Lieb *et al.*, 2000).

It is noteworthy to point out that almost all copper enzymes have direct or indirect interactions with oxygen which underlines the quite close relationships that exist between this gas and copper. Many oxidative reactions are catalyzed by copper enzymes. Therefore, it could be hypothesized that these enzymes in a first instance played functions involving in some way the scavenging of molecular oxygen and of its partially reduced intermediates, through the oxidative catalysis of suitable substrates easily found within cells. These functions, during the evolution, have been modified and adapted to new requirements connected with the strong increase of the oxygen partial pressure in the atmosphere and of its consequent larger utilization in biological systems.

It is evident from this brief introduction that the evolution of the oxygen transport proteins is still an interesting but unsolved problem. The achievement of a detailed understanding of the structural properties that are pertinent to the function of these proteins will definitely help in a better understanding on the general frame in which the evolution of oxygen binding proteins occurred.

The present work has been focused on the investigation of the structure of one member of this class of metalloproteins: hemocyanins.

3.2 Hemocyanins

Hcs are found as extracellular oligomers, freely dissolved in the hemolymph of some species of arthropods and molluscs. The molecular architecture, organization and subunit size are different in the two phyla. In Mollusca, the minimal functional unit (FU) (containing one binuclear copper active site) has a molecular mass (MM) of about 50 kDa while in Arthropoda, its MM is about 75 kDa. These subunits contribute to the organization of the native molecule according to completely different patterns.

3.2.1 Distribution

As mentioned above, Hcs are expressed only in some species of classes of invertebrates belonging to the phyla of Arthropoda and Mollusca as shown in fig. 3.1.

Within the phylum Arthropoda, Hcs have been found in some members of four classes: Chilopoda, Crustacea, Merostomata and Arachnida. Molluscan classes containing Hcs are Amphyneura (chitons), Gastropoda, Cephalophoda and Bivalvia (Ellerton *et al.*, 1983).

When they are present, with few exceptions, Hcs generally represent the major component in the hemolymph, constituting the 90–98% of total proteins.

3.2.2 Amino acid composition

While in arthropods the primary sequence of one single structural subunit (SS) consists of about 650 amino acids (657 in *Panulirus interruptus*, 628 in *Limulus polyphemus*) in mollusca it consists of about 450 amino acids. There is overall a low sequence homology among the two phyla (about 10%). The only similarities are located in the area of the second copper binding site (van Holde & Miller, 1995). Within the phylum of mollusca, most of the comparisons show sequence identity of 40 to 46% (Miller *et al.*, 1998). The closest similarities lie not between subunits of the same molecule but between corresponding pairs in different molecules.

It is important to notice that Hcs are glycoproteins. While in arthropods, the N-linked oligosaccharides are present in small amounts, from 1 to 2.5% w/w, in molluscan Hcs they represent from 2.5 to 9% of the total protein mass (van Kuik *et al.*, 1990; Stoeva *et al.*, 1995).

3.2.3 Secondary structure

In arthropods, the polypeptide chain is organized in three domains. Domain I, in the N-terminal region, is mainly folded in α-helices arranged in an antiparallel fashion. This region is deeply buried into the protein matrix and presents a marked hydrophobic character. Domain III is the largest one and is mainly composed by a seven stranded β-barrel structure (fig. 3.2).

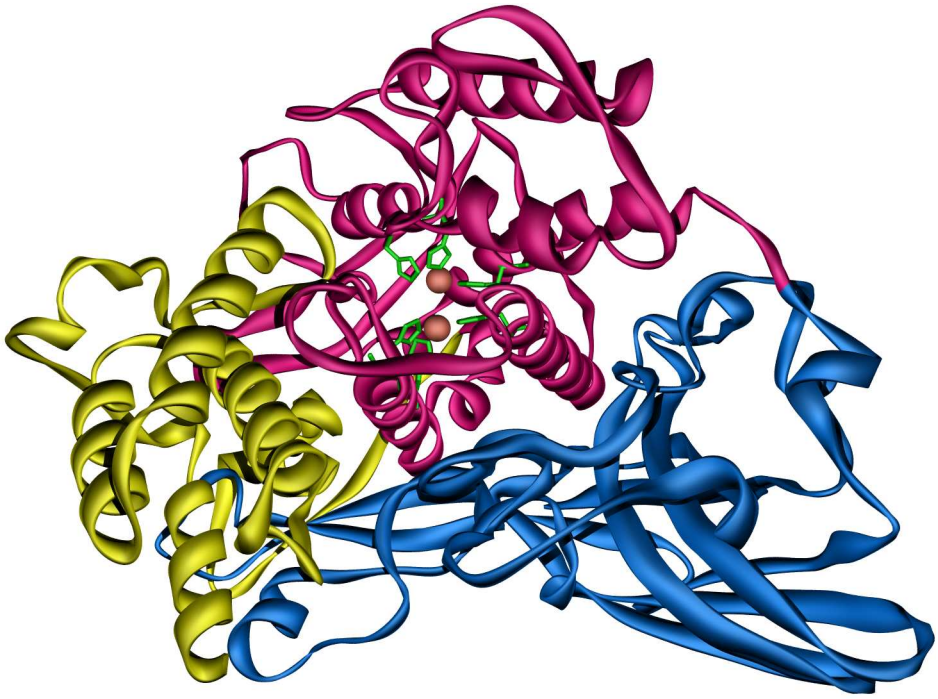


Figure 3.2: Secondary and tertiary structure of *L. polyphemus* FU II. The three domains are shown with different colors (domain I in yellow, domain II in magenta and domain III in blue). Atomic coordinates are taken from the Protein Data Bank (PDB) (Bernstein *et al.*, 1977; Berman *et al.*, 2000) entry 1LLA (Hazes *et al.*, 1993).

In molluscs, the secondary structure can be considered related to that of arthropods after the deletion of domain I and the topological shift of domain III (Jaenicke & Decker, 2004). The X-ray structure of *Octopus dofleini* FU "g" (Od*g*) Hc (Cuff *et al.*, 1998) shows an α-helix domain where the active site is located corresponding to domain II of arthropod Hcs. Domain III of arthropod Hcs is still recognizable in the Od*g* structure, which shows a β-barrel structure similar to that of arthropod Hcs (fig. 3.3).

3.2.4 The active site

The active site in the Hcs is composed of two Cu ions directly bound to the polypeptide chain by three histidine residues.

The definition of the state of oxidation of the two copper ions is the pre-

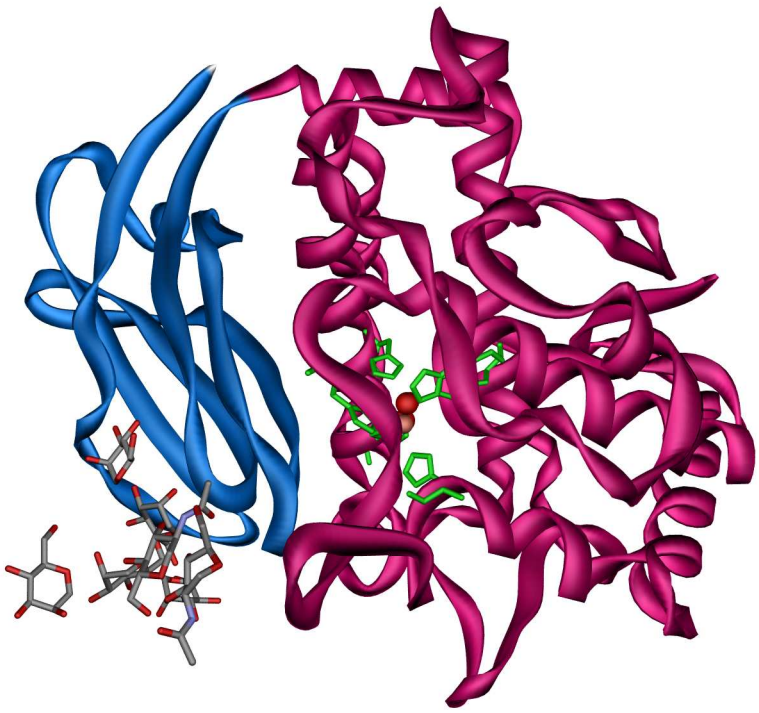


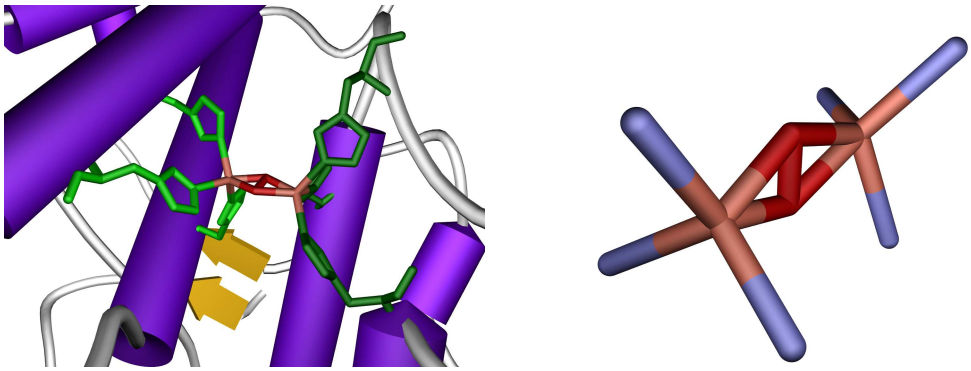
Figure 3.3: Secondary and tertiary structure of *O. dofleini* FU. The two domains are shown with different colors (domain I in magenta, domain II in blue). The color show analogous domains in arthropod FUs (see fig. 3.2). Atomic coordinates are taken from the PDB entry 1JS8 (Cuff *et al.*, 1998).

requisite to understanding the oxygen binding mechanism in these proteins. In the deoxy-form, the two copper ions are in their lower state of oxidation Cu(I) and the protein is colorless and lacks of electron paramagnetic resonance (EPR) signal. In the oxy-form, the definition of the state of oxidation of the Cu ions is more difficult. Even if acquiring a deep blue color, the protein is still EPR silent. Moreover, magnetic susceptibility measurements show that both oxy- and deoxy-Hcs are diamagnetic (Nakamura & Mason, 1960; Dooley *et al.*, 1978; Solomon *et al.*, 1976).

On the other side, oxy-Hc shows an absorption band in the near UV region (about 340 nm). This band is peculiar of some binuclear copper or cobalt complexes and has been attributed to a peroxide-Cu(II) charge transfer transition. Consequently, a model in which two Cu(II) ions are linked by a peroxyde ion in the oxy-Hc has been proposed (Bannister & Wood, 1969). The same conclusion has been drawn from the analysis of circular dichroism and low temperature

absorption spectra (Mori *et al.*, 1975).

On the basis of this results, the diamagnetism of oxy-Hc and the lack of EPR signal has been attributed to a strong antiferromagnetic spin coupling of the two cupric ions through a bridging ligating system while resonance Raman studies have established that oxygen is bound as peroxide dianion (Freedman *et al.*, 1976). The actual explanation on the oxygen binding is that both the copper ions pass from the cuprous, Cu(I), to the cupric, Cu(II), oxidation state with one electron transfer from each metal to the oxygen molecule which, in turn, results to be bonded as a peroxide dianion according to the reaction: [Cu(I) Cu(I)] $+O_2 \leftrightarrows$ [Cu(II) O_2^{2-} Cu(II)]. The synthesis and characterization by Kitajima *et al.* (1992) of $\mu - \eta^2 : \eta^2$ peroxo dinuclear Cu(II) complexes has lead to the currently accepted formulation of the oxy-Hc in which one peroxide molecule is bridging the two Cu(II) ions and no external ligands are present in the active site. The X-ray diffraction data (fig. 3.4) as well as X-ray absorption spectroscopy analysis (Sabatucci, 1999) seem to confirm this formulation.



(a) (b)

Figure 3.4: (a) Coordination of copper ions in oxy-Hc (Miller *et al.*, 1998). (b) Schematic representation of the geometry of copper binding atoms at the active site.

3.2.5 Reactivity of the active site

The Hc reactivity refers essentially to the reactivity of its basic forms namely oxy-, deoxy- and apo- form.

The Hc active site is an asymmetrically exposed binuclear metal site deeply

embedded into an hydrophobic cage provided by aliphatic and aromatic side chains as evidenced from X-ray crystallography. This cage is fundamental in controlling the reversibility of the oxygen binding process because it provides an electronic insulation, which prevents any electron transfer involving the binuclear metal site. Furthermore, the low dielectric constant of this surrounding strongly helps the reversibility of the oxygen binding process, increasing the strength of electrostatic interactions between the two Cu(II) and the charge of the peroxide dianion. Accordingly, only molecular oxygen can enter/leave this kind of site. As a consequence of the asymmetry of metal ions exposure to the external medium, the reactivity of Hc active site is dominated by the most exposed metal ion, namely copper A. For this reason, it has been called "fast reacting". The role of the more internal copper (copper B) consists in the maintenance of the conformation and tightness of the active site region. It provides a very important electrostatic potential contribution to the reactivity of Cu_A, improving the reversibility of the oxygen binding process. Oxygen is the only Hc ligand that is bound as a bridge in between the two metal ions in a $\mu - \eta^2 : \eta^2$ binding modality.

Quite interestingly, Hr, the non-heme iron oxygen binding proteins, need two metal ions too to assure the reversibility of dioxygen binding. In this case, oxygen is bound by just one of the two iron ions, the other one providing an essential electrostatic assistance to the reversibility of the binding process. The Fe-Fe distance in Hr is 3.5 Å, exactly the same as Cu-Cu in Hcs (Riggs-Gelasco *et al.*, 1995).

Oxy- and deoxy-Hc forms can be considered in thermodynamic equilibrium depending on the concentration of oxygen in solution. The deoxy-form can react with a variety of monodentate and bidentate ligands. Among them, three molecules are peculiar: oxygen bound as a bridge, carbon monoxide, whose binding produces a luminescence emission, and cyanide, which is the only molecule able to sequentially extract the two metal ions giving the apo form. This last is a two-step reaction. In the first step, only the most accessible copper ion is removed. More slowly or upon increasing cyanide concentration, the other copper ion will be removed.

Oxy-Hcs too can react with suitable molecules as cathecols. In this case, the reaction center is located on the bounding peroxide. The reaction follows a free

radical chemistry producing quinons and consuming oxygen. This peculiarity of oxy-Hc constitutes the basis of cathecol oxidase, peroxide dismutase and peroxydase activity of Hcs (Salvato & Beltramini, 1990). These activities can be considered true enzymatic reactions where Hc is working as a catalyst. The similarity with tyrosinases and phenol oxydases is only apparent having the reaction mechanisms and stoichiometry quite different. It must be underlined that tyrosinase and phenol oxydase are enzymes that start a reaction which must be quenched in a short time. This normally occurs with the irreversible damaging of the protein itself during the reaction (Salvato *et al.*, 1998).

3.2.6 Quaternary structure

The quaternary structure of both arthropodan and molluscan Hcs are characterized by basic units built up by several FU non covalently bound in the case of arthropodan Hcs and covalently bound for molluscan Hcs. This basic unit is called FU or oxygen binding unit as it is responsible for the binding of a single oxygen molecule. In arthropods the FU is also the SS but in molluscs, different numbers of covalently bound FUs represent the SS. Native Hcs are oligomers made from six (in arthropods) and ten and more such SS (in molluscs and some arthropods).

Arthropods

The basic structure of an arthropod Hc has a MM of about 450 kDa and sedimentation coefficient of 16 S, consisting of six kidney-shaped FU, each containing a single active site of about 75 kDa and sedimentation coefficient 5 S. They are arranged to form an hexameric trigonal antiprism (point symmetry 322). The dimensions of this structure determined by electron microscopy (EM) techniques and X-ray diffraction are in good agreement.

The basic, hexameric structure (fig. 3.5) or building block, can further oligomerize as to form dimers (900 kDa, 24 S), tetramers (36 S), hexamers (49 S) up to the highest aggregate, the octamer (62 S) found in the hemolymph of the horseshoe crab *L. polyphemus*. It is worthwhile to note that often, only one of these aggregate forms is present in the hemolymph of a given species. When different aggregation states are present, the concentration of one of them is

much higher than all the others. Fig.3.6 shows the distribution of Hc aggregation states in the arthropods.

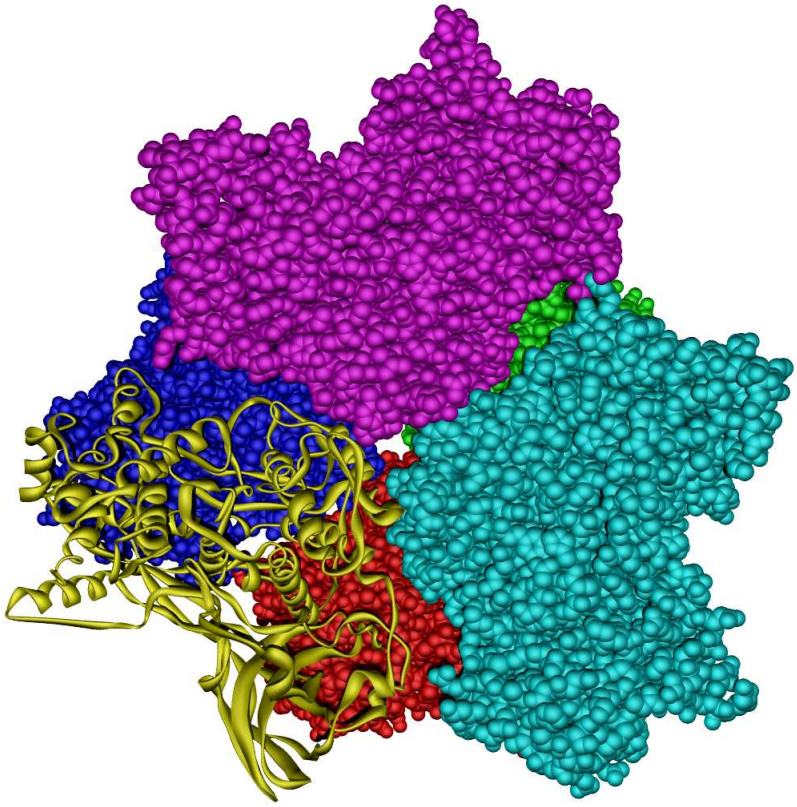


Figure 3.5: Quaternary structure of a *P. interruptus* basic hexameric unit. Atomic coordinates are taken from the PDB entry 1HC1–1HC6 (Volbeda & Hol, 1989).

Generally, the 5 S FU of arthropodan Hcs are a heterogeneous population of isoforms with different amino acid composition and oxygen binding properties. These heterogeneities have an important physiological role in the modulation of the stability of the basic unit and the building-up of its oligomers. The different kinds of subunits can be often efficiently separated by ionic exchange chromatography and electrophoresis. Various studies on distinct subunits have shown that they are characterized by distinct amino acid chains and that they contribute to the formation of the hexamer with different percentages. Very often, however, unlike the native molecules, the isolated subunits show no cooperative behavior (Markl & Decker, 1992; Truchot, 1992).

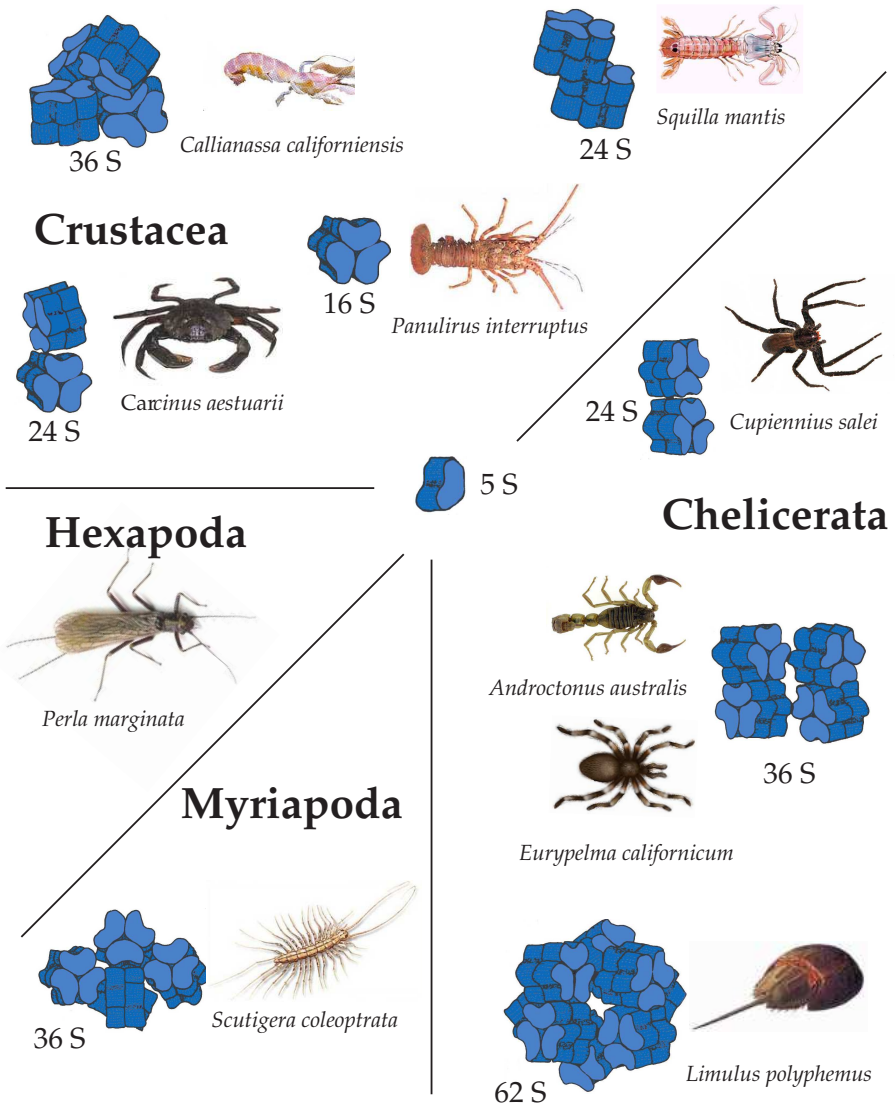


Figure 3.6: Quaternary structure of Hcs extracted from different arthropods. All multimers are composed of a minimum SS 5 S and its hexamer, 16 S.

The association/dissociation behaviors of aggregates and basic units of arthropod Hcs are controlled by the presence of alkaline metal ions and, more efficiently, by pH. These processes are almost completely reversible in the pH range between 5 and 10. The dissociation of the aggregates into basic units is mainly controlled by the concentration of Ca^{2+} or Mg^{2+} ions, whereas alkaline pH allows the dissociation of the basic units. Lowering back the pH into neutrality is often enough to obtain the reassociation of 5 S chains. In the case of reassociation of single purified 5 S components, the 16 S molecules formed very often do not present any cooperativity, in contrast with the native proteins.

The most detailed information on the structure of the arthropodan Hcs come from the X-ray crystallography data. Up to now, different proteins have been crystallized: a deoxy-form from *P. interruptus* (containing two subunits) (Volbeda & Hol, 1989) whose crystals diffract up to 3.2 Å resolution and homohexamer of subunit II of *L. polyphemus* both in oxy-form that diffract at 2.4 Å resolution (Magnus *et al.*, 1994) and in deoxy-form at 2.18 Å resolution (Hazes *et al.*, 1993). The crystal can be described as a trimer of dimers, since the subunit contacts seem to be tighter at the interface along the 2-fold axis than that along the 3-fold one.

Molluscs

Molluscan Hcs follow a completely different organization pattern as opposed to the arthropods. They assemble into truly enormous complexes with high MM. Fig.3.7 shows the relative size of two types of molluscan Hcs in scale with the 70 S ribosome.

At the EM they show a hollow cylindrical symmetry, the dimensions of the cylinder depending on the different species (fig. 3.8). Micrographs of gastropod Hcs show an external diameter of about 330 Å and an inner cavity of about 100 Å, the lateral projections are about 350–380 Å high and appear to be arranged in six parallel rows, perpendicular to the cylinder axis. Cephalophod and monoplacophoran Hcs show the same diameter at the EM but a lower height (about 160 Å) and different inner radii (fig. 3.9). The cylinders have an overall D_5 symmetry with the pentameric axis running through the main cylinder axis (Mellema & Klug, 1972).

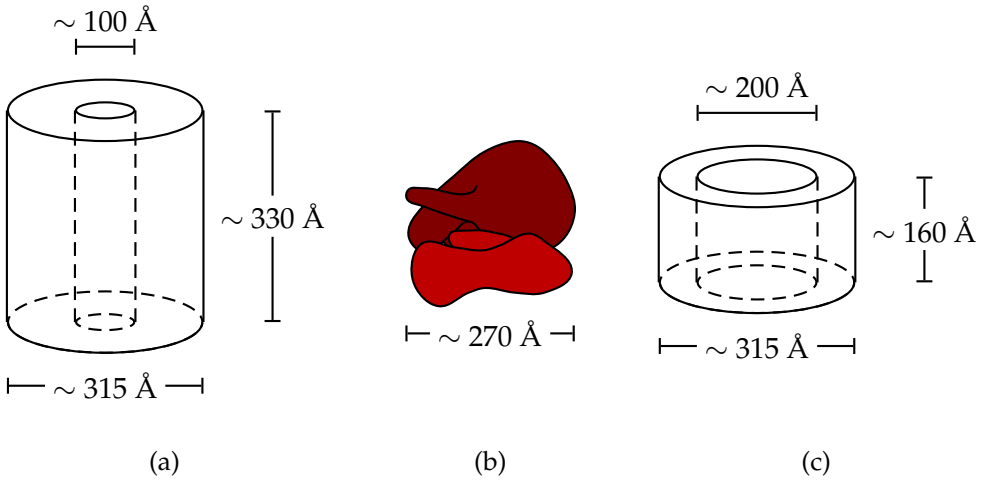


Figure 3.7: The dimensions of *Rapana venosa* (a) and *Octopus vulgaris* (c) Hcs shown in scale to the 70 S ribosome (b).

Hemocyanin	Dissociation pattern
Gastropod	102 S $\leftrightarrow$ 64 S $\rightarrow$ 20 S $\rightarrow$ 16 S $\rightarrow$ 11 S
Cephalopod	Decapod 57 S $\rightarrow$ 20 S $\rightarrow$ 16 S $\rightarrow$ 11 S
	Octopod 49 S $\rightarrow$ 20 S $\rightarrow$ 16 S $\rightarrow$ 11 S

Table 3.2: The dissociation pattern of molluscan Hcs.

The current view of the entire protein includes a MM of about 9 MDa in gastropods (105 S) while in cephalopod is about 2.5–3.8 MDa.

Unlike arthropod Hcs, in the molluscan proteins the FU (containing a single binuclear copper site) do not correspond to SS. The minimal SS (11 S) is common to all classes, and is a unique polypeptide chain composed of seven (in cephalopods) or eight (in gastropods) different oxygen binding domains of about 50 kDa each (Miller *et al.*, 1998; Cuff *et al.*, 1998; Lang, 1988; Lang & van Holde, 1991). This SS assembles to decamers in cephalopod or didecamers in gastropod Hcs (fig. 3.10).

As in the arthropod Hcs, raising of pH and the removal of divalent cations produces the dissociation of the native form. Gastropod Hcs reversibly dissociates in a 64 S species which further dissociates to 16 S and finally 11 S form. The dissociation pattern of cephalopod Hcs, as detected by ultracentrifugation methods, passes through a 20 S form at physiological pH and one again to an 11 S form, at alkaline pH. The dissociation pattern is reviewed in table 3.2.

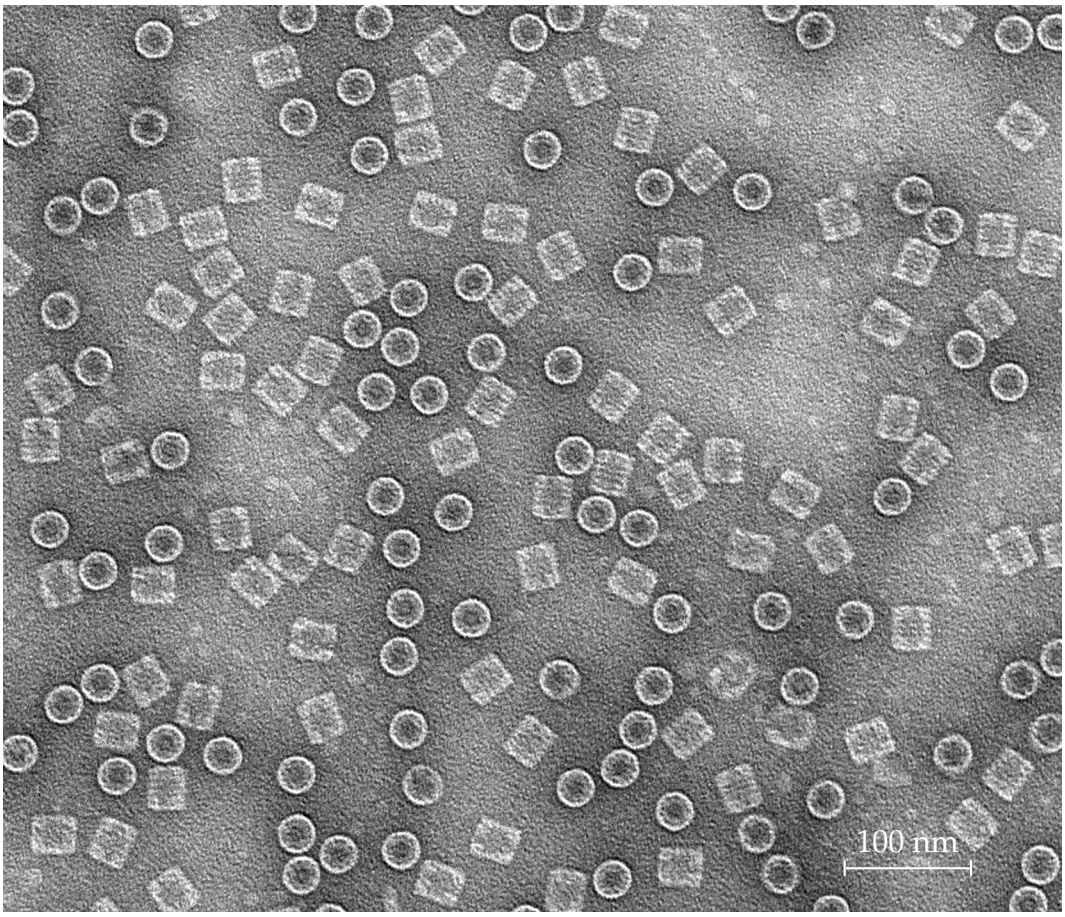


Figure 3.8: Electron micrograph of negatively stained *R. venosa* Hc. End views of the molecules appear as circles and side views as rectangles.

The peculiar feature of the native molecule structure is the dimension of the cylinders. Even if many proteins are organized in cylinders, it is very hard to find stable cylindrical molecules with limited dimensions as in the case of molluscan Hcs. Super aggregation to form rod-like particles can be induced in buffers containing excess Ca^{2+} ions or in dissociation/reassociation experiments (van Holde *et al.*, 1992). High concentration of Ca^{2+} can induce the formation of ordered micro domains showing random orientations (fig. 3.11).

Given the large size and high molecular mass of this protein, it is very difficult to crystallize the native form. All the information we have on the quaternary structure of this form come mainly from the EM techniques and SAS experiments (Meissner *et al.*, 2000; Mouche *et al.*, 2003; Sabatucci, 1999) (fig 3.12). On the contrary, the structure of two 50 kDa FUs have been solved by X-ray

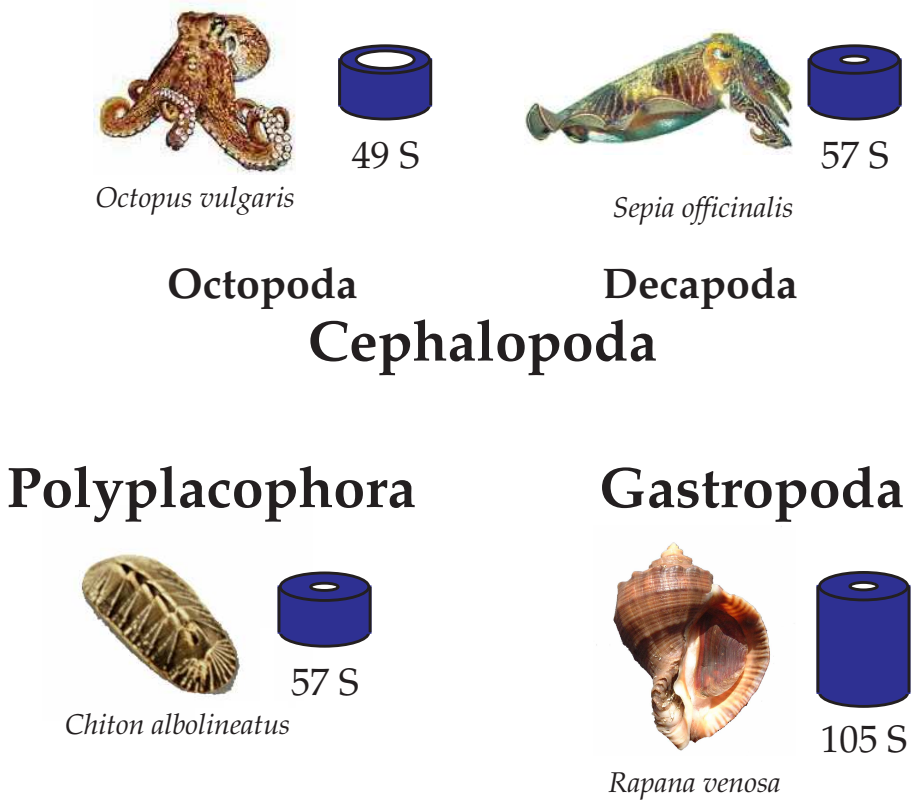


Figure 3.9: Distribution of various types of Hc molecules in molluscs.

crystallography: the oxy-FU Od*g* of *O. dofleini* (Cuff *et al.*, 1998) at 2.8 Å resolution and the deoxy-FU RtH2*e* of *R. venosa* (Perbandt *et al.*, 2003) at 3.38 Å.

3.2.7 Cooperativity

All Hcs bind oxygen cooperatively and some of them like *Eurypelma californicum* Hc have a very strong cooperative binding with Hill coefficients of 6–7 (Hartmann & Decker, 2002; Erker *et al.*, 2005). This requires the existence of different protein conformations. Extensive studies have been carried out on arthropod Hcs involving the analysis of oxygen binding and it's regulation by allosteric effectors (Menze *et al.*, 2001; Hellmann *et al.*, 2001). The protein conformations in oxy- and deoxy- form present differences at all hierarchical levels of the structure of complex multi-hexameric arthropod Hcs. High resolution crystallographic structures of oxy- and deoxy-homohexamers of *L. polyphemus* Hc showed not only different Cu-Cu distances but also a slight shift of domain

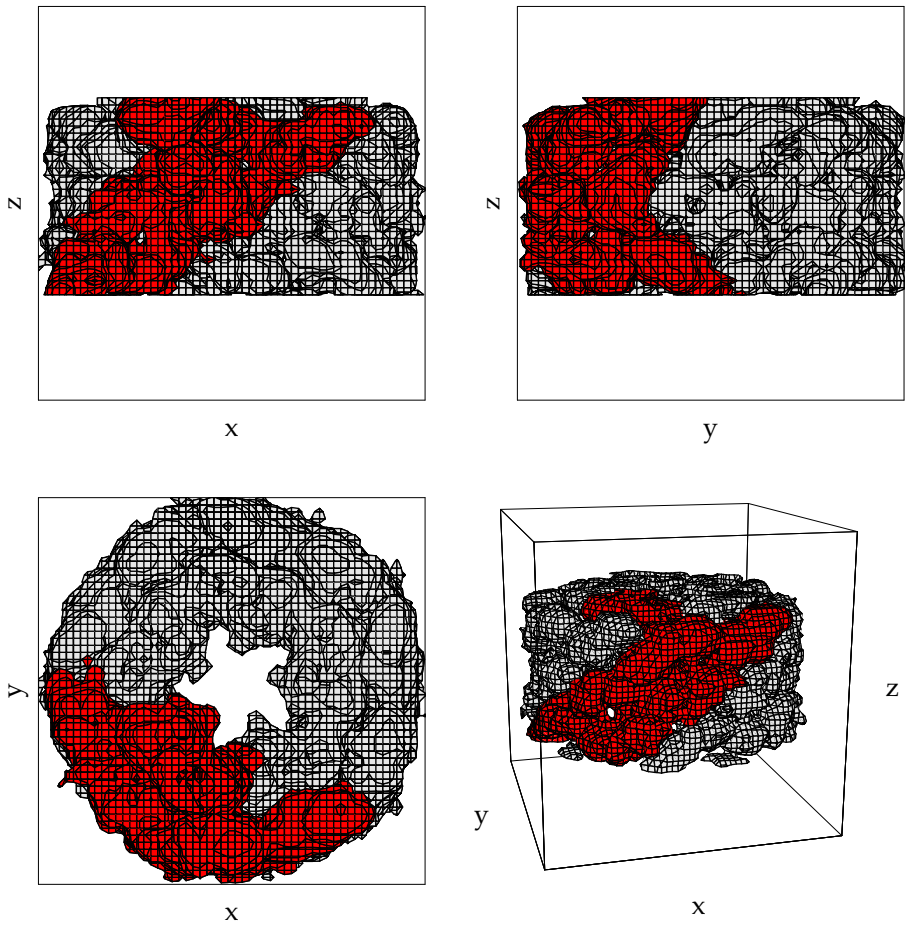


Figure 3.10: Electron density of the upper decamer of *Haliotis tuberculata* type 1 Hc by cryo-EM. In red is shown the SS (Meissner *et al.*, 2000).

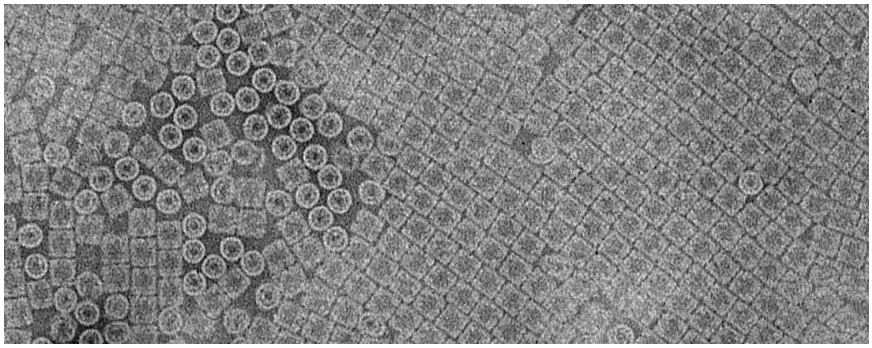


Figure 3.11: Electron micrograph of negatively stained *R. venosa* Hc treated with excess Ca^{2+} in solution (40 mM).

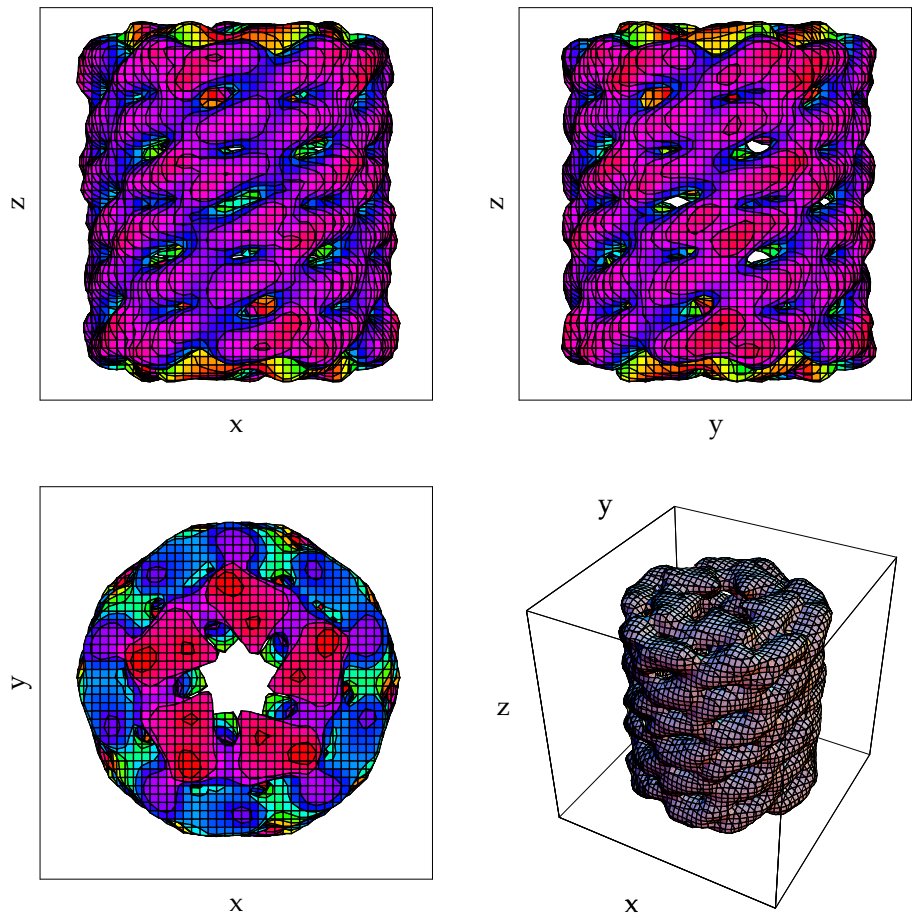


Figure 3.12: 23.5 Å Electron density map of the *Megathura crenulata* type 1 Hc didecamer obtained by cryo-EM and automated data collection (Zhu *et al.*, 2003).

I with the respect to the other two domains (Hazes *et al.*, 1993; Magnus *et al.*, 1994; Volbeda & Hol, 1989). This movement rotates one trimer of the hexamer against the other and is transmitted between tightly associated hexamers in 2×6-meric half molecules and then to whole 4×6-meric, as shown from SAXS and cryo-EM data on *E. californicum* Hc (Hartmann & Decker, 2002; Decker *et al.*, 1996). However, this structural transition lags behind oxygen binding (Erker *et al.*, 2005).

Much less information is known about functional properties of molluscan Hcs. It has been shown, from optical diffraction of EM images and small angle neutron scattering data, that molluscan Hcs exist in two structurally different

conformations in oxy- and deoxy- conditions (van Breemen *et al.*, 1979; Hartmann *et al.*, 2001). In contrast to the arthropod Hcs, the changes in quaternary structure of *M. crenulata* molluscan Hc are rather small and involve the rearrangement of two C-terminal located FUs while only minor movements can be observed within the 60 N-terminal located FUs that make the wall of the cylinder (Hartmann *et al.*, 2001).

4 Materials and methods

4.1 Protein collection and purification

The proteins utilized in this work are Hcs from the gastropod mollusc *Rapana venosa* collected in the Black sea (Varna, Bulgaria). The proteins were purified from the hemolymph of live animals after a cut in the foot. The hemolymph thus obtained was maintained in ice with the addition of protease inhibitors (Complete®, Boheringer, Manheim).

The purification of the native protein involved first a low-speed centrifugation (20 minutes at 27000 g) to eliminate debris and insoluble impurities and then the solution was dialysed overnight at 4 °C against Tris-HCl 10 mM, $CaCl_2$ 20 mM, pH 7.5 buffer. The heavy red-yellow-brown precipitate formed during dialysis, consisting of carotenoids and other pigments was removed.

Hcs were obtained by sedimentation in a preparative Beckman XL70 ultracentrifuge (fixed angle rotor 70 TI), for 90 minutes at 310000 g and 4–6 °C. The dark blue pellet was redissolved in the Tris-HCl 20 mM, $CaCl_2$ 20 mM, pH 7.5 buffer up to a final concentration of 40–60 mg ml^{-1}. The solutions were added up to 20% w/v of sucrose as a cryoprotectant and were stored in a frozen state at -20 °C. The experimental samples were prepared by dialysis of the appropriate quantity of protein against the chosen buffer.

All chemicals were of analytical grade and were used without any other purification.

4.1.1 Preparation of protein monolayers

A solution of PEG 1000 0.05% w/v and 100 μg ml^{-1} Hcs was mixed 1:1 with 1% w/v $(NH_4)_6Mo_7O_{24}/NH_3$ at pH 6.2 buffer. A drop of this solution was placed on freshly cleaved mica dried with P_2O_5 and the excess was blotted. The mica with the protein solution was left to evaporate under a closed Petri

dish overnight.

4.2 Spectroscopic measures

Protein concentration was determined with a Hewlett Packard 8452 diode array spectrophotometer and a Perkin Elmer Lambda 16 spectrophotometer. All spectra between 240 and 600 nm were corrected against light scattering effects using the equation (Salvato *et al.*, 1979): $A_c(\lambda) = A_m(\lambda) - \left(\frac{K}{\lambda^4}\right)$ where $A_c(\lambda)$ is the corrected absorbance, $A_c(\lambda)$ is the measured absorbance and K is the absorption amount at λ_{ref} multiplied λ_{ref}^4 (λ_{ref} is the reference wavelength, 450 nm, where it is assumed that the protein has zero absorption).

The protein concentration was calculated from the absorbance at 278 nm with the extinction coefficient $\varepsilon_{278}^{0.1\%} = 1.38$ ml mg^{-1} cm^{-1} per l'Hc di *R. venosa* (Boteva *et al.*, 1991).

4.3 Electron microscopy

Electron micrographs of Hcs were obtained by unbuffered negative staining with 1% uranyl acetate and 0.01 mg ml^{-1} of final protein concentration. Samples were deposited on glow-discharged carbon support films. Observations, made in the Department of Biology of the University of Padova (Padova, Italy) using a Hitachi H600 transmission EM, were carried out at a magnification of 400000 $\times$ and recorded on Kodak SO163 planar film in turn developed in full-strength Kodak D19 developer for 12 minutes.

For the examination of Hcs monolayers, the mica with the protein were carbonated and the protein was separated from mica by flotation on 1% uranyl acetate and successively transported to a 400-mesh copper grid.

All the preparations and measurements were carried out in collaboration with Mr. Giuseppe Tognon, CNR Center for biomedical technologies, University of Padova (Padova, Italy).

4.4 Atomic force microscopy

Atomic force microscopy (AFM) of dried protein monolayers on mica were made on a Molecular Imaging AFM in contact mode at the Department of Chemistry, University of Firenze (Firenze, Italy) with the collaboration of Dr. Massimo Innocenti and Daniela Besana (INFM Operative Group Grenoble, Grenoble, France).

4.5 Protein crystallography

Whole proteins were purified from lower MM fragments by Fast protein liquid chromatography (Amersham Pharmacia Biotech) with Superose 6B and Superdex 200 HR 10/30 gel filtration columns. Consequently, the protein samples were dialyzed against $CaCl_2$ 20 mM, Tris/Hcl 20 mM pH 7.0 buffer by ultracentrifugation (Centricon concentrator). The protein concentrations used in all crystallization trials were from 2–12 mg ml^{-1}.

Crystallization screens have been performed using the robot at the EMBL Grenoble (Grenoble, France). More than 600 trials were set up by the sitting drop vapor method, at 20 °C. Crystal screen kits commercially available from Hampton Research, Molecular Dimension, Emerald Biostructure and Jena Bioscience were used as crystallizing agents in the reservoir. Crystallization drops were prepared by mixing 0.05 µl of protein solution with the same volume of reservoir in the 96-well Greiner plates. The trials have been monitored by the Mosquito apparatus (Molecular Dimension) that takes images at regular time interval (once a day for 6 weeks). After a couple of hits, PEG 6000 and 8000 were identified as a good precipitant for Hcs. The conditions were then optimized by hand, by fine tuning of PEG 4000–8000 concentrations, pH in 7.0–8.6 range and additives in 24- or 48-well Linbro plates. All trials have been performed at 20 and 4 °C.

The crystals were mounted in a cryoloop, dipped in glycerol 15–24% v/v as a cryoprotectant and immediately transferred to a nitrogen stream at 100 K. They have been tested on the synchrotron light sources at the European Synchrotron Radiation Facility (ESRF) (Grenoble, France), on high brilliance beamlines ID13, ID23 and ID29.

All protein crystallography, X-ray diffraction data collection and analysis were made in collaboration with Prof. Adriana Zagari department of Biology, University of Napoli "Federico II" (Napoli, Italy) and Dr. Consiglia Tedesco, University of Salerno (Salerno, Italy).

4.6 SAXS sample preparation and measurements

SAXS measurements have been taken at the ESRF (Grenoble, France), public beamline ID2 (high-brilliance) under the supervision of Dr. Theyencheri Narayanan, Dr. Stéphanie Finet and Dr. Claudio Ferrero.

The measurements were made under standard experimental setup. The wavelength of the monochromatic $\left(\frac{\Delta\lambda}{\lambda} = 10^{-4}\right)$ X-ray beam was 0.995 Å (at $E = 12.4$ keV). The scattering pattern was recorded on a two-dimensional (2D), ESRF developed, FReLoN CCD camera coupled to a Thomson X-ray image intensifier (XRII) (Narayanan *et al.*, 2001). The CCD resolution was 2048×2048 pixels or 1024×1024 in binned mode. The sample-detector distance was set to three meters so that the scattering vector q ranged from 0.079 to 2.315 nm^{-1} (with q defined as $q = 2\pi s = \frac{4\pi}{\lambda}\sin\theta$). All the measurements were done at different protein concentrations to improve signal-to-noise ratio and to ensure that no protein-protein interactions were present. Data reduction was performed using the in-house programs made by Peter Bösecke and involved flux, detector response, exposure time and sample transmission normalization and in a subsequent step frame averaging, azimuthal integration and buffer subtraction to produce a mono-dimensional scattering profile (fig. 4.1).

The samples were prepared according to table 4.1. For samples number 3 and 4 the salt was added just before the experiments and for the other samples a dialysis was performed with at least three buffer changes.

Sample	Buffer
1	Tris 50 mM, $CaCl_2$ 20 mM, pH 7.5
2	Phosphate 100 mM, pH 7.5
3	Tris 50 mM, NaF 100 mM , pH 7.5
4	Tris 50 mM, NaF 300 mM , pH 7.5

Table 4.1: Composition of *R. venosa* samples examined with SAXS.

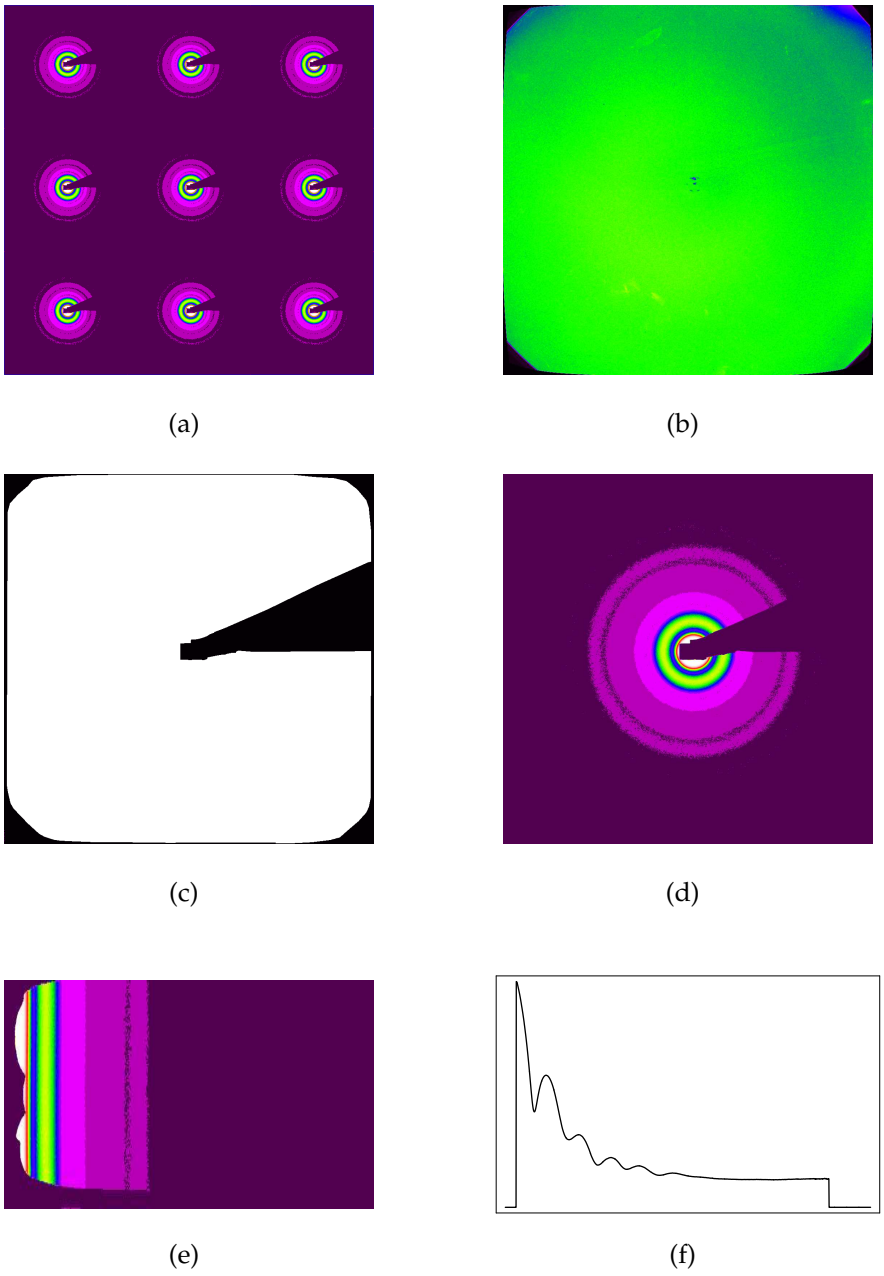


Figure 4.1: Example of SAXS data reduction: (a) acquiring of multiple raw exposures; (b) flat exposure; (c) invalid pixels mask; (d) normalized and averaged image; (e) azimuthal integration; (f) monodimensional scattering profile.

The samples were tested with different acquisition times to determine the longest exposition without radiation damage. Expositions of 0.2 and 0.5 s were chosen as the best ones.

4.7 SAXS data modelling

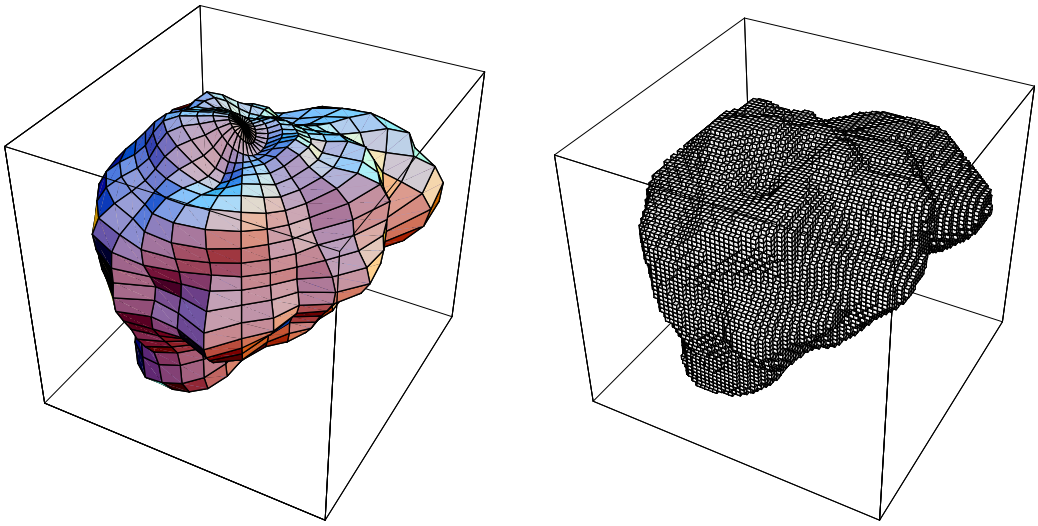
The experimental SAXS data were analyzed by indirect transform program GNOM for the estimate of the radii of gyration and maximum dimensions of particles. The envelope of arthropod and molluscan Hc FUs has been obtained by multipole expansion methods computed using the program CRYSOL up to the harmonics order 15. The atomic coordinates of the FU were obtained from the PDB under the accession numbers 1HCY for arthropods (*P. interruptus* FU at 3.2 Å, Volbeda & Hol (1989)) and 1LNL for molluscs (*R. venosa* FU at 3.3 Å, Perbandt *et al.* (2003)).

The obtained envelopes of the proteins were analyzed by custom made programs written in *Mathematica* (version 5.0, Wolfram Research, Inc. (2003)) and GNU Pascal (gpc-20040516, Free Software Foundation (1991b)). The listings of all custom made programs can be found in appendix A–C. The analysis was divided into following steps: determination of the discrete protein mask, generation of sphere models from the mask, theoretical scattering curve and cost function calculations, volume scale fitting of the sphere models and SAXS data oligomer modelling.

4.7.1 Discrete protein mask

Since working with spherical harmonics is computationally expensive, it was decided to transform the continuous shape information of the protein to discrete form represented by a shape mask: a 0/1 3D matrix with cells of 1 $Å^3$. In this case 0 represents the solvent and 1 the protein.

The *Mathematica* program `ReduceFromY.nb` does this operation (appendix A.1.1). It takes as input the shape parameters f_{lm} obtained from the program CRYSOL and generates the 0/1 shape mask of the protein (fig. 4.2).



(a) (b)

Figure 4.2: Discrete protein mask creation. Protein envelope (a) and its discrete shape mask (b).

4.7.2 Sphere models generation

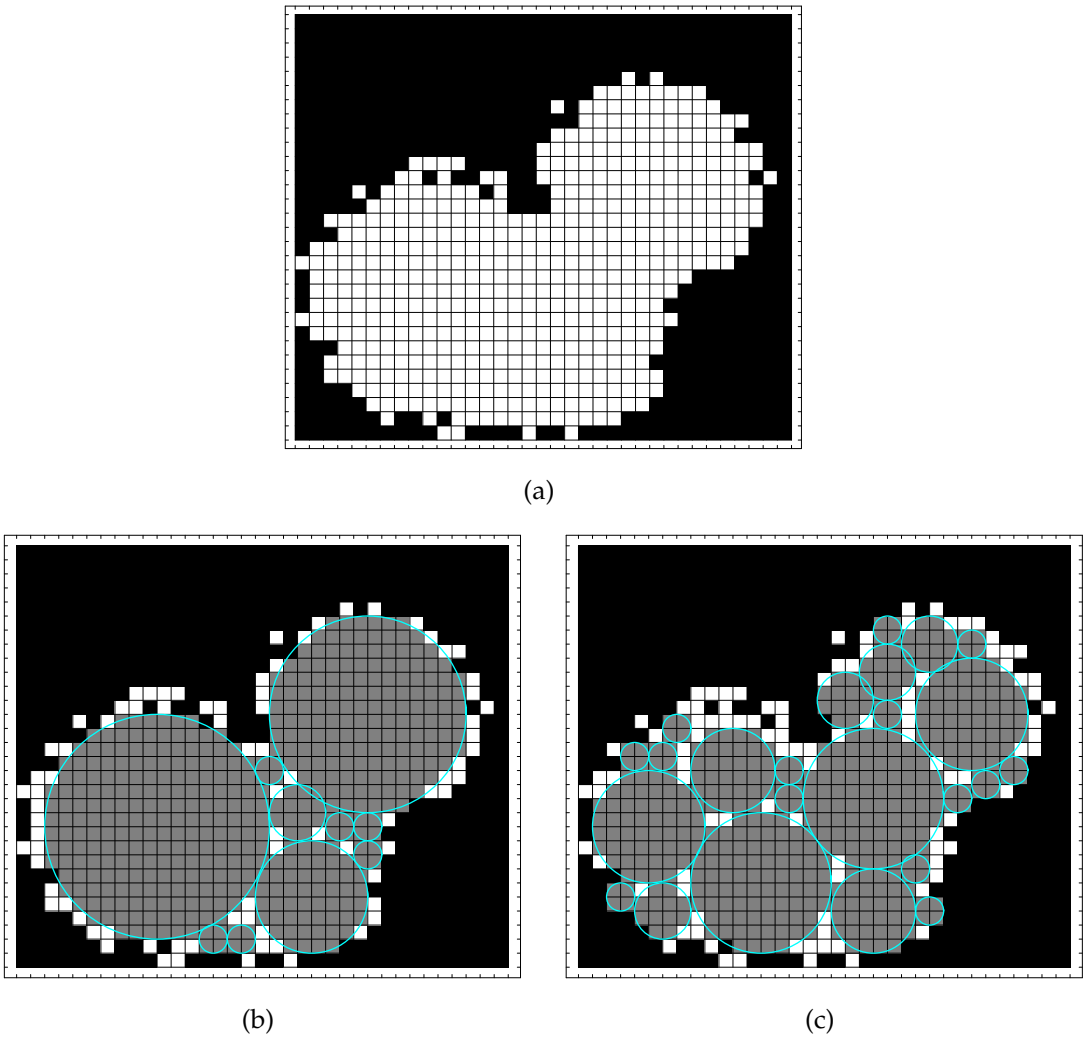
Given the 0/1 shape matrix of a protein, it is possible to search the protein space and fill it with progressively smaller, non overlapping spheres and higher resolution. This algorithm has been coded in the `3DSearch.nb` program (appendix A.1.2). It takes the shape matrix and starting radius as input parameters and returns sphere models with progressively smaller spheres (up to 1 Å in radius). Fig. 4.3 shows the example of a 2D search of the spheres with two different starting radii, from the simulated 2D shape matrix.

4.7.3 Theoretical scattering curve calculations

The theoretical scattering curve of the sphere model FUs was calculated using a modified, faster to compute Debye formula (Debye, 1915):

$$I_M(q) = n\phi^2(qr)\left[1 + \frac{2}{n}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{\sin\left(qd_{ij}\right)}{qd_{ij}}\right] \tag{4.1}$$

where r is the radius of the spheres, q is the scattering vector length, d_{ij} is the distance between spheres i and j and $\phi^2(t) = 9\left(\frac{\sin t - t\cos t}{t^3}\right)^2$ with $t = qr$. In the case of models with spheres of different radii, the equation used (Spinozzi,



(a)

(b)

(c)

Figure 4.3: 2D spheres search example with different starting radii. (a) Starting 2D array; (b) spheres search result with starting radius of 10; (c) spheres search result with starting radius of 5.

personal communication) was:

$$I_M(q) = \sum_{i=1}^{n} \phi^2(qr_i) + 2\left[\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \phi^2(qr_i)\,\phi^2(qr_j)\,\frac{\sin(qd_{ij})}{qd_{ij}}\right] \tag{4.2}$$

4.7.4 Cost function calculations

The degree of difference between two scattering functions, the cost function (CF), represented by the χ^2 was calculated as (Press *et al.*, 1992):

$$\chi^2 = \sum_i \left(\frac{I_{exp}(q_i) - a - bI_{calc}(q_i)}{\sigma_{I_{exp}}(q_i)}\right)^2 \tag{4.3}$$

where I_{exp} is the experimental scattering intensity, I_{calc} is the theoretical scattering curve, $\sigma_{I_{exp}}(q_i)$ is the error of the experimental curve at scattering angle q and a and b are least squares fit scale parameters of the two curves (background and scale factors).

If the experimental curve does not have error values associated to the intensities or when comparing two theoretical curves, a modified CF equation is used:

$$\chi^2 = \sum_i \Big(I_{exp}(q_i) - a - bI_{calc}(q_i)\Big)^2 \tag{4.4}$$

The procedure returns the χ^2 value as well as the coefficients a and b with their uncertainties σ_a and σ_b and the "goodness of fit" value.

4.7.5 Volume scale fitting

The best fit scale of the sphere model was calculated by changing the size of the model from 0.2 V_0 to 8.0 V_0 in steps of 0.2 (with V_0 representing the original sphere model volume) and comparing its scattering curve to that of the experimental crystallographic unit. The size factor that gave the model with the smallest χ^2 was chosen as the correct one (fig. 4.4).

This procedure is implemented in the program `DeltaVSearch.nb` (appendix A.1.3).

4.7.6 Oligomer modelling

Low resolution structures of native, oligomeric proteins were modelled from monomer sphere models and theoretical or experimental SAXS curves.

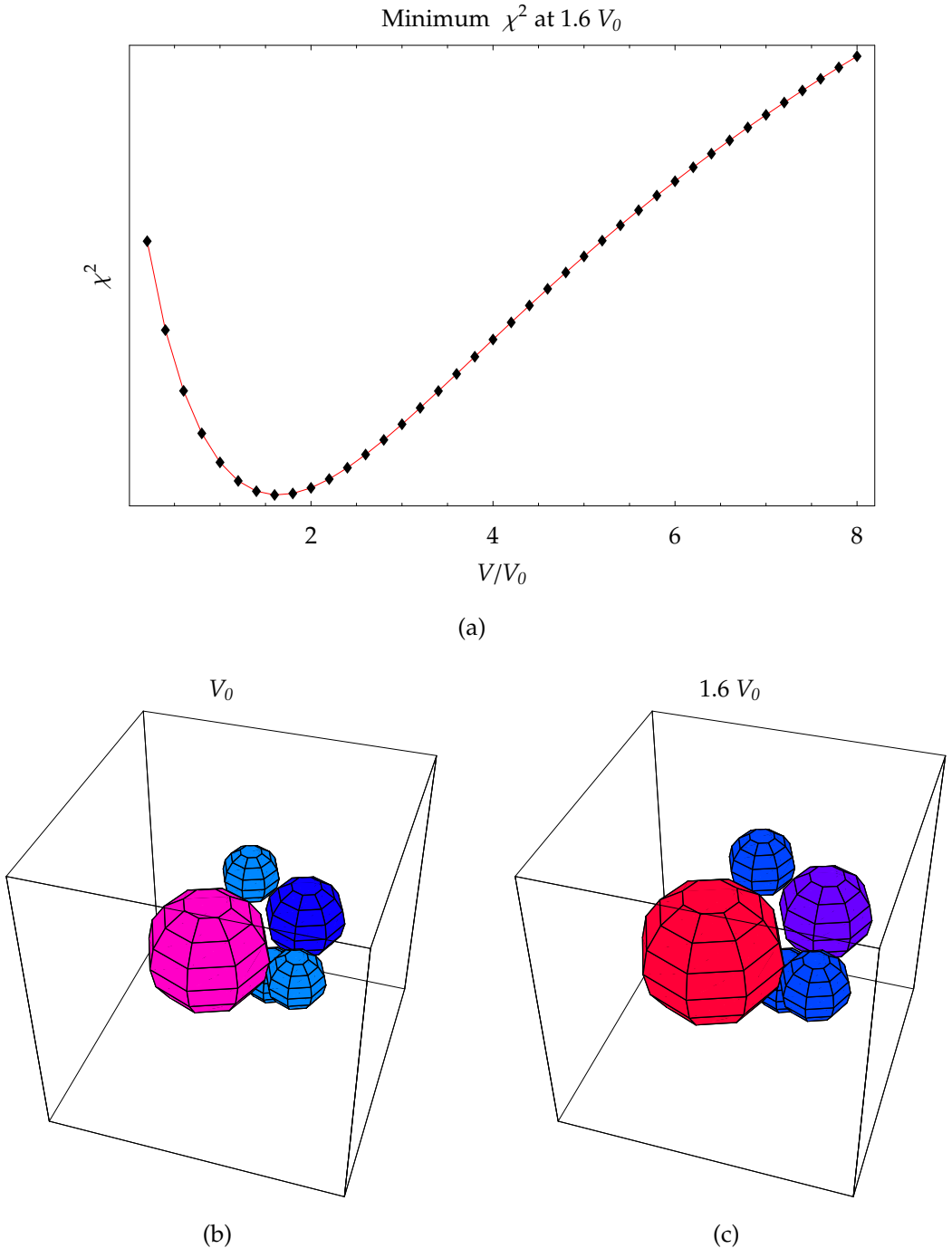


Figure 4.4: Volume best fit search of the sphere monomer (a) with initial (b) and fitted volume (c).

In order to obtain a satisfactory fit, an efficient method for the search of a global minimum of a multivariable function must be used. Since the number of parameters used in such a fit is quite large (from 7 to 97 in more complex models), the simulated annealing (SA) method was chosen (Metropolis *et al.*, 1953; Kirkpatrick *et al.*, 1983). The algorithm is based on the Monte Carlo technique but with some modifications which makes it less prone to get trapped in local minima. The algorithm works by varying the free parameters of the starting model (such as the position, the orientation and size of the SSs) generating a new model. The scattering curve of this new model is computed using the Debye formula and the difference between this curve and the experimental scattering curve of the protein is calculated. The new model becomes automatically accepted as the best model if the CF is lower than that of the previous configuration. If it has a higher CF, then it is accepted with probability $\exp\left(\frac{-\Delta CF}{T}\right)$. The probability of the acceptance of the two models follows the Boltzmann distribution, $\frac{P_1}{P_2} = \exp\left(\frac{CF_2 - CF_1}{T}\right)$ where T is the temperature of the distribution P. The modelling starts at high temperatures, accepting therefore many uphill moves and exploring the parameters space. The process is repeated and after some iterations the temperature is lowered according to the equation $T_{n+1} = T_n \times T_{red}$ where T_{red} is in the order of 0.9 to 0.99 for more precise fits. Fig. 4.5 shows the flow diagram of the implemented SA algorithm.

The main disadvantage of the SA algorithm is the difficulty to determine the fit parameters like initial and final temperature, the temperature decrease factor and others. Several trial runs are necessary for manual optimization of these parameters. Once the fit parameters were found, many runs were executed due to the stochastic nature of the SA algorithm and to make sure that the program converges to one solution. The results presented in this work represent the typical output of several runs of the program.

The fit is implemented in the Pascal program `idnsaxsfit.pas` (appendix B.1.1, fig. 4.6) and takes as input a sphere model of the FU, an experimental or theoretical scattering curve to fit to, a parameters file which specifies the starting parameters of the SS model (position, rotation and size of each FU), the SA algorithm parameters (starting and ending temperatures, the temperature decrease factor etc.) and the symmetry of the oligomeric molecule. In this version only D_n symmetries are implemented, with n ranging from 1 to 5,

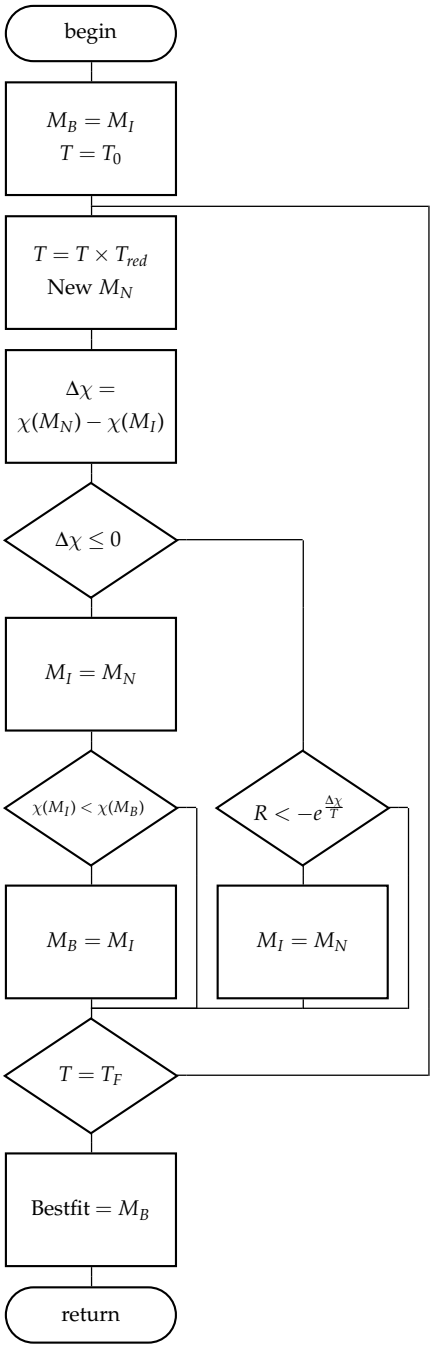


Figure 4.5: Flow diagram of the implemented SA algorithm. (M_B) best model, (M_I) initial model, (M_N) new model, (T) temperature factor, (T_0) initial temperature, (T_f) final temperature, (T_{red}) temperature decrease factor, (χ) weight of a model, ($\Delta\chi$) difference between model weights, (R) random number in [0,1] range.

compatible with those of molluscan and arthropodan hemocyanins.

The fitting program utilizes some external public domain programs during run-time. These are RASMOL (Sayle & Milner-White, 1995), for the interactive drawing of the sphere models, and GNUPLOT (Williams & Kelley, 2005), for the drawing of SAXS curves from the models. The communication between these programs is carried out with the help of `gnuplotpipe` and `rasmolpipe` programs written in C (gcc 3.3 20030304, Free Software Foundation (1991a)) with listings in appendix C.

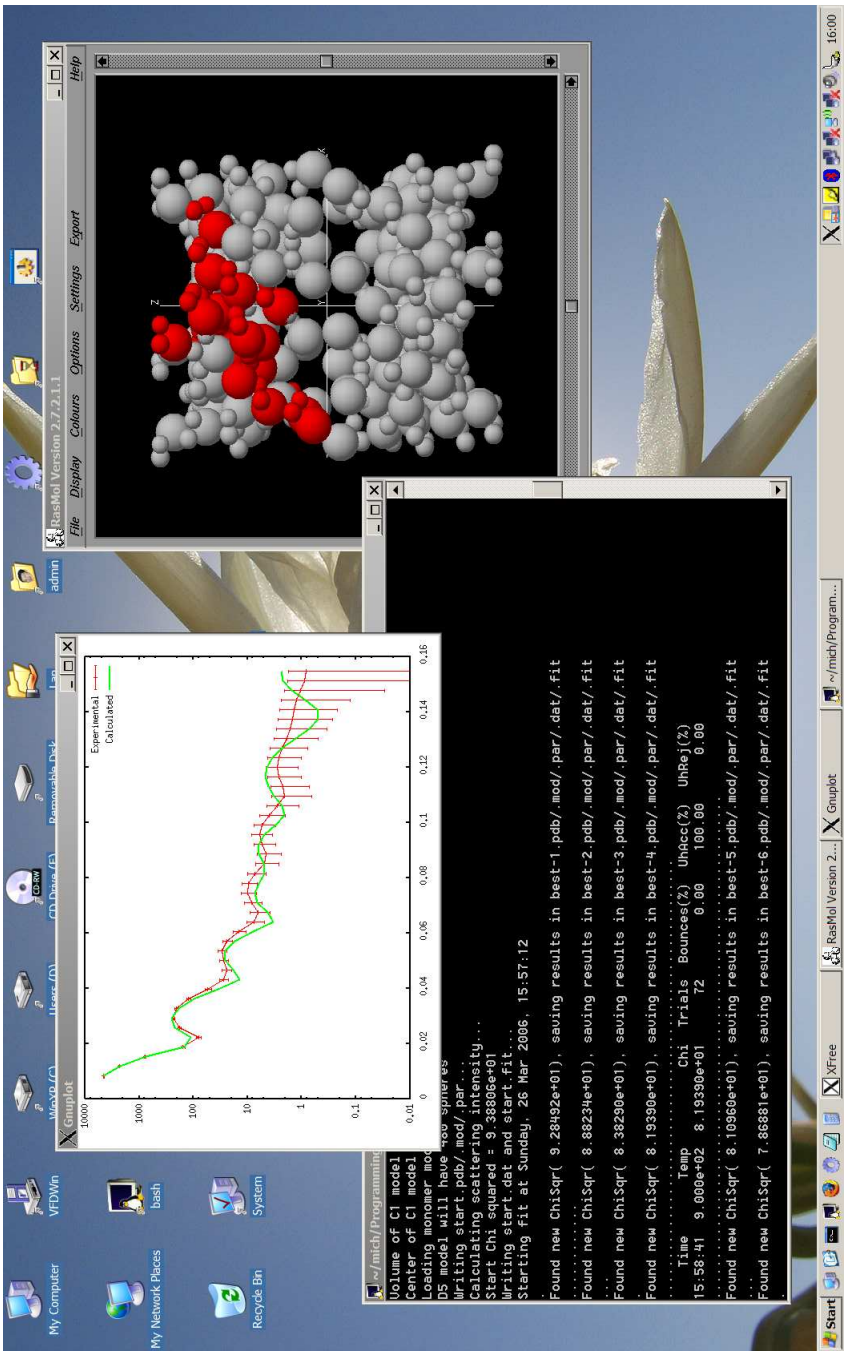


Figure 4.6: Screenshot of the fitting program `idnsaxsfit.pas` (bottom) and auxiliary programs GNUPLOT and RASMOL (top) for data visualization running on Intel/Microsoft Windows with Cygwin platform.

5 Results

5.1 Protein crystallography

Out of 600 crystallization trials made with the robot, only a couple of hits in few months were observed. They consisted of a microcrystalline precipitate as indicated by weak birefringence under an optical microscope. After manual optimization of crystallization conditions, crystals of various morphology have been obtained with scarce reproducibility (fig. 5.1). Most are rod-like and only few are hexagonal. They are very small (with longest edge of 20–50 μm), thin, very fragile and difficult to manipulate. Despite considerable effort was spent finding stabilizing solutions, they would crush as soon as they are taken out from the mother liquor.

The copper presence was proved by a fluorescence scan around the copper K edge, recorded on a single crystal (fig. 5.2).

The best data collection up to a resolution of 6–7 Å (fig. 5.3) has a completeness of 40% due to crystal decay form radiation damage. Indexing procedures have been attempted with programs MOSFLM and DENZO (Leslie, 1992; Otwinowski & Minor, 1997). Both programs failed indexing such a large unit cell. Consistently, *a* and *b* axes are about 300 Å while the longest *c* axis could not be estimated. Such a large unit cell is compatible with the whole hemocyanin molecule.

5.2 Electron and atomic force microscopy

Every mica foil with the deposited protein sample was cut in two parts. One part was used to transfer the protein monolayer to a EM copper grid and the other part was imaged directly with the AFM. Only samples that gave positive results to the presence of monolayers with the EM were imaged with

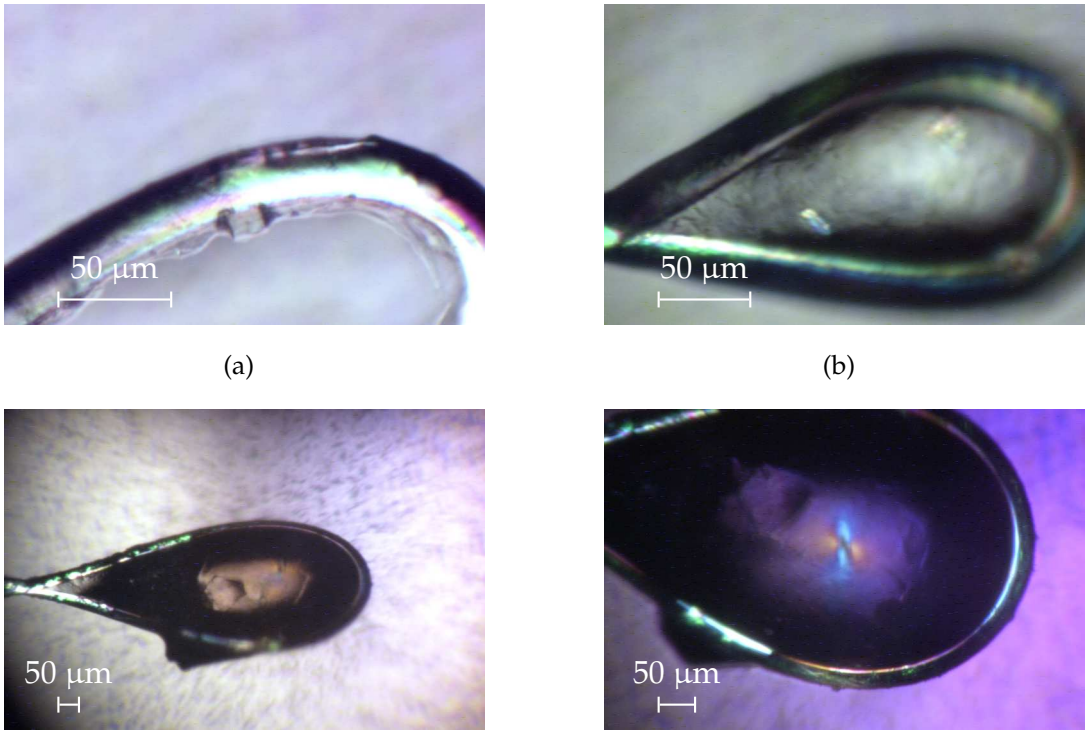


Figure 5.1: Example gallery of *R. venosa* protein crystals obtained. Crystals of different size and morphology can be seen.

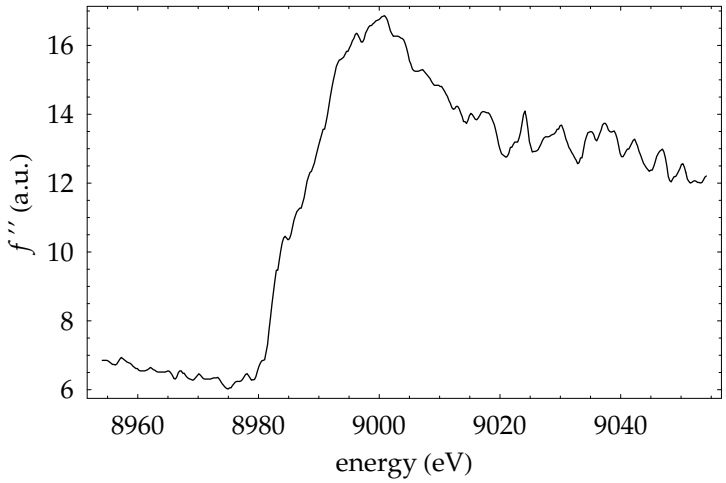


Figure 5.2: Copper K edge fluorescence scan from single crystal.

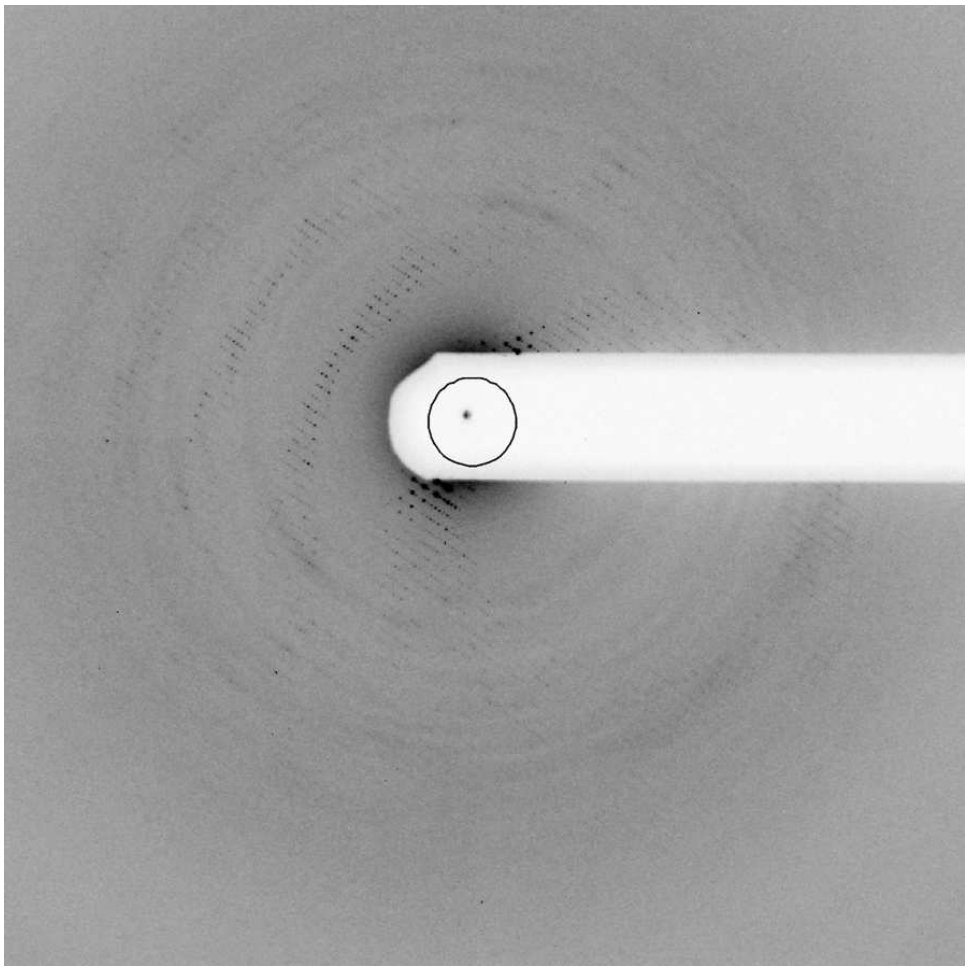


Figure 5.3: Sample diffraction pattern from *R. venosa* protein crystal. The crystals diffracted up to 6–7 Å resolution.

the AFM.

Monolayers of whole *R. venosa* Hcs have been observed with the EM and negative staining technique. Ordered domains of proteins have been found on the majority of mica samples transferred to copper grids. These domains present longitudinal and axial projections of whole Hc molecules (fig. 5.4).

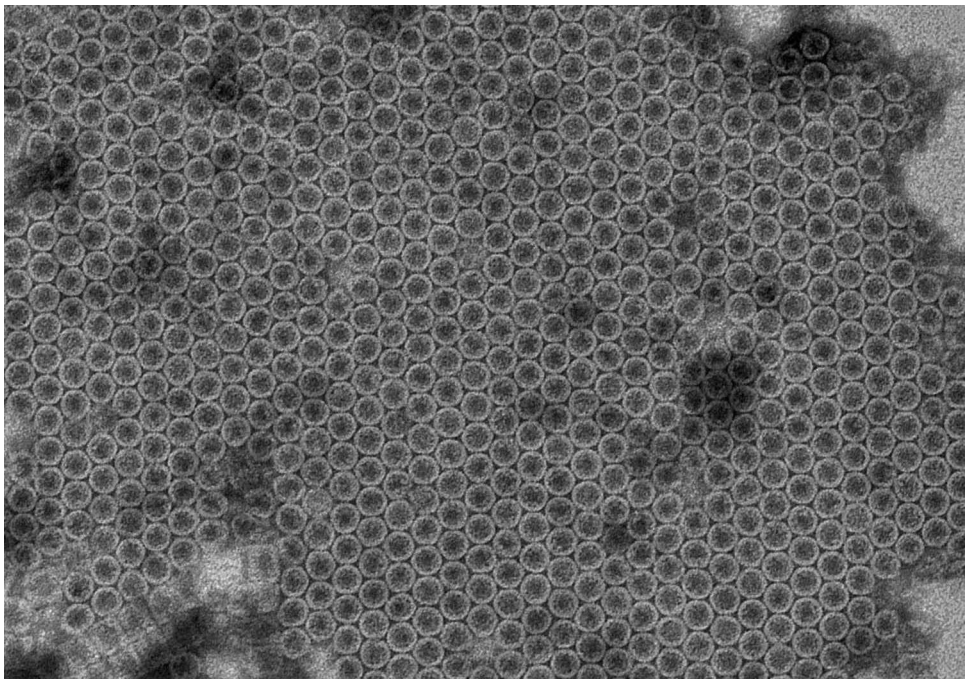
In contrast, it was quite difficult to find an ordered domain with the AFM. Several samples were examined, both in air and covered with protein buffer. Better conditions were found with scans on dry samples. Only in some cases ordered domains were observed with periodicity of ~ 350 Å (fig. 5.5). We were unable to find an ordered domain with a higher resolution tip.

5.3 SAXS measurements

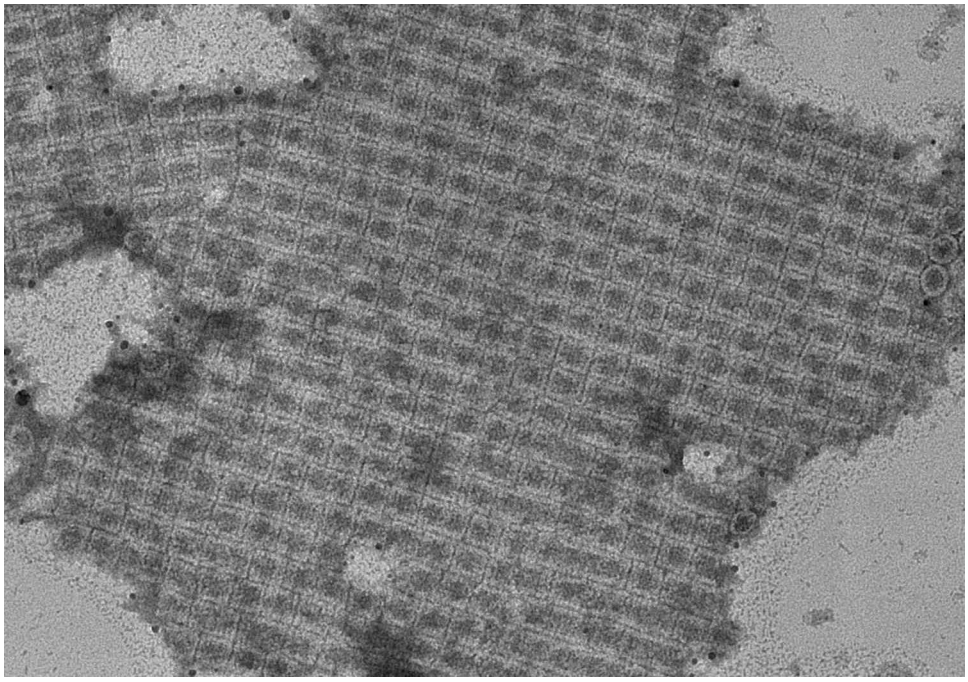
SAXS measurements were collected from whole molecules of *R. venosa* Hcs in native conditions and in the presence of cosmotropic (stabilizing, salting-out) salts PO_4^{3-} and F^- (fig. 5.6). In the present work, only the native protein (in Tris/Ca^{2+} buffer) SAXS curve were used for modelling since this condition is the most similar to the one used by cryo-EM electron density data. SAXS curves in the presence of stabilizing salts show small differences with the native curve meaning that the overall quaternary structure is preserved with some local differences. The structure of proteins in this conditions will be analyzed when there will be an accepted structural model of molluscan Hcs.

5.4 Arthropod Hc modelling

Arthropod Hc modelling consisted of the creation of a sphere model of a SS and fitting it to the calculated SAXS curve of the whole molecule. The SS was build from atomic coordinates of the *P. interruptus* Hc (PDB entry 1HCY) by a process outlined in section 4.7. The SAXS curve of a whole molecule (hexamer) was calculated with CRYSOL from PDB entries 1HC1–1HC6.

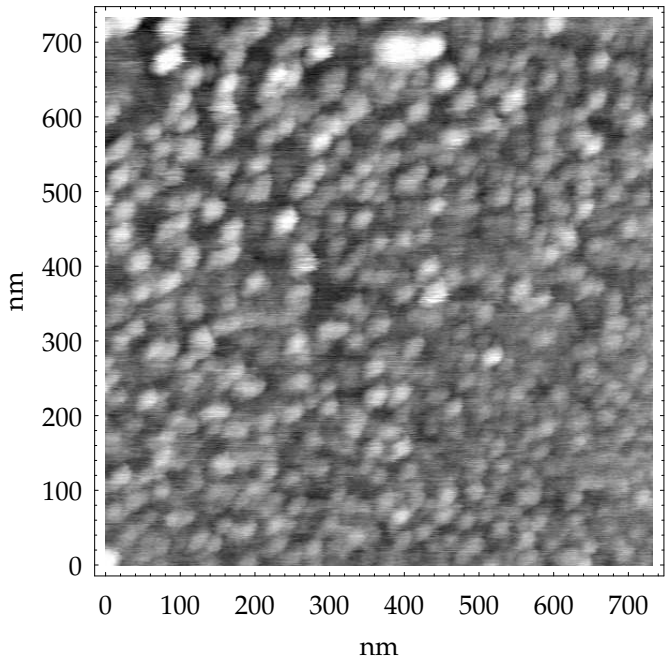


(a)

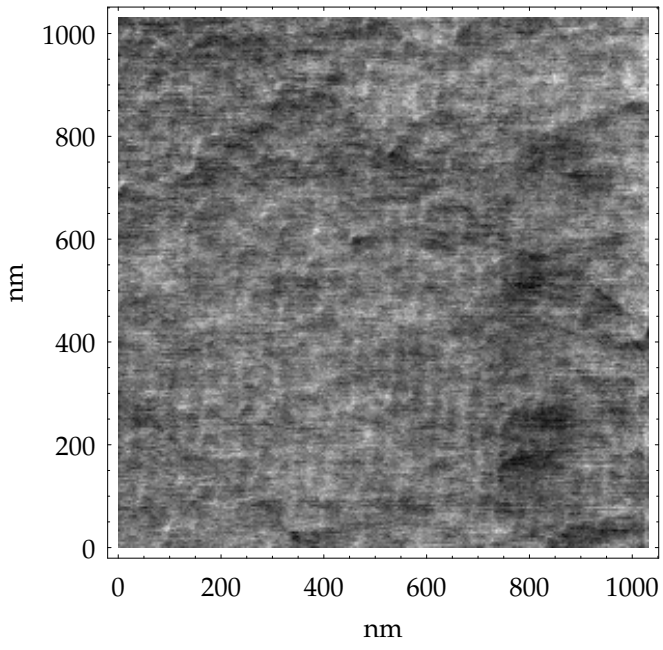


(b)

Figure 5.4: Axially (a) and longitudinally (b) arranged domains of *R. venosa* Hc monolayers.



(a)



(b)

Figure 5.5: AFM scans of *R. venosa* protein monolayers imaged in protein buffer (a) and in air (b).

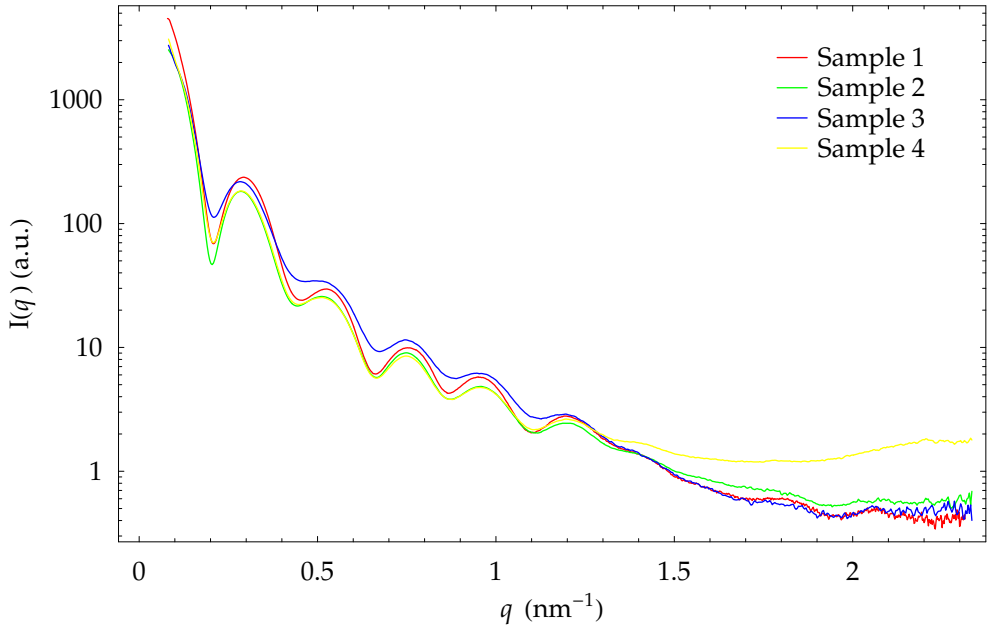


Figure 5.6: SAXS curves obtained from *R. venosa* protein samples in solution.

5.4.1 Sphere models generation

The first step involved the calculation of the envelope function of the protein, obtained with the program CRYSOL (fig. 5.7) followed by its transformation to a discrete mask (program `ReduceFromY.nb`). At this stage many sphere models were computed with different starting radii. The radii used were 21, 18, 15, 12, 10 and 8 Å. For every starting radius, different models were obtained by successive removal of progressively smaller spheres. The nomenclature used for the models is "m(starting radius, cut-off radius)" where cut-off radius is the radius of the biggest spheres removed from the models (fig. 5.8). The theoretical scattering functions from each sphere model were calculated using the modified Debye formula (eq. 4.2).

5.4.2 Best sphere model search

It was clear that models with few, large spheres were much smaller than the SS monomer. Therefore, a volume fit was attempted with each sphere model. The volume varied between 0.2 V_0 to 8.0 V_0 in steps of 0.2 (with V_0 representing the original sphere model volume).The volume scale fit results of all models

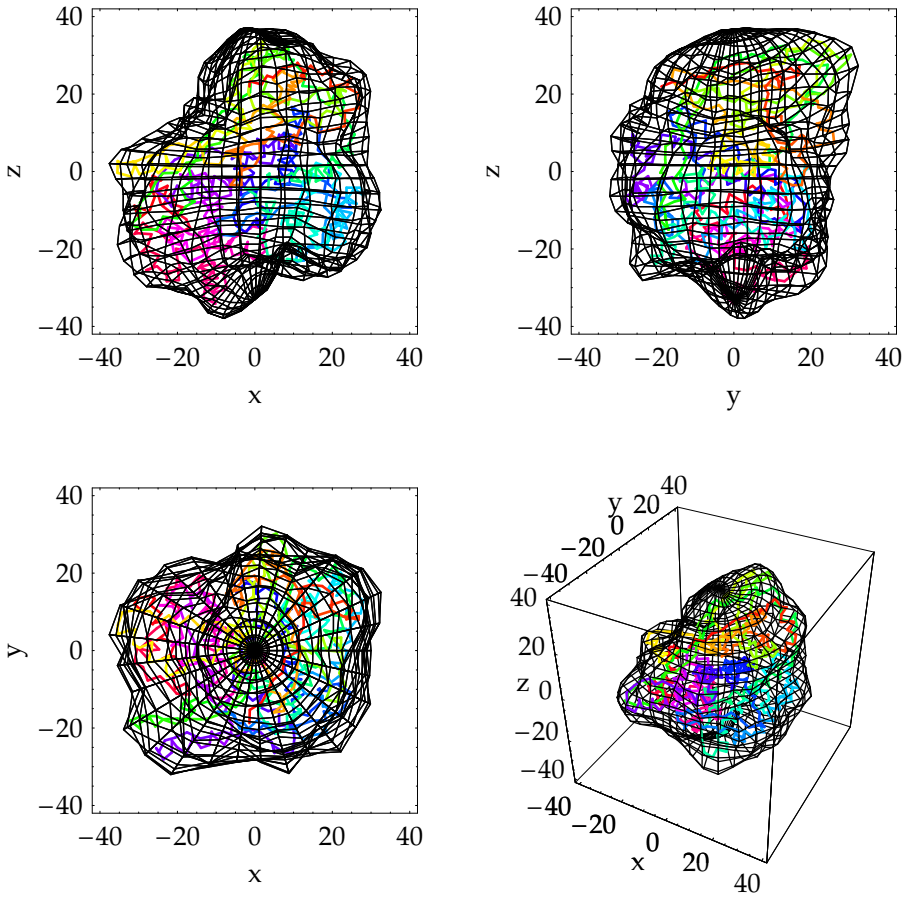


Figure 5.7: Envelope function of the *P. interruptus* SS shown with the C_α chain.

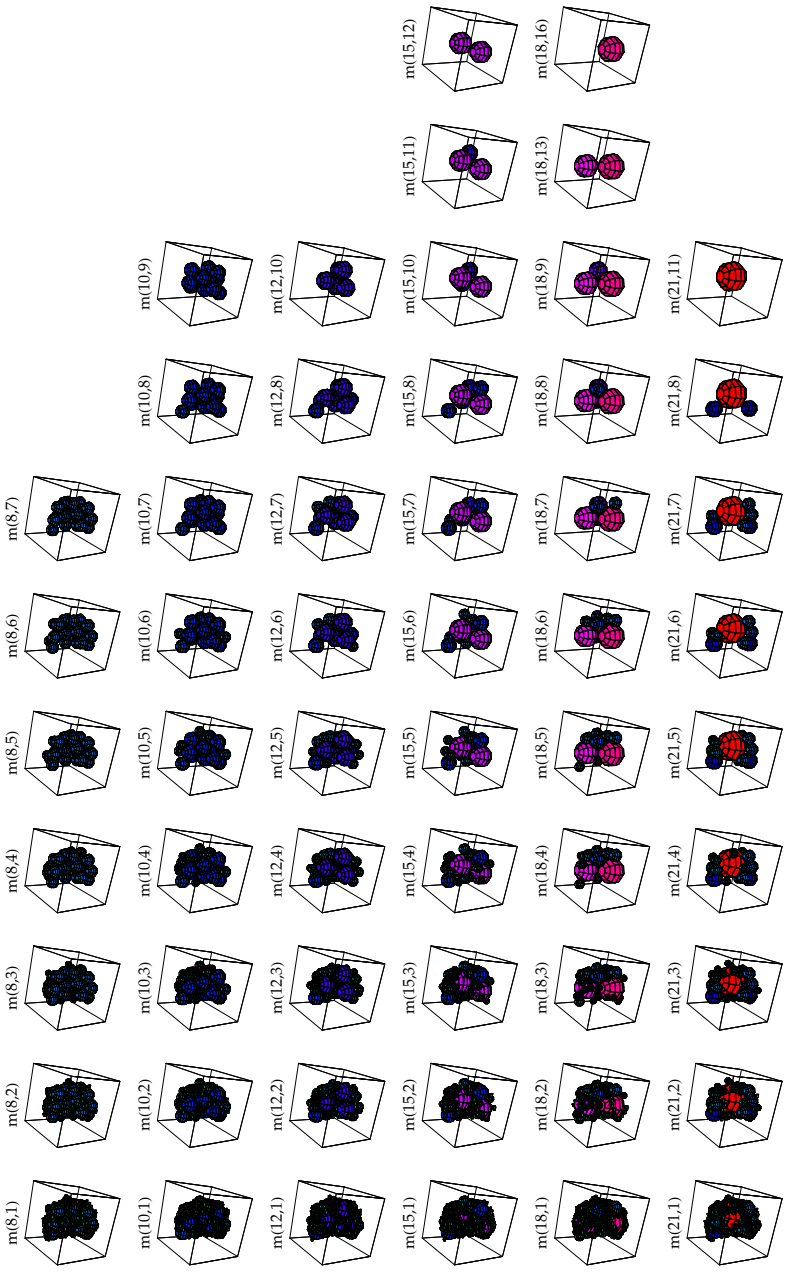


Figure 5.8: Sphere models of the *P. interruptus* SS with different starting radii.

are shown in fig. 5.9 and the final, size corrected models are shown in fig. 5.10.

The model used for the following Debye method simulations had to be composed of a reasonably small number of spheres (less than 15–20, not computationally intensive) and had to produce a similar scattering curve to the monomer as calculated from all atomic coordinates. Fig. 5.11 shows the model complexity (expressed as number of spheres) against the precision of each model (expressed as χ^2 or the difference with the SAXS curve from the atomic coordinates). The smallest sphere in the model should also have a radius bigger than $\frac{\pi}{q_{max}}$ where q_{max} in the maximum scattering angle in the fitting curve.

5.4.3 Debye method simulation

Several sphere models have been tested and the model m(18,8) was chosen for Debye method simulation of a whole arthropod D_3 Hc molecule. The program `idnsaxsfit.pas` was used to model a hexameric protein by fitting the positions and size of the SS sphere model with the SAXS curve of the hexamer calculated from atomic coordinates. Figure 5.12 shows the starting and fitted or final D_3 models, as compared to the hexamer built from atomic coordinates (fig. 5.13). The SAXS curves of the starting and final models are shown in figure 5.14.

5.5 Molluscan Hc modelling

Since gastropod Hc SS are composed of 16 homologous FUs, a two step method was applied. In a first step, a sphere model of a single FU has been created from atomic coordinates of the *R. venosa* subunit "2*e*" (PDB entry 1LNL). Secondly the positions of each FU in the SS were computed from the electron density map of an analogous gastropod Hc *H. tuberculata* (Meissner *et al.*, 2000). The use of *H. tuberculata* electron density map the reconstruction of *R. venosa* Hc is validated since the amino acid sequence homology in these species is very high ($> 70\%$, Lieb *et al.* (2000)) which involves high structural homology.

Once a sphere model of a SS has been created, the whole D_5 molecule was fitted to *R. venosa* experimental SAXS curve by Debye method simulations. The simulations were performed with FUs composed of one and many sphere models.

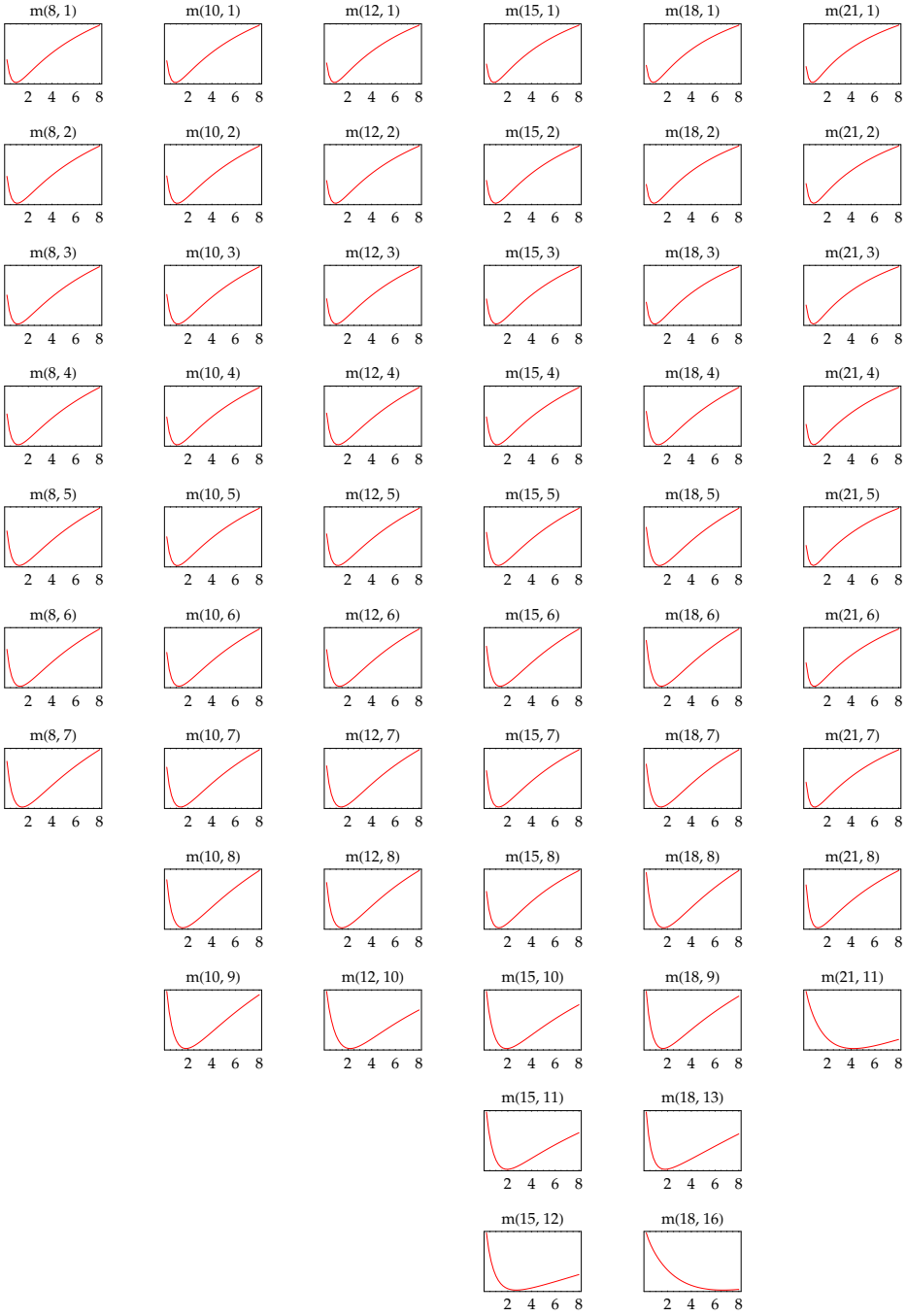


Figure 5.9: Volume fit results for each sphere model of the *P. interruptus* SS. The graphs show the volume increment factor $\left(\frac{V}{V_0}\right)$ versus χ^2.

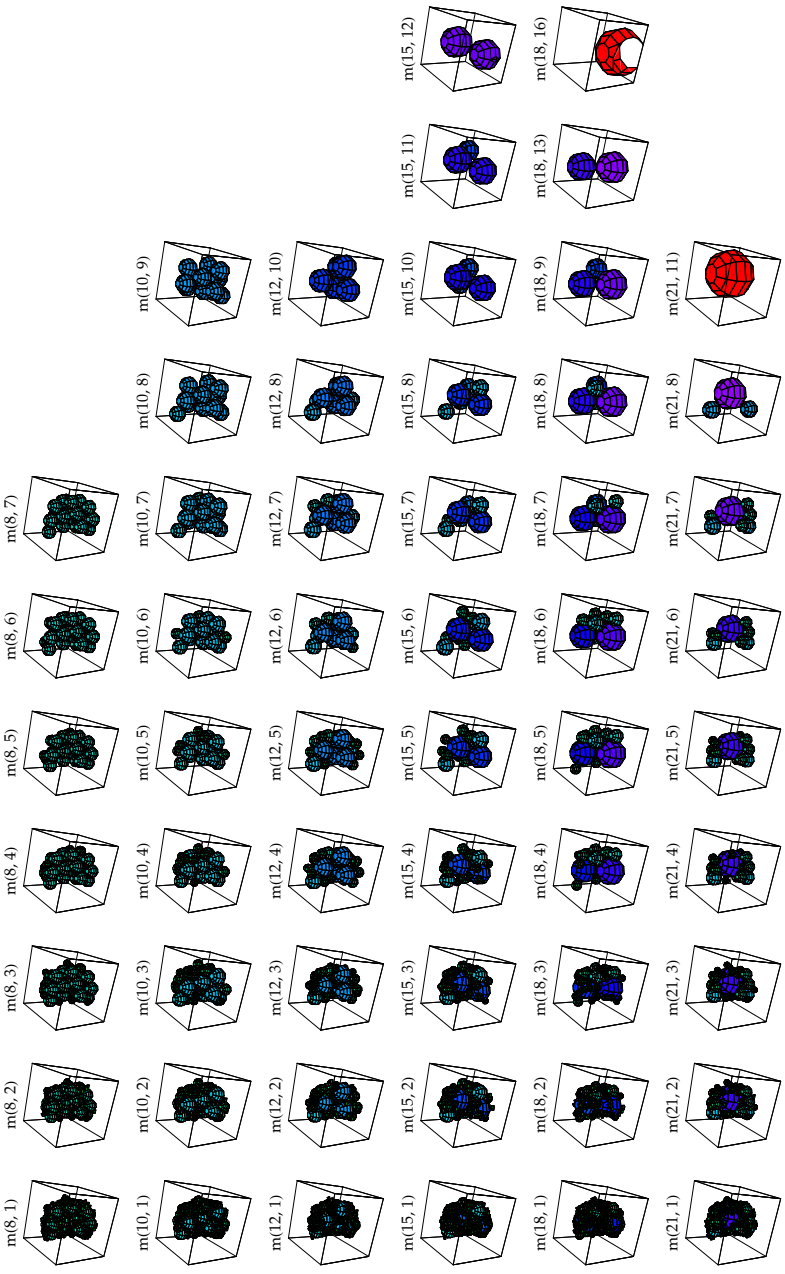


Figure 5.10: Size corrected sphere models of the *P. interruptus* SS with different starting radii.

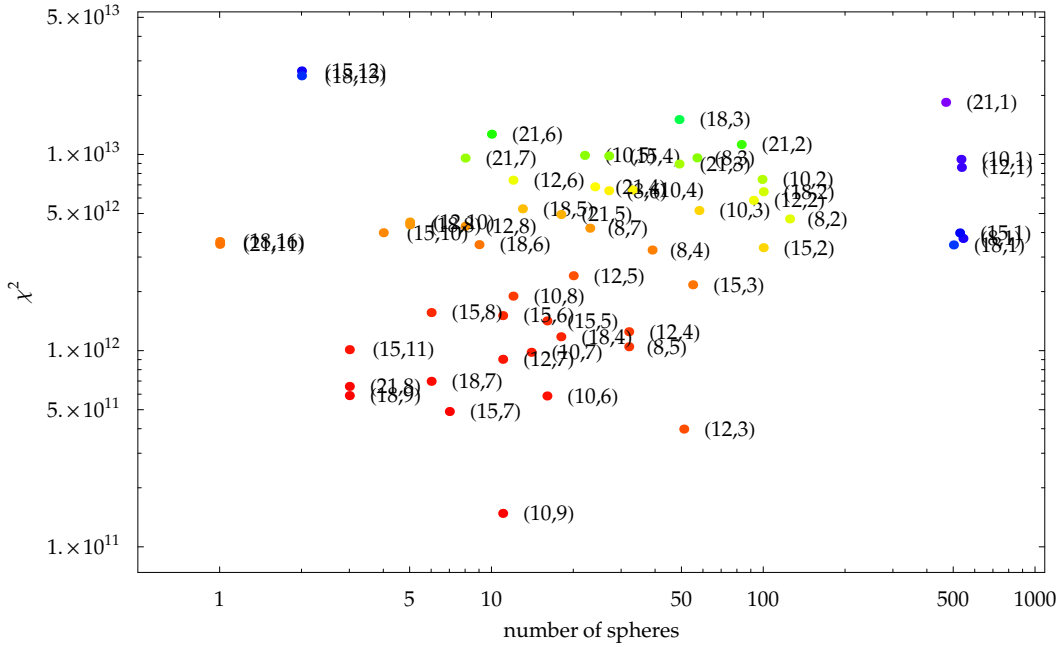


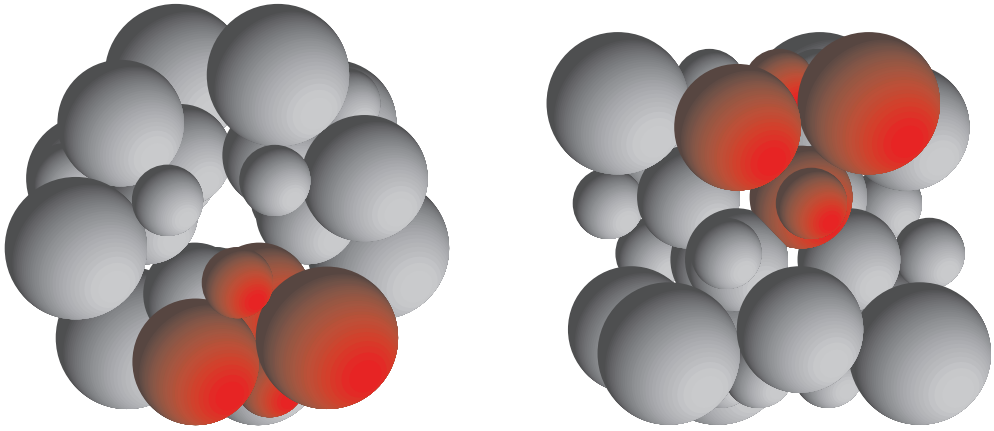
Figure 5.11: Model complexity (number of spheres per model) plotted against model precision (expressed as χ^2).

5.5.1 Sphere models generation

Sphere models were generated from protein envelopes in an analogous way as in arthropod modelling (fig. 5.15). The final sphere models (with starting radii 19, 17, 15, 12, 10, and 8 Å), their volume correction curves and model precisions are showed in figures 5.16–5.18. Model m(19,8) and one-sphere models were used for the Debye method simulations.

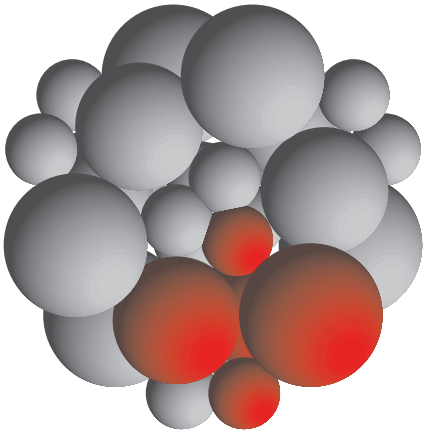
5.5.2 FU positions

The positions of the FUs inside the SS where computed from the electron density map of *H. tuberculata* Hc (fig. 3.10). Exploring the structural subunit with a spherical probe of 12.5 Å produced 16 separated lobes. The mean centers of each lobe gave the position of the FUs inside the SS (fig. 5.19). The whole molecule is constructed by applying the D_5 symmetry to the SS (fig. 5.20). This was the starting model for one sphere fits.

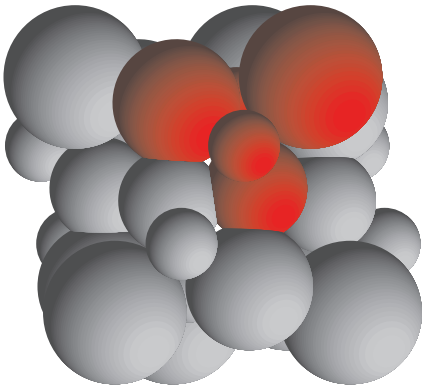


(a)

(b)

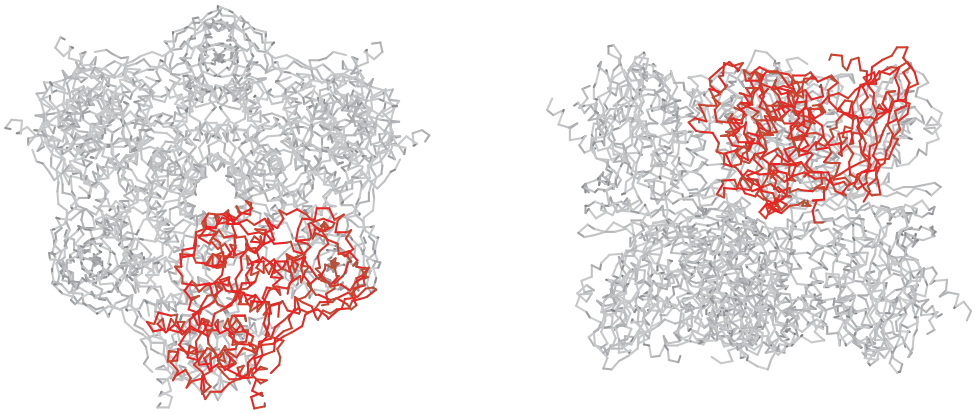


(c)

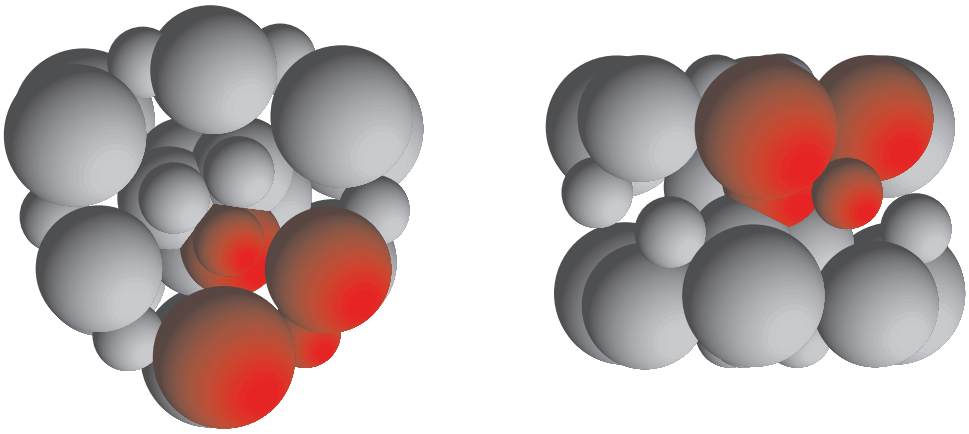


(d)

Figure 5.12: Axial and longitudinal projections of the starting model (a), (b) and of the fitted model (c), (d).



(a) (b)



(c) (d)

Figure 5.13: Axial and longitudinal projections of the crystallographic hexamer of *P. interruptus* Hc. Protein backbone (a)–(b) and sphere models (c)–(d) representation.

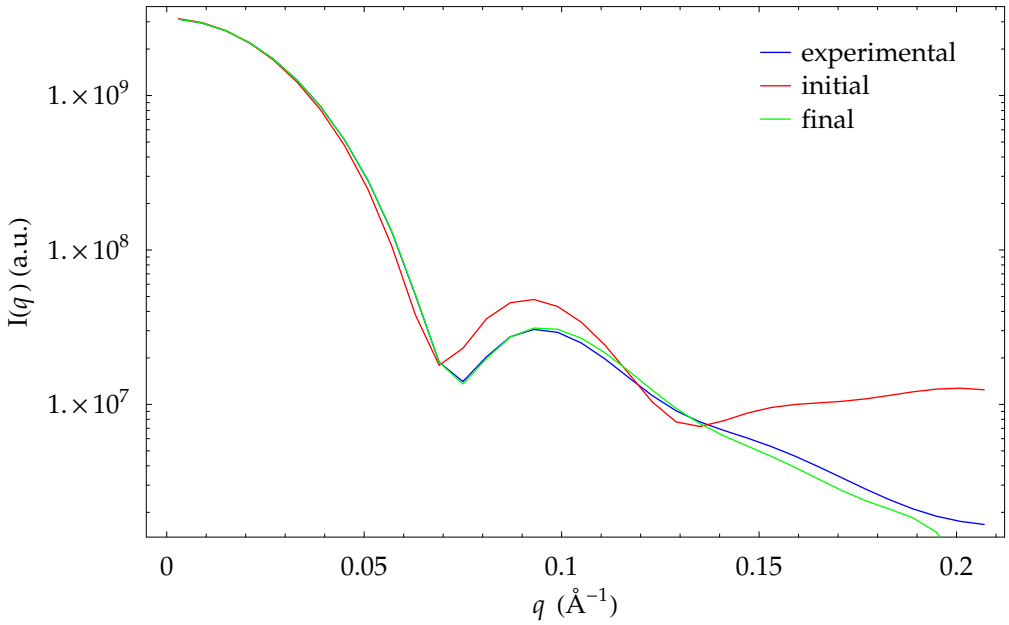


Figure 5.14: SAXS curves of the starting model, fitted model and the experimental fitting curve (obtained from atomic coordinates) of *P. interruptus* Hc.

5.5.3 Debye method simulation: one sphere FU

The fitting parameters of one sphere FU were the positions of each sphere and the radius of the spheres. The final model and its scattering function are showed in fig. 5.21 and 5.22. The fitted radius of the FU is 29.4 Å.

5.5.4 Debye method simulation: many spheres FU

The fitting parameters of many spheres FU were the positions and rotations of each FU in the SS and their volume. The initial and final models with their scattering functions are shown in fig. 5.23 and 5.24. The volume increase of the fitted model is 1.37 V_0.

All fits produced had the scattering function very similar to the experimental one but the models were different. Even modifying various fitting parameters (like enabling or disabling the sphere overlap), it was impossible to obtain a unique solution. The reasons for a non converging fit could be a simple monomer approximation or a high number of parameters.

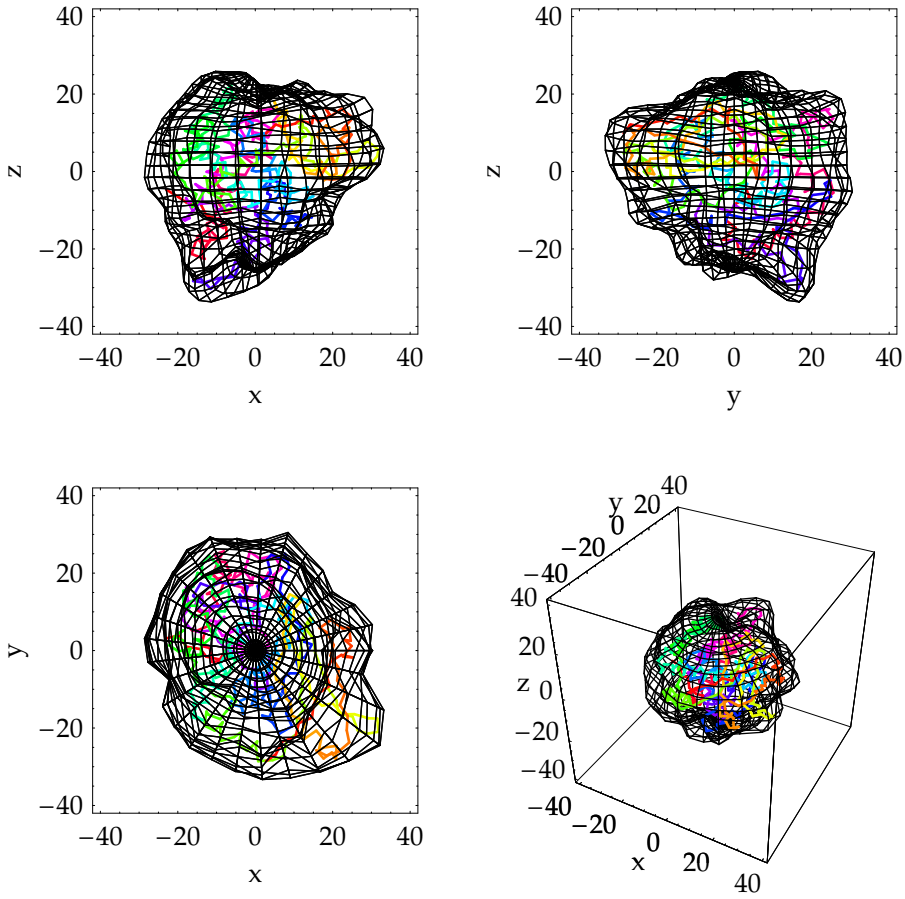


Figure 5.15: Envelope function of the *R. venosa* FU shown with the C_α chain.

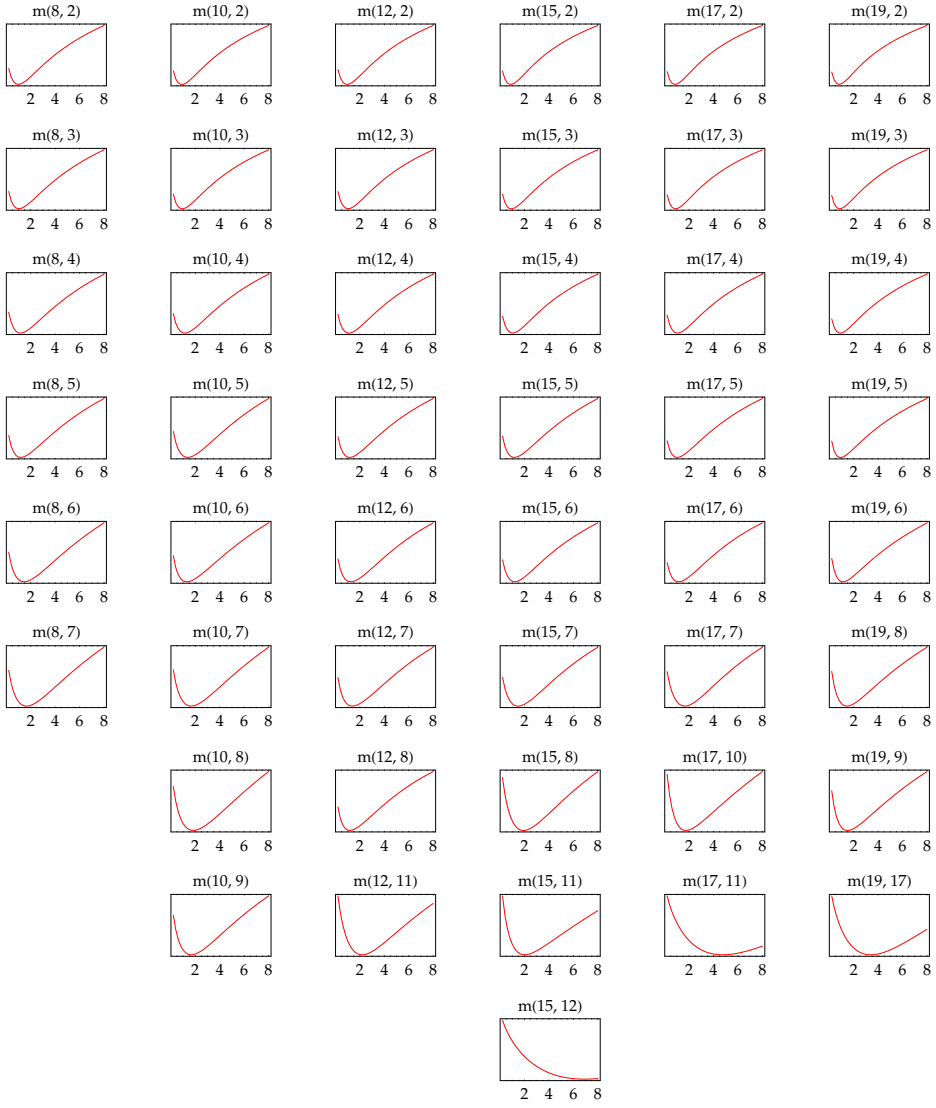


Figure 5.16: Volume fit results for each sphere model of the *R. venosa* FU. The graphs show the volume increment factor $\left(\frac{V}{V_0}\right)$ versus χ^2.

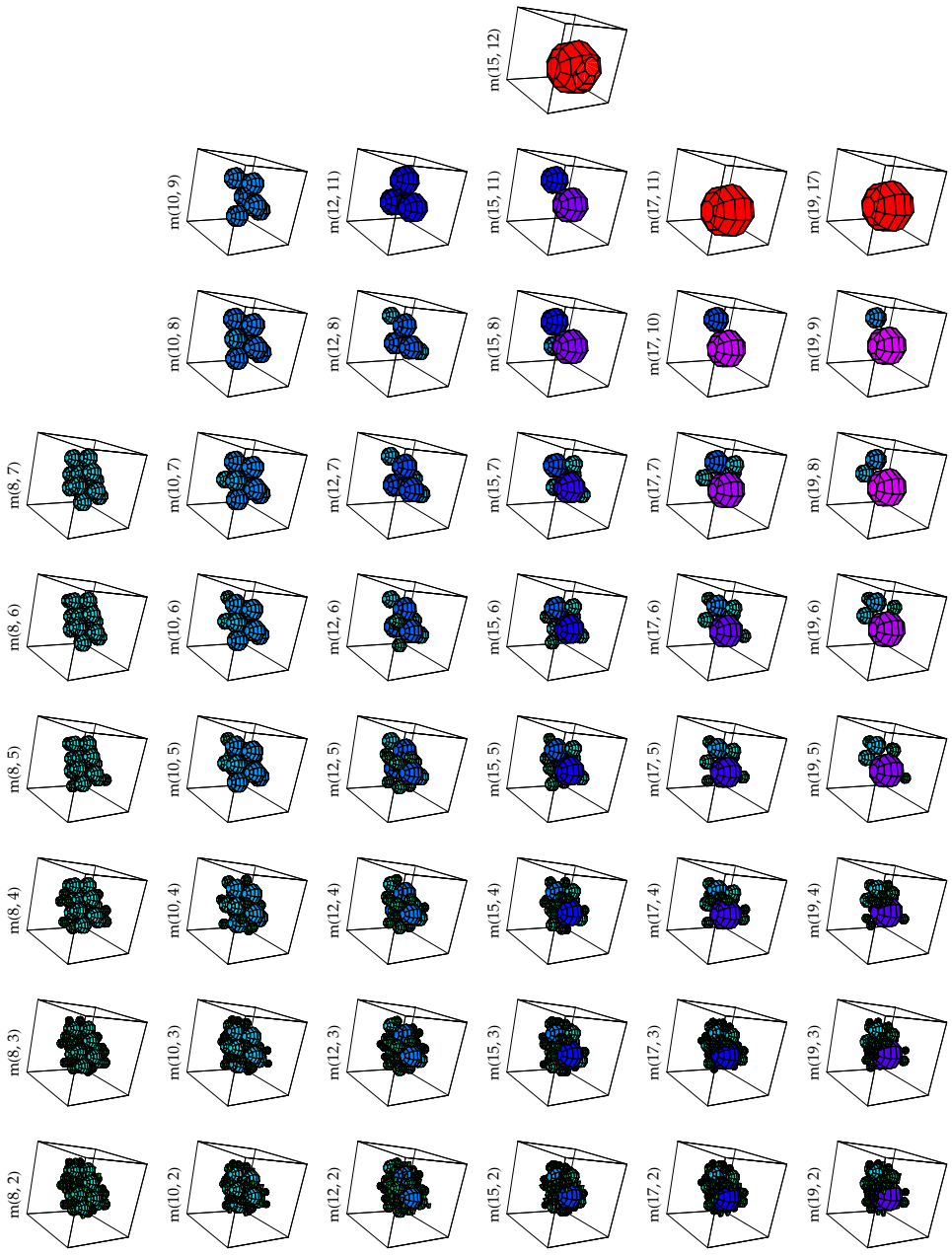


Figure 5.17: Size corrected sphere models of the *R. venosa* FU with different starting radii.

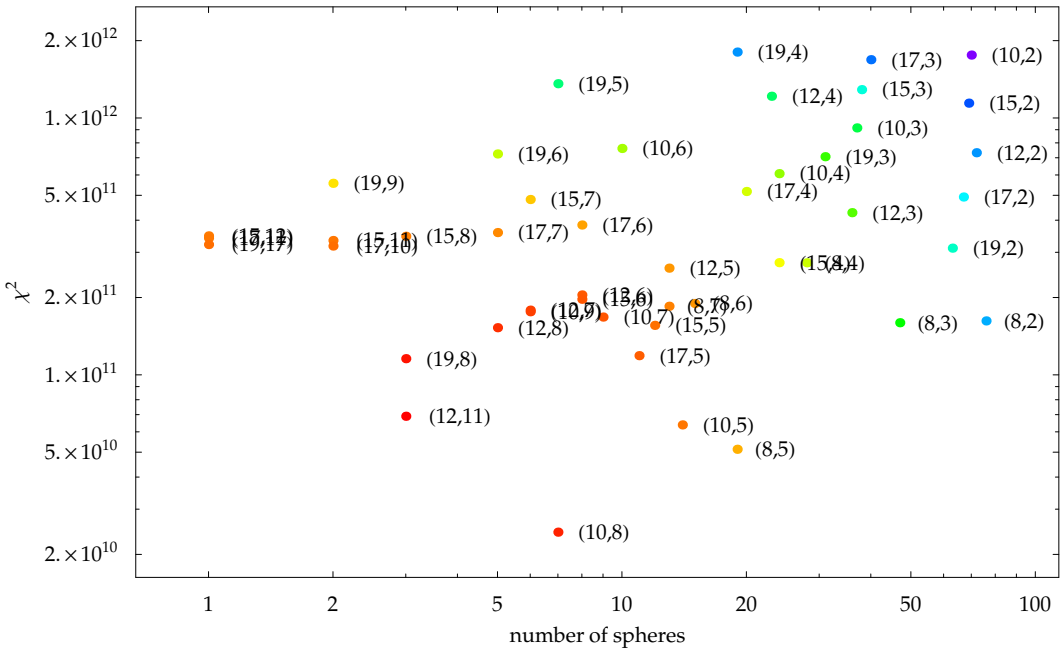


Figure 5.18: Model complexity (number of spheres per model) plotted against model precision (expressed as χ^2).

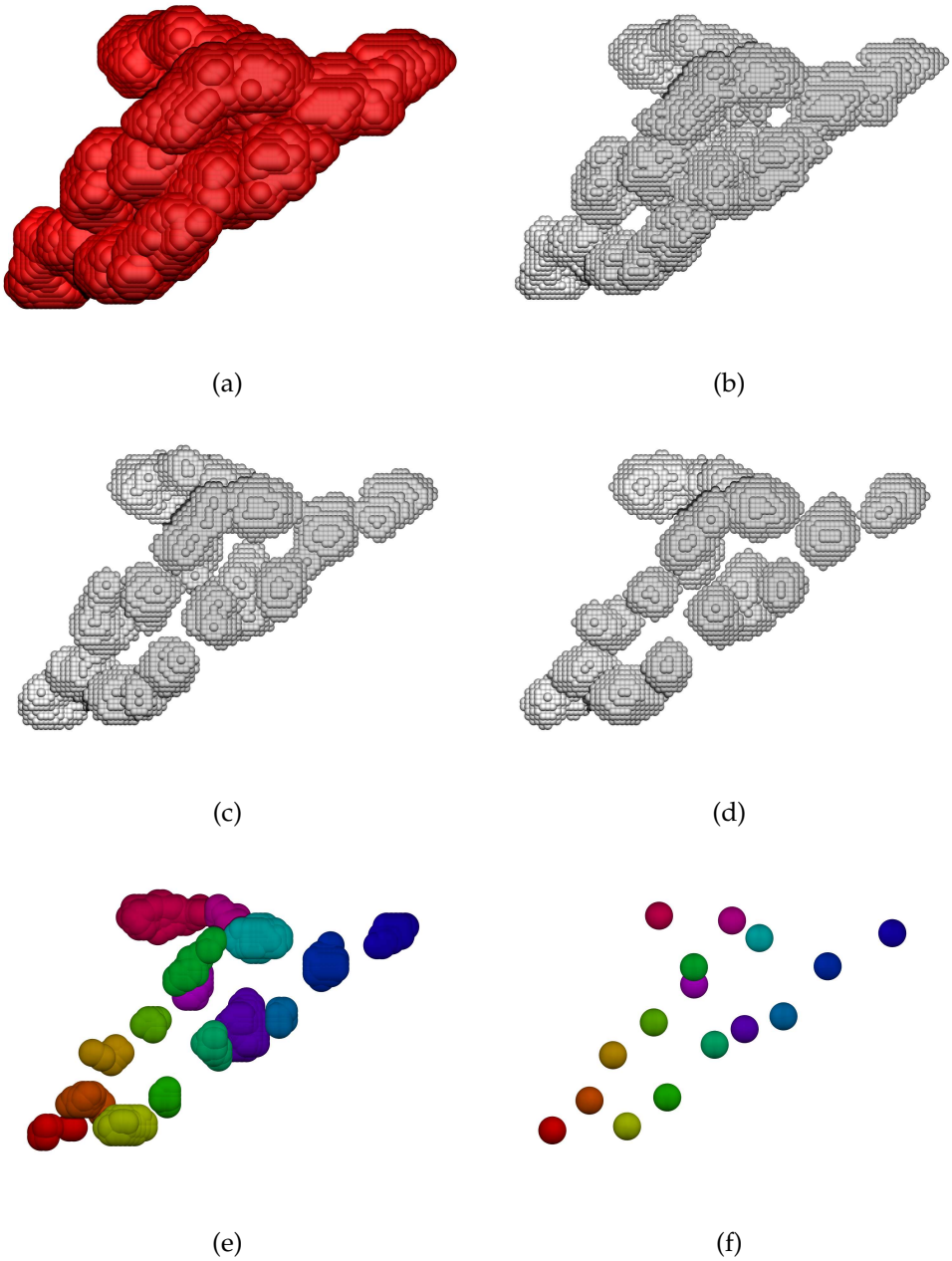
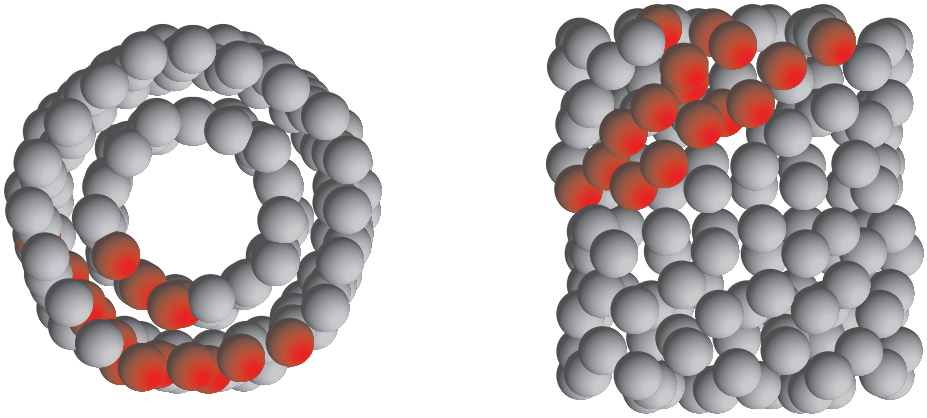
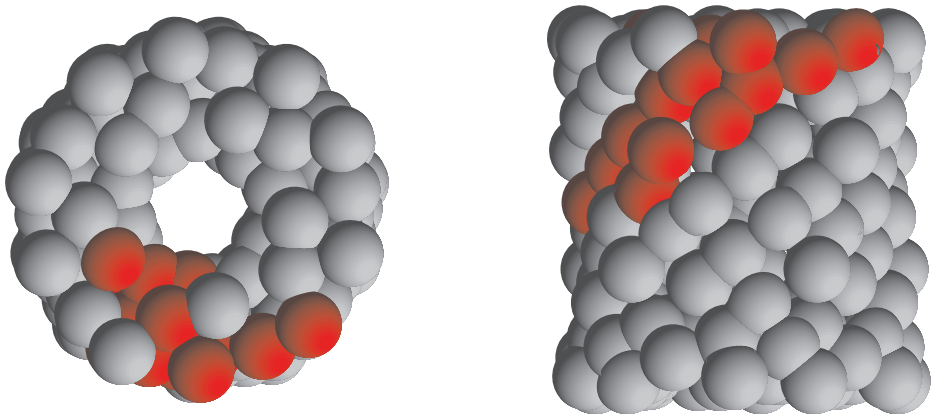


Figure 5.19: Calculations of the positions of FUs in *H. tuberculata* SS. The possible positions for spherical probes of 6.4 (b), 9.6 (c) and 12.8 Å radii (d) are shown. Only exploring the structure with 12.8 Å probe produced 16 distinct lobes corresponding to separated FUs. (e) Probe sphere centers in each lobe; (f) mean of probe sphere centers in each lobe.



(a) (b)

Figure 5.20: Axial (a) and longitudinal (b) projections of the starting model of *R. venosa* Hc.



(a) (b)

Figure 5.21: Axial (a) and longitudinal (b) projections of the fitted model of *R. venosa* Hc.

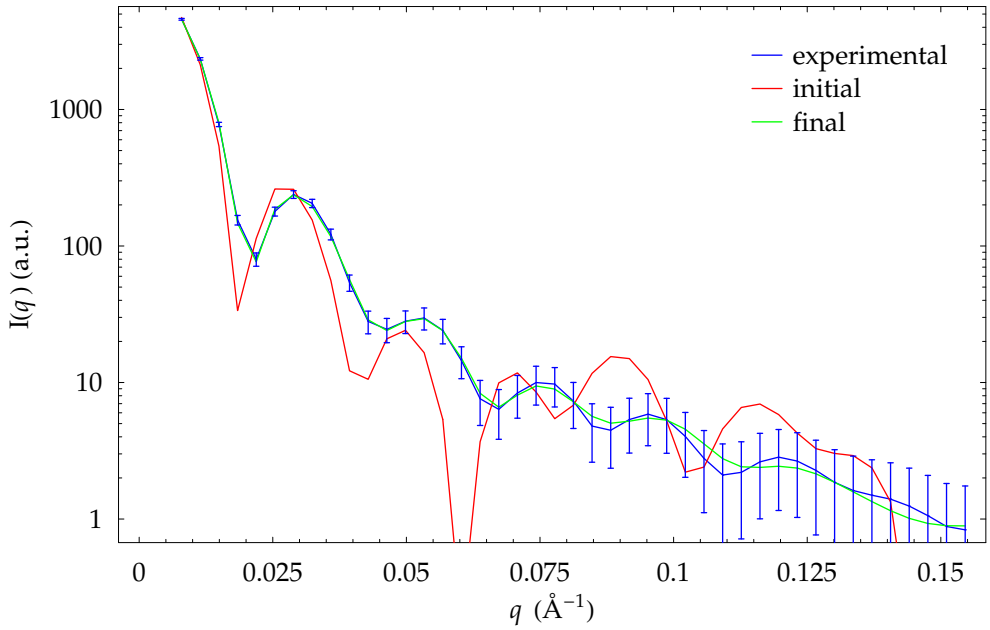
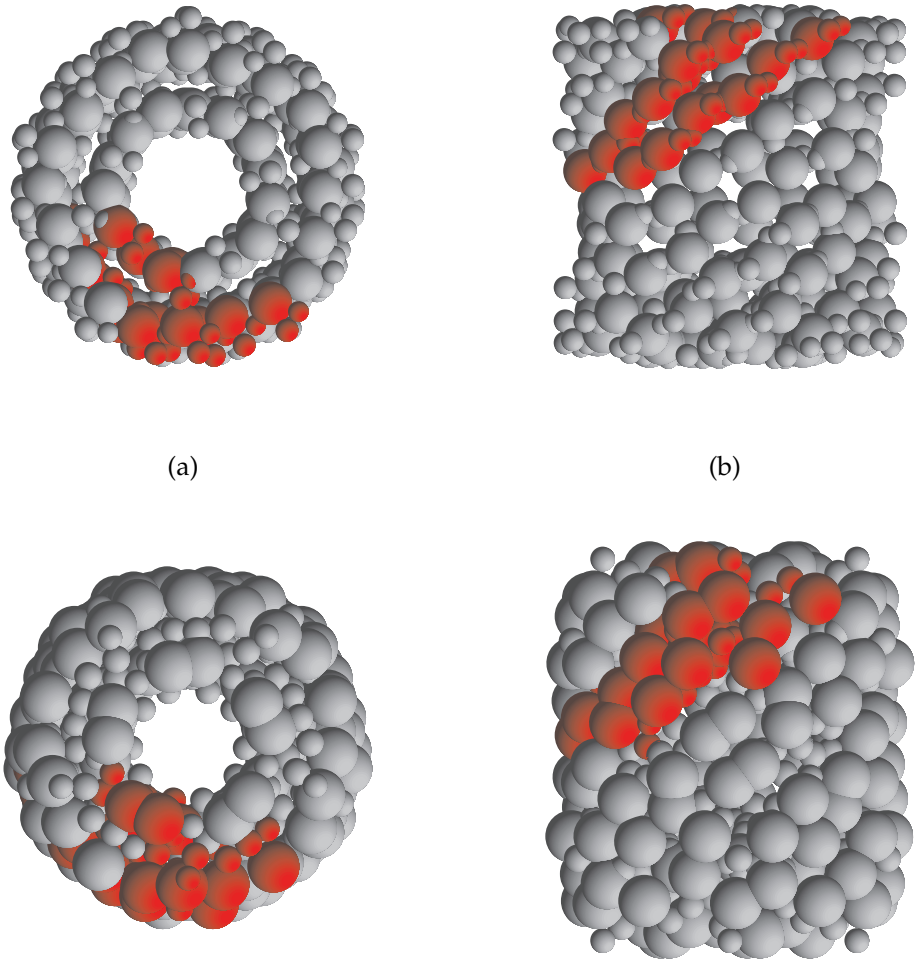


Figure 5.22: SAXS curves of the starting model, fitted model and the experimental fitting curve of one sphere FU *R. venosa* Hc.



(a) (b)

(c) (d)

Figure 5.23: Axial and longitudinal projections of the starting model (a), (b) and of the fitted model (c), (d).

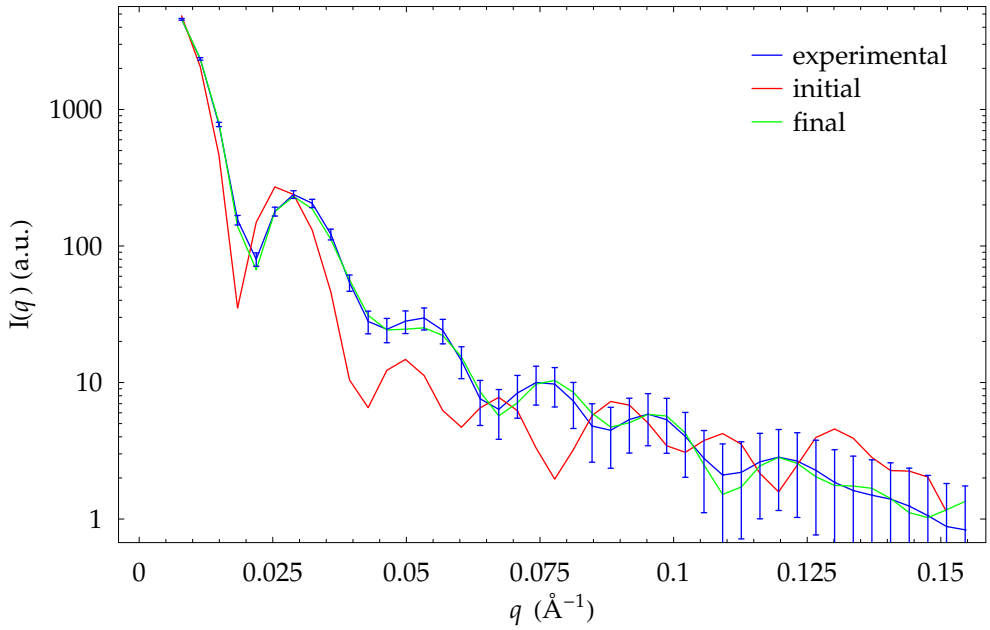


Figure 5.24: SAXS curves of the starting model, fitted model and the experimental fitting curve of many spheres FU *R. venosa* Hc.

6 Discussion

There are several methods and programs available for the determination of the quaternary structure of proteins in solution from SAXS data. These methods do not require any prior knowledge about the structure of the protein under exam and are optimized for the use of simple symmetries (i.e. P_1, P_2, P_3, P_4 and P_{222} as implemented in DAMMIN) and work very well for small particles or protein subunits. More complex, oligomeric proteins with D_n type symmetries like Hc, must be analyzed with these programs in the same way as asymmetric particles and produce less precise results with much more computing time.

To overcome this limit in the analysis of Hc structure, a different approach was attempted. The SAXS data analysis is not calculated *ab initio* but is build from simplified starting models obtained from other methods like (cryo-)EM with image analysis, protein crystallography and other biochemical and biophysical methods. The subunits of the starting model are represented as an assembly of spheres of different radii. In this way, not only the positions of the subunits, but also their orientations can be obtained. This method should be more precise than methods that use spheres with constant radius and much less computationally intensive than methods that utilize all atom representations. In addition, the implementation of the exact symmetry in the calculation procedures results in much more efficient algorithms that in turn produce more accurate results in the same amount of time (or, alternatively, much faster execution with the same accuracy).

To increase the precision of starting models, whole molecule protein crystallography and AFM measurements were attempted but up to now provided little extra information while cryo-EM particle reconstruction and protein crystallography of isolated subunits remain the most useful techniques.

6.1 Conclusions

This study describes a new method for quaternary structure determination of complex oligomeric proteins with the use of small angle X-ray scattering (SAXS) data. The method is very useful when:

- some structural information of the constituting monomer are known (i.e. low resolution shape, overall dimensions or high resolution structure, electron density maps etc.)

- the symmetry of the whole oligomer is known

- a SAXS curve of a whole oligomeric protein is known

Many oligomeric proteins are characterized in such mixed biophysical and biochemical way. One example of such proteins are hemocyanins, giant oxygen binding proteins found in several species of invertebrates.

The method involves two phases: first, a sphere model (with different radii spheres) of the structural subunit is build from electron microscopy or high resolution data and, second, a modelling program is coded to fit the positions and orientations of the structural subunits in the whole molecule to its experimental SAXS curve.

This method was successfully applied to arthropodan hemocyanins. The monomers of the protein examined were described as being composed of spheres with different radii and whole oligomeric protein was reconstructed by SAXS data modelling with custom made programs. The reconstructed arthropod hemocyanin protein had the right dimensions and a very similar subunit position and orientation compared to the structure of the native protein obtained from X-ray diffraction. However, using a quasi-symmetric (in one or two axes) subunit sphere model, different yet analogous solutions were observed. This indicates that sphere models used in such an approach need to be highly asymmetrical, if possible across all three axes, if the goal is to find the exact orientation of the monomers in the oligomer.

The custom made modelling programs (with listings in appendix A–C) were created to work with any protein with D_n symmetry (like hemocyanins) but also other symmetries could be easily implemented. The program shows

at runtime the best models found so far in interactive windows for immediate validation. It can be compiled with free compilers to executable code for any UNIX-like operating system like Linux, Apple Mac OS X and Microsoft Windows with Cygwin support.

The method was also tested with a much more complex molluscan hemocyanin protein. The results show that is possible to model a whole D_5 molecule in an accurate way only at low resolutions, with one sphere per functional unit. The trials with more than one sphere failed to converge to a unique solution. One possible explanation is the overall spherical and, when simplified to sphere models, symmetrical functional unit. Asymmetric models could be generated only with a high number of spheres per functional unit, increasing the computing time and lowering the precision of the fit.

The modelling of whole molluscan hemocyanins could be improved if the exact positions of functional units in the oligomer are known. Accordingly a whole molecule AFM and crystallography was attempted. The resolution of AFM scans of protein monolayers was lower than electron microscopy studies and was not used for modelling purposes. Protein crystals were obtained from highly purified whole protein samples and diffraction patterns were recorded with the completeness of 40% (at 6–7 Å). If sufficient diffracting crystals could be grown, the copper presence in the active site would allow multiwavelenght anomalous diffraction technique to determine the exact positions of the functional units in the molecule.

A Mathematica source codes

A.1 Model building

A.1.1 ReduceFromY.nb

```
<<Graphics`ParametricPlot3D`
<<Graphics`Graphics`
<<Calculus`VectorAnalysis`
<<Graphics`Graphics3D`
<<DiscreteMath`ComputationalGeometry`
<<Graphics`Shapes`

pdbfile = ReadList["1HCY_cleaned.pdb", String];\n
  pdbdatao = Table[{ToExpression[StringTake[pdbfile[[i]], {31, 38}]],
ToExpression[StringTake[pdbfile[[i]], {39, 46}]],
ToExpression[StringTake[pdbfile[[i]], {47, 54}]]}, {i, 1, Length[pdbfile]}];\n
  mo = Mean[pdbdatao];
  data = Import["1HCY00.txt", "Table"];
  n = 15;
  flmt = Table[1, {x, n + 1}, {y, 2  n + 1}];
  Dimensions[flmt]
  For[a = 1,
      a \[LessEqual] \[Sum]\+a = 1\%n + 1a, a++, \

l = data[[a, 1]];
m = data[[a, 2]];
c1 = data[[a, 3]];
c2 = data[[a, 4]];
flmt[[l + 1, m + n + 1]] = c1;
If[data[[a, 6]] \[Equal] 1,
        flmt[[l + 1, -m + n + 1]] = c2];
];
flm[l_, m_] := flmt[[l + 1, m + n + 1]];
  rey2[l_, m_, \[Theta]_, \[Phi]_] :=
      Chop[Which[m \[Equal] 0, SphericalHarmonicY[l, m, \[Theta], \[Phi]],
          m > 0, ty = SphericalHarmonicY[l, m, \[Theta], \[Phi]];
           (ty + Conjugate[ty])/Sqrt[2.0],
          m < 0, ty = SphericalHarmonicY[l, -m, \[Theta], \[Phi]];
           -\[ImaginaryI]*(ty - Conjugate[ty])/Sqrt[2.0]]];
  env2[\[Theta]_, \[Phi]_] := \[Sum]\+l = 0\%n(\[Sum]\+m = -l\
\%l(flm[l, m]*rey2[l, m, \[Theta], \[Phi]])));
fxy[x1_, x2_, y1_, y2_, y_] :=  x1 + (((x2 - x1)*(y - y1))/(y2 - y1));
r=29.466044+(2*1.611)
sphp=SphericalPlot3D[env2[theta,phi],{theta,0.0,Pi},{phi,0.0,2*Pi}];//Timing
Save["sphp.txt",sphp];
dcell=1;
clen=40;
ybox=Table[0,{x,-clen,clen,dcell},{y,-clen,clen,dcell},{z,-clen,clen,dcell}];
Dimensions[ybox]
blen=Length[ybox];
DeleteFile[{"ybox.txt","ybox_final.txt"}];
For[x=1,x\[LessEqual]blen,x++,
      Print[x,"/",blen];
```

```
49        For[y=1,y\[LessEqual]blen,y++,
50          For[z=1,z\[LessEqual]blen,z++,
51              rc=N[fxy[-clen,clen,1,blen,{x,y,z}]];
52              sphc=N[CoordinatesFromCartesian[rc,Spherical]];
53              If[r*env2[sphc[[2]],sphc[[3]]]\[GreaterEqual]sphc[[1]],
54                ybox[[x,y,z]]=1];
55              ];
56          ];
57        DeleteFile["ybox.txt"];
58        Save["ybox.txt",ybox];
59        Print["Saved ybox(",x,")"];
60        ];//Timing
61  Save["ybox_final.txt",ybox];
```

A.1.2 3DSearch.nb

```
1   <<Graphics`Shapes`
2   <<"ybox_final_40_1_81_81_81.txt";
3   data=ybox;
4   Dimensions[data]
5   mx=Dimensions[data][[1]];
6   my=Dimensions[data][[2]];
7   mz=Dimensions[data][[3]];
8   fdir="/";
9   ndata=data;
10  clist={};
11  startr=8;
12  endr=1;
13  deltar=1;
14  DeleteFile[fdir<>"clist.txt"];
15  For[r=startr,r\[GreaterEqual]endr,r=r-deltar,
16    Print[r];
17    For[x=1,x\[LessEqual]mx-2r+1,x++,
18      For[y=1,y\[LessEqual]my-2r+1,y++,
19        For[z=1,z\[LessEqual]mz-2r+1,z++,
20          inside=True;
21          For[a=x,a\[LessEqual]x+2r-1,a++,
22            For[b=y,b\[LessEqual]y+2r-1,b++,
23              For[c=z,c\[LessEqual]z+2r-1,c++,
24                If[((a-(r+0.5)-(x-1))^2+(b-(r+0.5)-(y-1))^2+
25                  (c-(r+0.\5)-(z-1))^2\[LessEqual]r^2),

27              If[ndata[[a,b,c]]\[NotEqual]1,inside=False;Goto[endloop]];
28            ];
29          ];
30        ];
31      ];
32      Label[endloop];

34      If[inside==True,Print[{x,y,z}];
35        clist=Flatten[{clist,{{x,y,z,r}}},1];
36        For[a=x,a\[LessEqual]x+2r-1,a++,
37          For[b=y,b\[LessEqual]y+2r-1,b++,
38            For[c=z,c\[LessEqual]z+2r-1,c++,

40              If[((a-(r+0.5)-(x-1))^2+(b-(r+0.5)-(y-1))^2+
41                (c-(r+\0.5)-(z-1))^2\[LessEqual]r^2),ndata[[a,b,c]]=0.5];
42              ];
43            ];
44          ];
45        ];
46      ];
47    ];
48  ];
49  ];//Timing
50  Save[fdir<>"clist.txt",clist];
```

```
Run["net send mich Mathematica finished!"];
Save[fdir<>"ndata.txt",ndata];

cubesh={};
For[x=1,x\[LessEqual]mx,x++,
    For[y=1,y\[LessEqual]my,y++,
    For[z=1,z\[LessEqual]mz,z++,

    If[ndata[[x,y,z]]\[Equal]1,
      cubesh=Flatten[{cubesh,{{x-1,y-1,z}}},1]];
    ];
  ];
];
cubesh=Cuboid/@cubesh;
cph=Show[Graphics3D[cubesh],Axes\[Rule]True,AxesLabel\[Rule]{"x","y","z"},
  PlotRange\[Rule]All,Shading\[Rule]False];
Show[{cph,Table[TranslateShape[
  Graphics3D[Sphere[clist[[a,4]],6,6]],{clist[[a,1]]+clist[[a,4]]-1,
    clist[[a,2]]+clist[[a,4]]-1,clist[[a,3]]+clist[[a,4]]-1}],{a,1,
      Length[clist]}]},PlotRange\[Rule]{{0,mx},{my/2,my},{0,mz}}];
Show[Table[TranslateShape[
    Graphics3D[Sphere[clist[[a,4]],8,8]],{clist[[a,1]]+clist[[a,4]]-1,
      clist[[a,2]]+clist[[a,4]]-1,clist[[a,3]]+clist[[a,4]]-1}],{a,1,
    Length[clist]}]];
```

A.1.3 DeltaVSearch.nb

```
<< Graphics`Graphics`
<< Graphics`MultipleListPlot`
<< LinearAlgebra`MatrixManipulation`
fxy[x1_, x2_, y1_, y2_, y_] :=
  x1 + (((x2 - x1)*(y - y1))/(y2 -
          y1));
\[Phi][q_Real, r_Real] :=
  N[3.0*(Sin[q*r] - q*r*Cos[q*r]\/(q*r)\^3)];\
dist[i_Integer, j_Integer] :=
    N[Sqrt[(cc[[i, 1]] - cc[[j, 1]])^2 + (cc[[i, 2]] -
                cc[[j, 2]])^2 + (cc[[i, 3]] -
                cc[[j, 3]])^2]];
int[q_Real] :=
    N[\[Sum]\+i = 1\%n(\[Phi][q, cc[[i, 4]]]^2) +
        2*\[Sum]\+i = 1\%n - 1\[Sum]\+j = i + 1\%n(\[Phi][
                  q, cc[[i, 4]]]*\[Phi][q, cc[[j, 4]]]*
                Sin[q*dist[i, j]]\/q*dist[i, j])];\
linfit[x_List, y_List, sig_List, mwt_Integer] :=
    Module[{n, sx = 0.0, sy = 0.0, ss, i, wt, sxoss, st2 = 0.0, b = 0.0, t,
        a, siga, sigb, sigdat, chi2,
        q},
n = Length[x];
If[ mwt \[NotEqual] 0,
ss = 0.0;
For[i = 1,i \[LessEqual] n, i++,
  wt =  1.0/sig[[i]]^2;
  ss += wt;
  sx += x[[i]]*wt;
  sy += y[[i]]*wt;
],
For[i = 1, i \[LessEqual] n, i++,
  sx += x[[i]];
  sy += y[[i]];
];
ss = n;
];
sxoss = sx/ss;
If[mwt \[NotEqual] 0,
For[i = 1, i \[LessEqual] n, i++,
t = (x[[i]] - sxoss)/sig[[i]];
```

```
41  st2 += t*t;
42  b += t*y[[i]]/ sig[[i]];
43  ],
44  For[i = 1, i \[LessEqual] n, i++,
45  t = x[[i]] - sxoss;
46  st2 += t*t;
47  b += t*y[[i]];
48  ]
49  ];
50  b /= st2;
51  a = (sy - sx*b)/ ss;
52  siga = Sqrt[(1.0 + sx*sx/(ss*st2))/ss];
53  sigb = Sqrt[1.0/st2];
54  chi2 = 0.0;
55  q = 1.0;
56  If[mwt \[Equal] 0,
57  For[i = 1, i \[LessEqual] n, i++,
58    chi2 += (y[[i]] - a - b*x[[i]])^2];
59  sigdat = Sqrt[chi2/(n - 2)];
60  siga *= sigdat;
61  sigb *= sigdat,
62  For[i = 1, i \[LessEqual] n, i++,
63    chi2 += ((y[[i]] - a - b*x[[i]])/sig[[i]])^2];
64  If[n > 2, q = GammaRegularized[0.5*(n - 2), 0.5*chi2],
65            Print["To few data points!"]];
66  ];

68   (*\ output : \ {a, b, siga, sigb, chi2, q}\ \ Print[
69                  a, "+", b, "*x"];\ *)
70  {a, b, siga, sigb, chi2, q}
71  ];
72  crydata = Import["experimental_lnl100.txt", "Table"][[All, {1, 3}]];
73  crydata = Take[crydata, 15];
74  qin = crydata[[All, 1]];
75  cryerr = 0.0;
76  lp = LogListPlot[crydata, PlotJoined \[Rule] True, Axes \[Rule] False,
77        Frame \[Rule] True, PlotStyle \[Rule] Hue[0.7],
78        FrameLabel \[Rule] {\*"\"q (\!A\^-1)\"", "Int (AU)"},
79        PlotStyle \[Rule] {Thickness[0.005]},
80        DisplayFunction \[Rule] Identity];
81  np = ListPlot[crydata, PlotJoined \[Rule] True, Axes \[Rule] False,
82        Frame \[Rule] True, PlotStyle \[Rule] Hue[0.7],
83        FrameLabel \[Rule] {\*"\"q (\!A\^-1)\"", "Int (AU)"},
84        PlotStyle \[Rule] {Thickness[0.005]},
85        DisplayFunction \[Rule] Identity];
86  Show[GraphicsArray[{np, lp}], Axes \[Rule] False, Frame \[Rule] True,
87      DisplayFunction \[Rule] $DisplayFunction];
88  cutlimit=2;
89  sdescr=Import["sphere model descriptions rap.txt","Table"];
90  smallm=DeleteCases[sdescr,{_,Apply[Alternatives,Range[0,cutlimit-1]],_}];
91  (*smallm=Cases[sdescr,{_,1|2,_}];*)
92  dl=Length[smallm];
93  alldata=Range[dl];
94  DeleteFile["allmodels.txt"];
95  Save["allmodels.txt",smallm];
96  smallm
97  For[mc = 1, mc \[LessEqual] dl, mc++,
98  mcoords = {smallm[[mc, 1]], smallm[[mc, 2]]};
99  mname = "spheres_" <> ToString[mcoords[[1]]] <> "_1.txt";
100 coords = Import[mname, "Table"];
101 Print[Length[coords], " spheres found in file ", "spheres_" <>
102   ToString[mcoords[[1]]] <> "_1.txt"];
103 tobecut = Apply[Alternatives, Range[-1, mcoords[[2]]]];
104 cc = DeleteCases[coords, {_, _, _, tobecut}];
105 Print["Deleting spheres with r=", tobecut];
106 Print[Length[cc], " spheres left"];
107 n = Length[cc];
108 Print["Calculating intensities of" <> ToString[mcoords]];
```

```
109 minv = 0.2;
110 maxv = 8.0;
111 deltav = 0.2;
112 ccori = cc;
113 vint = Table[
114 (
115 cc = ccori;
116 cc = cc*\@v\%3.0;
117 Print[v, "␣", ToString[fxy[0, 100, minv, maxv, v]], "%"];
118 intlist = Table[{qin[[m]], int[qin[[m]]]}, {m, 1, Length[qin]}];
119 result = linfit[intlist[[All, 2]], crydata[[All, 2]], {}, 0];
120 Flatten[{v, result}]
121  (*\ Output : \ v, a, b, siga, sigb, chi2, q\ *)
122 ), {v, minv, maxv, deltav}
123 ];

125 alldata[[mc]] = vint;
126 MultipleListPlot[vint[[All, {1, 6}]], PlotJoined \[Rule] True,
127   PlotStyle \[Rule] {Hue[0], Thickness[0.003]}, Frame \[Rule] True,
128   Axes \[Rule] False, FrameLabel \[Rule] {TraditionalForm[\*"\"V/\!V\_0\""],
129    TraditionalForm[\*"\"\!\[Chi]\^2\""]}, PlotRange \[Rule] All];
130 DeleteFile["alldata.txt"];

132 Save["alldata.txt", alldata];
133 Print["Done␣with␣", ToString[mc], "/", ToString[dl], ",␣",
134   ToString[mc/dl*100.0], "%"];
135 ]; // Timing

137 Run["shutdownxp.exe␣-h"]
```

B Pascal source codes

B.1 SAXS data modelling

B.1.1 idnsaxsfit.pas

```
{
  version 0.34
  last modified: 20051026
  - corrected Exp underflow error
  - added SaveSeed
  - random and only one move var
  - 0.3: changed to Rapana modeling
  - addeed centers move
  - added coditional fits
  - 0.34: speedup in CalcInt
}

program IDnSAXSFit;

uses GPC;

const
  Version='20051026_0.34';
  ExitError=1;
  ExitOK=0;
  Ang=1;
  IExp=2;
  IExpErr=3;
  ICalc=4;
  Limit=1;
  Step=2;
  ARed=3;
  BRed=4;
  MaxPoints=150; // Maximum exp data points
  MaxSpheres=30; // Max spheres in model monomer
  MaxModels=20; //Max number of monomers in structural subunit
  MaxSpheresDn=6000; // Max spheres in whole model, MaxSph*MaxMod*5*2 for D5
  MaxDists=17997000; // Binomial(MaxSpheresDn,2)
  HiChi=1.0E100;
  TwoPi=2.0*Pi;
  PiOverTwo=Pi/2.0;
  ThreePiOverTwo=3.0*Pi/2.0;
  FourPiOverThree=4.0*Pi/3.0;
  AngleOffset=0.001;
  ExpLowLimit=-700.0;

type
  TStr2=String[2];
  TStr8=String[8];
  TStr32=String[32];
  TData=array[1..4,1..MaxPoints] of Double;
  TSphereData=record
    x,y,z,r:Double;
```

```
49    end;
50    TSphereDataFlag=record
51      x,y,z,r:Double;
52      Chain:Char;
53    end;
54    TModelData=record
55      cx,cy,cz,a,b,c:Double;
56    end;
57    TParameters=record
58      cv:Double;
59      ModelData:array[1..MaxModels] of TModelData;
60    end;
61    TZParamCoor=record
62      vx0,vy0,vz0,cv:Double;
63      SphereData:array[1..MaxSpheres] of TSphereData;
64    end;
65    TNParamCoor=record
66      vx,vy,vz,cx,cy,cz,a,b,c:Double;
67    end;
68    TDistsTable=array[1..MaxDists] of Double;
69    TLinFitRes=record
70      FitA,FitB:Double;
71      SigFitA,SigFitB:Double;
72      ChiSqr,GofFQ:Double;
73    end;
74    TBestFitParams=record
75      LinFitRes:TLinFitRes;
76      cv:Double;
77      ModelData:array[1..MaxModels] of TModelData;
78    end;

80  var
81    GnuplotPID,RasmolPID,idum:Integer;
82    ExpDataFN,ModelFN,ModelParamsFN:String[50];
83    Points,Spheres,Models,SpheresDn,Symmetry,GoodLimit,FailLimit:Integer;
84    Data:TData;
85    ModelZC1:TZParamCoor;
86    ModelNC1:array[1..MaxModels] of TNParamCoor;
87    sx:TParameters;
88    ModelDn:array[1..MaxSpheresDn] of TSphereDataFlag;
89    DistsTable:TDistsTable;
90    ErrorsInExpData,Solid,Boing,SaveVars:Boolean;
91    FitCenters,FitVolumes,FitAngles:Boolean;
92    Phi:array[1..MaxSpheresDn] of Double;
93    LinFitRes:TLinFitRes;
94    glinext,glinextp:Integer;
95    glma:array[1..55] of Double;
96    gljdum:Integer=1;
97    t0,tn,at,bt,SolidAmount:Double;
98    CPar,VPar,APar:array[1..4] of Double;
99    fvars:Text;
100   m:array[1..9] of Double;
101   ca,cb,cc,sa,sb,sc,cacc,cbsc,ccsa:Double;
102   BestFitParams:TBestFitParams;

104 function ran3(var idum:Integer):Double;
105 const
106   mbig=4.0e6;
107   mseed=1618033.0;
108   mz=0.0;
109   fac=2.5e-7; (* 1/mbig *)
110 var
111   i,ii,k:Integer;
112   mj,mk:Double;
113 begin
114   if (idum<0) then begin
115     mj:=mseed+idum;
116     if mj>=0.0 then mj:=mj-mbig*Trunc(mj/mbig)
```

```
117         else mj:=mbig-Abs(mj)+mbig*Trunc(Abs(mj)/mbig);
118       glma[55]:=mj;
119       mk:=1;
120       for i:=1 to 54 do begin
121         ii:=21*i mod 55;
122         glma[ii]:=mk;
123         mk:=mj-mk;
124         if (mk < mz) then mk:=mk+mbig;
125         mj:=glma[ii]
126       end;
127       for k:=1 to 4 do begin
128         for i:=1 to 55 do begin
129           glma[i]:=glma[i]-glma[1+((i+30) mod 55)];
130           if (glma[i]<mz) then glma[i]:=glma[i]+mbig
131         end
132       end;
133       glinext:=0;
134       glinextp:=31;
135       idum:=1
136     end;
137     glinext:=glinext+1;
138     if (glinext=56) then glinext:=1;
139     glinextp:=glinextp+1;
140     if (glinextp=56) then glinextp:=1;
141     mj:=glma[glinext]-glma[glinextp];
142     if (mj<mz) then mj:=mj+mbig;
143     glma[glinext]:=mj;
144     ran3:=mj*fac
145   end; { ran3 }

147   function RandomDouble(var idum:Integer;LoLim,HiLim:Double):Double;
148   var td:Double;
149   begin
150     td:=LoLim+((HiLim-LoLim)*ran3(idum));
151     if (td>HiLim) or (td<LoLim) then begin
152       WriteLn('Something''s␣wrong␣with␣ran3(',idum,',',LoLim:0:6,',',HiLim:0:6,')!
153   ␣␣␣␣␣␣Aborting...');
154       Halt(ExitError);
155     end;
156     RandomDouble:=td;
157   end; { RandomDouble }

159   function RandomInteger(var idum:Integer;LoLim,HiLim:Integer):Integer;
160   var ti:Integer;
161   begin
162     ti:=Trunc(LoLim+(((HiLim+1)-LoLim)*ran3(idum)));
163     if (ti>HiLim) or (ti<LoLim) then begin
164       WriteLn('Something''s␣wrong␣with␣ran3(',idum,',',LoLim,',',HiLim,')!
165   ␣␣␣␣␣␣Aborting...');
166       Halt(ExitError);
167     end;
168     RandomInteger:=ti;
169   end;  { RandomInteger }

171   function GetPID(ProgName:String):Integer;
172   var
173     f:Text;
174     tpid:Integer;
175   begin
176     Assign(f,ProgName+'.pid');
177     {$I-}
178     Reset(f);
179     {$I+}
180     if IOResult<>0 then begin
181       WriteLn('Cannot␣read␣',ProgName,'␣PID␣from␣file␣',ProgName,'.pid!');
182       WriteLn('Execute␣',ProgName,'␣first.');
183       Halt(ExitError);
184     end;
```

```
185   ReadLn(f,tpid);
186   GetPID:=tpid;
187   Close(f);
188   WriteLn('Assuming␣PID␣for␣',ProgName,'␣is␣',tpid);
189 end; { GetPID }

191 function Radius(v:Double):Double;
192 begin
193   Radius:=0.6203504908994000166680*(v**(1/3));
194 end; { Radius }

196 function Volume(r:Double):Double;
197 begin
198   Volume:=FourPiOverThree*r**3;
199 end; { Volume }

201 procedure CalcInt;
202 var
203   QPos:Integer;

205   function Intensity(q:Double):Double;
206   var
207     i,j,DistIndex:Integer;
208     fp,sp,qd,qr:Double;
209   begin
210     fp:=0;
211     sp:=0;
212     DistIndex:=0;
213     qr:=q*ModelDn[1].r;
214     Phi[1]:=3.0*((Sin(qr)-qr*Cos(qr))/(qr**3));
215     for i:=1 to SpheresDn-1 do begin
216       for j:=i+1 to SpheresDn do begin
217         if i=1 then begin
218           qr:=q*ModelDn[j].r;
219           Phi[j]:=3.0*((Sin(qr)-qr*Cos(qr))/(qr**3));
220         end;
221         Inc(DistIndex);
222         qd:=q*DistsTable[DistIndex];
223         sp:=sp+Phi[i]*Phi[j]*Sin(qd)/qd;
224       end;
225       fp:=fp+Sqr(Phi[i]);
226     end;
227     fp:=fp+Sqr(Phi[SpheresDn]);
228     Intensity:=fp+2*sp;
229   end; { Intensity in CalcInt }

231   procedure LookUpDists;
232   var
233     j,k,DistIndex:Integer;
234   begin
235     DistIndex:=0;
236     Boing:=False;
237     for j:=1 to (SpheresDn-1) do begin
238       for k:=j+1 to SpheresDn do begin
239         Inc(DistIndex);
240         DistsTable[DistIndex]:=SqRt(Sqr(ModelDn[j].x-ModelDn[k].x)+
241           Sqr(ModelDn[j].y-ModelDn[k].y)+
242           Sqr(ModelDn[j].z-ModelDn[k].z));
243         if (Solid and
244           (DistsTable[DistIndex]<((ModelDn[j].r+ModelDn[k].r)*SolidAmount)))
245             then begin
246           Boing:=True;
247           Break;
248         end;
249       end;
250       if Boing then Break;
251     end;
252   end; { LookupDists in CalcInt }
```

```
254 begin
255   LookUpDists;
256   if not Boing then for QPos:=1 to Points do
257     Data[ICalc,QPos]:=Intensity(Data[Ang,QPos]);
258 //  WriteLn('CalcInt done!');
259 end; { CalcInt }

261 function GammLn(xx:Double):Double;
262 const
263   cof:array[1..6] of Double=(76.18009172947146, -86.50532032941677,
264     24.01409824083091, -1.231739572450155, 0.1208650973866179E-2,
265     -0.5395239384953E-5);
266 var
267   x,y,tmp,ser:Double;
268   j:Integer;
269 begin
270   x:=xx;
271   y:=x;
272   tmp:=x+5.5;
273   tmp:=(x+0.5)*Ln(tmp)-tmp;
274   ser:=1.000000000190015;
275   for j:=1 to 6 do begin
276     y:=y+1;
277     ser:=ser+cof[j]/y;
278   end;
279   GammLn:=tmp+Ln(2.5066282746310005*ser/x);
280 //  WriteLn('GammLn ok!');
281 end; { GammLn }

283 function GSer(a,x:Double):Double;
284 { rewrite with Exit }
285 const
286   itmax=100;
287   eps=3.0E-7;
288 var
289   gamser,ap,del,sum:Double;
290   n:Integer;
291 label RetLabel, FinishLabel;
292 begin
293   if x<=0.0 then begin
294     if x<0.0 then begin
295       WriteLn('x less than 0.0 in gser! Aborting...');
296       Halt(ExitError);
297     end;
298     gamser:=0.0;
299     goto FinishLabel;
300   end else begin
301     ap:=a;
302     del:=1.0/a;
303     sum:=del;
304     for n:=1 to itmax do begin
305       ap:=ap+1;
306       del:=del*x/ap;
307       sum:=sum+del;
308       if Abs(del)<Abs(sum)*eps then goto RetLabel;
309     end;
310     WriteLn('a too large, itmax to small in gser! Aborting...');
311     Halt(ExitError);
312   end;
313   RetLabel:
314 //  WriteLn('Trying GSer');
315   gamser:=sum*Exp(-x+a*Ln(x)-GammLn(a));
316   FinishLabel:
317   GSer:=gamser;
318 //  WriteLn('GSer ok!');
319 end; { GSer }
```

```
321 function GCF(a,x:Double):Double;
322 const
323   itmax=100;
324   eps=3.0E-7;
325   fpmin=1.0E-30;
326 var
327   b,c,d,h,an,del,TExp:Double;
328   i:Integer;
329 begin
330   b:=x+1.0-a;
331   c:=1.0/fpmin;
332   d:=1.0/b;
333   h:=d;
334   for i:=1 to itmax do begin
335     an:=-i*(i-a);
336     b:=b+2.0;
337     d:=an*d+b;
338     if Abs(d)<fpmin then d:=fpmin;
339     c:=b+an/c;
340     if Abs(c)<fpmin then c:=fpmin;
341     d:=1.0/d;
342     del:=d*c;
343     h:=h*del;
344     if Abs(del-1.0)<eps then Break;
345   end;
346   if i=itmax then begin
347     WriteLn('a too large, itmax to small in gcf! Aborting...');
348     Halt(ExitError);
349   end;
350 //  WriteLn('Trying GCF');
351   TExp:=-x+a*Ln(x)-GammLn(a);
352   if TExp<ExpLowLimit then GCF:=0 else GCF:=Exp(TExp)*h;
353 //  WriteLn('GCF ok!');
354 end; { GCF }

356 function GammQ(a,x:Double):Double;
357 begin
358   if ((x<0.0) or (a<=0.0)) then begin
359     WriteLn('Invalid arguments in GammaQ!');
360     Halt(ExitError);
361   end;
362   if x<(a+1.0) then GammQ:=1.0-GSer(a,x) else GammQ:=1.0-GCF(a,x);
363 //  WriteLn('GammQ ok!');
364 end; { GammQ }

366 procedure CalcChi;
367 var
368   i:Integer;
369   sx,sy,st2,b:Double=0.0;
370   ss,wt,sxoss,t,a,siga,sigb,sigdat,chi2,q:Double;
371 begin
372   if not Boing then begin
373     if ErrorsInExpData then begin
374       ss:=0.0;
375       for i:=1 to Points do begin
376         wt:=1.0/Sqr(Data[IExpErr,i]);
377         ss:=ss+wt;
378         sx:=sx+Data[ICalc,i]*wt;
379         sy:=sy+Data[IExp,i]*wt;
380       end;
381     end else begin
382       for i:=1 to Points do begin
383         sx:=sx+Data[ICalc,i];
384         sy:=sy+Data[IExp,i];
385       end;
386       ss:=Points;
387     end;
388     sxoss:=sx/ss;
```

```
389       if ErrorsInExpData then begin
390         for i:=1 to Points do begin
391           t:=(Data[ICalc,i]-sxoss)/Data[IExpErr,i];
392           st2:=st2+t*t;
393           b:=b+t*Data[IExp,i]/Data[IExpErr,i];
394         end;
395       end else begin
396         for i:=1 to Points do begin
397           t:=(Data[ICalc,i]-sxoss);
398           st2:=st2+t*t;
399           b:=b+t*Data[IExp,i];
400         end;
401       end;
402       b:=b/st2;
403       a:=(sy-sx*b)/ss;
404       siga:=SqRt((1.0+sx*sx/(ss*st2))/ss);
405       sigb:=SqRt(1.0/st2);
406       chi2:=0.0;
407       q:=1.0;
408       if not ErrorsInExpData then begin
409         for i:=1 to Points do chi2:=chi2+Sqr(Data[IExp,i]-a-b*Data[ICalc,i]);
410         sigdat:=SqRt(chi2/(Points-2));
411         siga:=siga*sigdat;
412         sigb:=sigb*sigdat;
413       end else begin
414         for i:=1 to Points do
415           chi2:=chi2+Sqr((Data[IExp,i]-a-b*Data[ICalc,i])/Data[IExpErr,i]);
416         if Points>2 then q:=GammQ(0.5*(Points-2),0.5*chi2) else begin
417           WriteLn('Too few data points for GammaQ! Aborting...');
418           Halt(ExitError);
419         end;
420       end;
421       LinFitRes.FitA:=a;
422       LinFitRes.FitB:=b;
423       LinFitRes.SigFitA:=siga;
424       LinFitRes.SigFitB:=sigb;
425       LinFitRes.ChiSqr:=chi2;
426       LinFitRes.GofFQ:=q;
427     end else begin
428       LinFitRes.FitA:=0.0;
429       LinFitRes.FitB:=0.0;
430       LinFitRes.SigFitA:=0.0;
431       LinFitRes.SigFitB:=0.0;
432       LinFitRes.ChiSqr:=HiChi;
433       LinFitRes.GofFQ:=0.0;
434     end;
435   //  WriteLn('CalcChi done!');
436   end; { CalcChi }

438   procedure GenerateNewModel;
439   var
440     ModelNum,ModelAngle:Byte;
441     WorkDone:Boolean;
442   begin
443     WorkDone:=False;
444     repeat
445       ModelNum:=RandomInteger(idum,1,Models);
446       ModelAngle:=RandomInteger(idum,0,6);
447       case ModelAngle of
448         0:   if FitVolumes then begin
449                repeat ModelZC1.cv:=sx.cv+
450                  RandomDouble(idum,-VPar[Step],VPar[Step])
451                until ((ModelZC1.cv+1.0)>(1/(VPar[Limit]+1))) and
452                  ((ModelZC1.cv+1.0)<=(VPar[Limit]+1));
453                WorkDone:=True;
454              end;
455         1:   if FitAngles then begin
456                ModelNC1[ModelNum].a:=sx.ModelData[ModelNum].a+
```

```
457                 RandomDouble(idum,-APar[Step],APar[Step]);
458               if (ModelNC1[ModelNum].a+AngleOffset)>=TwoPi then
459                 ModelNC1[ModelNum].a:=ModelNC1[ModelNum].a-TwoPi;
460               if (ModelNC1[ModelNum].a+AngleOffset)<0 then
461                 ModelNC1[ModelNum].a:=ModelNC1[ModelNum].a+TwoPi;
462               WorkDone:=True;
463             end;
464         2:  if FitAngles then begin
465               ModelNC1[ModelNum].c:=sx.ModelData[ModelNum].c+
466                 RandomDouble(idum,-APar[Step],APar[Step]);
467               if (ModelNC1[ModelNum].c+AngleOffset)>=TwoPi then
468                 ModelNC1[ModelNum].c:=ModelNC1[ModelNum].c-TwoPi;
469               if (ModelNC1[ModelNum].a+AngleOffset)<0 then
470                 ModelNC1[ModelNum].c:=ModelNC1[ModelNum].c+TwoPi;
471               WorkDone:=True;
472             end;
473         3:  if FitAngles then begin
474               repeat ModelNC1[ModelNum].b:=sx.ModelData[ModelNum].b+
475                 RandomDouble(idum,-APar[Step],APar[Step])
476               until ((ModelNC1[ModelNum].b+AngleOffset)<=Pi) and
477                 ((ModelNC1[ModelNum].b+AngleOffset)>=0.0);
478               WorkDone:=True;
479             end;
480         4:  if FitCenters then begin
481               repeat ModelNC1[ModelNum].cx:=sx.ModelData[ModelNum].cx+
482                 RandomDouble(idum,-CPar[Step],CPar[Step])
483               until Abs(ModelNC1[ModelNum].cx)<=CPar[Limit];
484               WorkDone:=True;
485             end;
486         5:  if FitCenters then begin
487               repeat ModelNC1[ModelNum].cy:=sx.ModelData[ModelNum].cy+
488                 RandomDouble(idum,-CPar[Step],CPar[Step])
489               until Abs(ModelNC1[ModelNum].cy)<=CPar[Limit];
490               WorkDone:=True;
491             end;
492         6:  if FitCenters then begin
493               repeat ModelNC1[ModelNum].cz:=sx.ModelData[ModelNum].cz+
494                 RandomDouble(idum,-CPar[Step],CPar[Step])
495               until Abs(ModelNC1[ModelNum].cz)<=CPar[Limit];
496               WorkDone:=True;
497             end;
498         else begin
499           WriteLn('Something is wrong with MoveWhat! Aborting ...');
500           Halt(ExitError);
501         end;
502       end;
503     until WorkDone;
504   end; { GenerateNewModel }

506   function PrintTimeDate:TStr32;
507   const
508     DOfW:array[0..6] of String[9]=('Sunday','Monday','Tuesday','Wednesday',
509       'Thursday','Friday','Saturday');
510   var
511     t:TimeStamp;
512   begin
513     GetTimeStamp(t);
514     PrintTimeDate:=DOfW[t.DayOfWeek]+', '+Date(t)+', '+Time(t);
515   end; { PrintTimeDate }

517   function PrintTime:TStr8;
518   var
519     t:TimeStamp;
520   begin
521     GetTimeStamp(t);
522     PrintTime:=Time(t);
523   end; { PrintTime }
```

```
525 procedure MakeDn(n:Integer);
526 var
527   i,j,Pos,nn:Integer;
528   DeltaPi,Angl,Dist,x0,y0,z0,r0,ta,nx,ny,RFac:Double;
529   tx0,ty0,tz0,OffsetX,OffsetY,OffsetZ:Double;
530   CChar:Byte;

532   procedure GenerateRotationMatrix(nn:Integer);
533   begin
534     ca:=Cos(ModelNC1[nn].a);
535     cb:=Cos(ModelNC1[nn].b);
536     cc:=Cos(ModelNC1[nn].c);
537     sa:=Sin(ModelNC1[nn].a);
538     sb:=Sin(ModelNC1[nn].b);
539     sc:=Sin(ModelNC1[nn].c);
540     cacc:=ca*cc;
541     cbsc:=cb*sc;
542     ccsa:=cc*sa;
543     m[1]:=cacc-cbsc*sa;
544     m[2]:=ca*cbsc+ccsa;
545     m[3]:=sb*sc;
546     m[4]:=-cb*ccsa-ca*sc;
547     m[5]:=cacc*cb-sa*sc;
548     m[6]:=cc*sb;
549     m[7]:=sa*sb;
550     m[8]:=-ca*sb;
551     m[9]:=cb;
552   end; { GenerateRotationMatrix in MakeDn }

554 begin
555   DeltaPi:=TwoPi/n;
556   Pos:=1;
557   RFac:=(ModelZC1.cv+1.0)**(1/3); //Do volumes
558 //  RFac:=1.0;  //Volume bypass
559   for nn:=1 to Models do begin
560     GenerateRotationMatrix(nn);
561     OffsetX:=ModelZC1.vx0+ModelNC1[nn].vx+ModelNC1[nn].cx;
562     OffsetY:=ModelZC1.vy0+ModelNC1[nn].vy+ModelNC1[nn].cy;
563     OffsetZ:=ModelZC1.vz0+ModelNC1[nn].vz+ModelNC1[nn].cz;
564     for i:=1 to Spheres do begin
565       CChar:=Ord('A');
566       tx0:=RFac*(ModelZC1.SphereData[i].x-ModelZC1.vx0)+OffsetX;
567       ty0:=RFac*(ModelZC1.SphereData[i].y-ModelZC1.vy0)+OffsetY;
568       tz0:=RFac*(ModelZC1.SphereData[i].z-ModelZC1.vz0)+OffsetZ;
569       r0:=ModelZC1.SphereData[i].r*RFac;
570       x0:=m[1]*(tx0-OffsetX)+m[2]*(ty0-OffsetY)+m[3]*(tz0-OffsetZ)+OffsetX;
571       y0:=m[4]*(tx0-OffsetX)+m[5]*(ty0-OffsetY)+m[6]*(tz0-OffsetZ)+OffsetY;
572       z0:=m[7]*(tx0-OffsetX)+m[8]*(ty0-OffsetY)+m[9]*(tz0-OffsetZ)+OffsetZ;
573       ModelDn[Pos].x:=x0;
574       ModelDn[Pos].y:=y0;
575       ModelDn[Pos].z:=z0;
576       ModelDn[Pos].r:=r0;
577       ModelDn[Pos].Chain:='A';
578       ModelDn[Pos+1].x:=-x0;
579       ModelDn[Pos+1].y:=y0;
580       ModelDn[Pos+1].z:=-z0;
581       ModelDn[Pos+1].r:=r0;
582       ModelDn[Pos+1].Chain:='B';
583       Pos:=Pos+2;
584       Dist:=SqRt(Sqr(x0)+Sqr(y0));
585       if x0>0.0 then Angl:=ArcTan(y0/x0) else
586         if x0<0.0 then Angl:=Pi+ArcTan(y0/x0) else
587         if y0>0.0 then Angl:=PiOverTwo else Angl:=ThreePiOverTwo;
588       for j:=1 to (n-1) do begin
589         CChar:=CChar+2;
590         ta:=Angl+j*DeltaPi;
591         nx:=Dist*Cos(ta);
592         ny:=Dist*Sin(ta);
```

```
593           ModelDn[Pos].x:=nx;
594           ModelDn[Pos].y:=ny;
595           ModelDn[Pos].z:=z0;
596           ModelDn[Pos].r:=r0;
597           ModelDn[Pos].Chain:=Chr(CChar);
598           ModelDn[Pos+1].x:=-nx;
599           ModelDn[Pos+1].y:=ny;
600           ModelDn[Pos+1].z:=-z0;
601           ModelDn[Pos+1].r:=r0;
602           ModelDn[Pos+1].Chain:=Chr(CChar+1);
603           Pos:=Pos+2;
604         end;
605       end;
606     end;
607   end; { MakeDn }

609   procedure SendKill(PID,UsrSig:Integer);
610   var KillStatus:Boolean;
611   begin
612     KillStatus:=Kill(PID,UsrSig);
613     if not KillStatus then begin
614       WriteLn('Something''s␣wrong␣with␣Kill(',PID,',',UsrSig,')!␣Aborting...');
615       Halt(ExitError);
616     end;
617   end; { SendKill }

619   procedure SavePDB(fn:String);
620   var
621     fOut:Text;
622     j:Integer;
623   begin
624     Assign(fOut,fn);
625     Rewrite(fOut);
626     for j:=1 to SpheresDn do
627       WriteLn(fOut,'ATOM␣␣␣␣␣␣␣␣␣CA␣␣UNK␣',ModelDn[j].Chain,'␣␣␣␣␣␣␣␣',
628         ModelDn[j].x:8:3,ModelDn[j].y:8:3,ModelDn[j].z:8:3,'␣␣1.00',
629         ModelDn[j].r:6:2,'␣␣␣␣␣␣␣␣␣␣C');
630     Close(fOut);
631     Flush(fOut);
632   end; { SavePDB }

634   procedure SaveDat(fn:String;WriteFileNumber:Boolean);
635   var
636     fOut:Text;
637     j,ModelCount:Integer;
638   begin
639     Assign(fOut,fn+'.dat');
640     Rewrite(fOut);
641   //  WriteLn(fOut,'␣Q␣␣␣␣␣␣␣␣␣␣␣␣␣I(Exp)␣␣␣␣␣␣␣␣I(Exp)Error␣␣␣␣␣␣␣I(Calc)');
642     for j:=1 to Points do
643       WriteLn(fOut,Data[Ang,j]:12,'␣␣',Data[IExp,j]:12,'␣␣',Data[IExpErr,j]:12,
644         '␣␣',Data[ICalc,j]:12);
645     Close(fOut);
646     Flush(fOut);
647     Assign(fOut,fn+'.fit');
648     Rewrite(fOut);
649     WriteLn(fOut,LinFitRes.FitA,'␣',LinFitRes.FitB,'␣',LinFitRes.SigFitA,'␣',
650       LinFitRes.SigFitB,'␣',LinFitRes.ChiSqr,'␣',LinFitRes.GofFQ);
651     WriteLn(fOut,ModelZC1.cv+1.0);
652     for ModelCount:=1 to Models do
653       WriteLn(fOut,ModelNC1[ModelCount].cx,'␣',ModelNC1[ModelCount].cy,
654         '␣',ModelNC1[ModelCount].cz,'␣',ModelNC1[ModelCount].a+AngleOffset,
655         '␣',ModelNC1[ModelCount].b+AngleOffset,'␣',
656         ModelNC1[ModelCount].c+AngleOffset);
657     WriteLn(fOut,'#␣FitA␣FitB␣SigFitA␣SigFitB␣ChiSqr␣GofFQ');
658     WriteLn(fOut,'#␣abs(cv)');
659     WriteLn(fOut,'#␣relative␣cx␣cy␣cz␣a␣b␣c␣for␣every␣monomer');
660     BestFitParams.LinFitRes:=LinFitRes;
```

```
661     BestFitParams.cv:=ModelZC1.cv+1.0;
662     for ModelCount:=1 to Models do begin
663       BestFitParams.ModelData[ModelCount].cx:=ModelNC1[ModelCount].cx;
664       BestFitParams.ModelData[ModelCount].cy:=ModelNC1[ModelCount].cy;
665       BestFitParams.ModelData[ModelCount].cz:=ModelNC1[ModelCount].cz;
666       BestFitParams.ModelData[ModelCount].a:=ModelNC1[ModelCount].a+AngleOffset;
667       BestFitParams.ModelData[ModelCount].b:=ModelNC1[ModelCount].b+AngleOffset;
668       BestFitParams.ModelData[ModelCount].c:=ModelNC1[ModelCount].c+AngleOffset;
669     end;
670     Close(fOut);
671     Flush(fOut);
672     if WriteFileNumber then begin
673       Assign(fOut,'best.fn');
674       Rewrite(fOut);
675       WriteLn(fOut,Copy(fn,6,Length(fn)-5));
676       Close(fOut);
677       Flush(fOut);
678     end;
679   end; { SaveDat }

681   procedure SaveMod(fn:String);
682   var
683     fOut:Text;
684     j,Pos,ModelCount:Integer;
685   begin
686     Assign(fOut,fn+'.mod');
687     Rewrite(fOut);
688     Pos:=1;
689     for j:=1 to Spheres do begin
690       WriteLn(fOut,ModelDn[Pos].x:0:6,' ',ModelDn[Pos].y:0:6,' ',
691         ModelDn[Pos].z:0:6,' ',ModelDn[Pos].r:0:6);
692      Pos:=Pos+2*Symmetry;
693     end;
694     Close(fOut);
695     Flush(fOut);
696     Assign(fOut,fn+'.par');
697     Rewrite(fOut);
698     for ModelCount:=1 to Models do
699       WriteLn(fOut,ModelNC1[ModelCount].vx+ModelZC1.vx0+
700         ModelNC1[ModelCount].cx:0:6,' ',ModelNC1[ModelCount].vy+ModelZC1.vy0+
701         ModelNC1[ModelCount].cy:0:6,' ',ModelNC1[ModelCount].vz+ModelZC1.vz0+
702         ModelNC1[ModelCount].cz:0:6,' ',ModelNC1[ModelCount].a:0:6,' ',
703         ModelNC1[ModelCount].b:0:6,' ',ModelNC1[ModelCount].c:0:6);
704     Close(fOut);
705     Flush(fOut);
706   end; { SaveMod }

708   procedure LoadParameters;
709   var
710     ts:String[100];
711     fn:Text;
712     ts1:Char;

714     procedure ParseString(ts:String);
715     var
716       es,esn:String[50];
717       Count,Code:Integer;
718       CodeError:Boolean;

720       procedure ParseError;
721       begin

723         WriteLn;
724         WriteLn('Warning reading parameters.txt:');
725         WriteLn('Can''t understand variable ',esn,' with value ',es);
726         Halt(ExitError);
727       end; { ParseError in ParseString in LoadParameters }
```

```
729     procedure Decode4Params(es:String;var p1,p2,p3,p4:Double;
730       var CodeError:Boolean);
731     var
732       CharPos,StartPos,Code1,Code2,Code3,Code4:Integer;
733     begin
734       CharPos:=1;
735       while (es[CharPos]=' ') or (es[CharPos]=#9) do Inc(CharPos);
736       while (es[CharPos]<>' ') and (es[CharPos]<>#9) do Inc(CharPos);
737       Val(Copy(es,1,CharPos-1),p1,Code1);
738       StartPos:=CharPos+1;
739       Inc(CharPos);
740       while (es[CharPos]=' ') or (es[CharPos]=#9) do Inc(CharPos);
741       while (es[CharPos]<>' ') and (es[CharPos]<>#9) do Inc(CharPos);
742       Val(Copy(es,StartPos,CharPos-StartPos),p2,Code2);
743       StartPos:=CharPos+1;
744       Inc(CharPos);
745       while (es[CharPos]=' ') or (es[CharPos]=#9) do Inc(CharPos);
746       while (es[CharPos]<>' ') and (es[CharPos]<>#9) do Inc(CharPos);
747       Val(Copy(es,StartPos,CharPos-StartPos),p3,Code3);
748       StartPos:=CharPos+1;
749       Inc(CharPos);
750       while (es[CharPos]=' ') or (es[CharPos]=#9) do Inc(CharPos);
751       while (es[CharPos]<>' ') and (es[CharPos]<>#9) do Inc(CharPos);
752       Val(Copy(es,StartPos,CharPos-StartPos),p4,Code4);
753       if (Code1<>0) or (Code2<>0) or (Code3<>0) or (Code4<>0) then
754         CodeError:=True else CodeError:=False;
755     end; { Decode4Params in ParseString in LoadParameters }

757   begin
758     Count:=1;
759     while ts[Count]<>'=' do Inc(Count);
760     esn:=Copy(ts,1,Count-1);
761     es:=Copy(ts,Count+1,Length(ts)-Count);
762     if esn='ExpDataFileName' then begin
763       ExpDataFN:=es;
764       WriteLn(' ',esn,' = ',es);
765     end else if esn='ErrorsInExpData' then begin
766       if es='1' then ErrorsInExpData:=True else
767         ErrorsInExpData:=False;
768       WriteLn(' ',esn,' = ',ErrorsInExpData);
769     end else if esn='ModelFileName' then begin
770       ModelFN:=es;
771       WriteLn(' ',esn,' = ',ModelFN);
772     end else if esn='ModelParameters' then begin
773       ModelParamsFN:=es;
774       WriteLn(' ',esn,' = ',ModelParamsFN);
775     end else if esn='Symmetry' then begin
776       Val(es,Symmetry,Code);
777       if Code<>0 then ParseError;
778       WriteLn(' ',esn,' = ',Symmetry);
779     end else if esn='Solid' then begin
780       if es='1' then Solid:=True else
781         Solid:=False;
782       WriteLn(' ',esn,' = ',Solid);
783     end else if esn='SolidAmount' then begin
784       Val(es,SolidAmount,Code);
785       if Code<>0 then ParseError;
786       Write(' ',esn,' = ',SolidAmount:0:2);
787       if not Solid then WriteLn(', ignored') else WriteLn;
788     end else if esn='GoodLimit' then begin
789       Val(es,GoodLimit,Code);
790       if Code<>0 then ParseError;
791       WriteLn(' ',esn,' = ',GoodLimit);
792     end else if esn='FailLimit' then begin
793       Val(es,FailLimit,Code);
794       if Code<>0 then ParseError;
795       WriteLn(' ',esn,' = ',FailLimit);
796     end else if esn='FitCenters' then begin
```

```
797         if es='1' then FitCenters:=True else
798           FitCenters:=False;
799         WriteLn('␣',esn,'␣=␣',FitCenters);
800       end else if esn='FitVolumes' then begin
801         if es='1' then FitVolumes:=True else
802           FitVolumes:=False;
803         WriteLn('␣',esn,'␣=␣',FitVolumes);
804       end else if esn='FitAngles' then begin
805         if es='1' then FitAngles:=True else
806           FitAngles:=False;
807         WriteLn('␣',esn,'␣=␣',FitAngles);
808       end else if esn='Centers' then begin
809         Decode4Params(es,CPar[Limit],CPar[Step],CPar[ARed],CPar[BRed],CodeError);
810         if CodeError then ParseError;
811         Write('␣',esn,'␣=␣',CPar[ARed]:0:2,'␣*␣Ran(',CPar[Limit]:0:2,
812           ',␣',CPar[Step]:0:2,')␣+␣',CPar[BRed]:0:2);
813         if not FitCenters then WriteLn(',␣ignored') else WriteLn;
814       end else if esn='Volumes' then begin
815         Decode4Params(es,VPar[Limit],VPar[Step],VPar[ARed],VPar[BRed],CodeError);
816         if CodeError then ParseError;
817         Write('␣',esn,'␣=␣',VPar[ARed]:0:2,'␣*␣Ran(',VPar[Limit]:0:2,
818           ',␣',VPar[Step]:0:2,')␣+␣',VPar[BRed]:0:2);
819         if not FitVolumes then WriteLn(',␣ignored') else WriteLn;
820       end else if esn='Angles' then begin
821         Decode4Params(es,APar[Limit],APar[Step],APar[ARed],APar[BRed],CodeError);
822         if CodeError then ParseError;
823         Write('␣',esn,'␣=␣',APar[ARed]:0:2,'␣*␣Ran(Full,␣',
824           APar[Step]:0:2,')␣+␣',APar[BRed]:0:2);
825         if not FitAngles then WriteLn(',␣ignored') else WriteLn;
826       end else if esn='Temperature' then begin
827         Decode4Params(es,t0,tn,at,bt,CodeError);
828         if CodeError then ParseError;
829         WriteLn('␣',esn,'␣=␣(',at:0:4,'␣*',t0:12,'␣+␣',
830           bt:0:2,')␣>=',tn:12);
831       end else if esn='SaveVars' then begin
832         if es='1' then SaveVars:=True else
833           SaveVars:=False;
834         WriteLn('␣',esn,'␣=␣',SaveVars);
835       end else begin
836         WriteLn;
837         WriteLn('Warning␣reading␣parameters.txt:');
838         WriteLn('Can''t␣understand␣variable␣',esn);
839         WriteLn('Should␣be␣one␣of:␣ExpDataFileName,␣ErrorsInExpData,␣',
840           'ModelFileName,␣ModelParameters,␣Symmetry,␣Solid,␣SolidAmount,␣',
841           'Centers,␣Volumes,␣Angles,␣FitCenters,␣FitVolumes,␣FitAngles,␣',
842           'GoodLimit,␣FailLimit,␣Temperature,␣SaveVars');
843         Halt(ExitError);
844       end;
845     end; { ParseString in LoadParameters }

847   begin
848     Assign(fn,'parameters.txt');
849     Reset(fn);
850     while not EOF(fn) do begin
851       ReadLn(fn,ts);
852       ts1:=ts[1];
853       if ((Copy(ts,1,1)<>'#') and (Ord(ts1)<>13)  and
854         (Length(ts)>0)) then ParseString(ts);
855     end;
856     Close(fn);
857     Assign(fvars,'variables.txt');
858     Rewrite(fvars);
859   end; { LoadParameters }

861   procedure LoadExpData(fn:String);
862   var
863     fIn:Text;
864   begin
```

```
865     Assign(fIn,fn);
866     Write(fn,' ');
867     Reset(fIn);
868     Points:=0;
869     FillChar(Data,SizeOf(Data),0);
870     FillChar(BestFitParams,SizeOf(BestFitParams),0);
871     while not EOF(fIn) do begin
872       Points:=Points+1;
873       if Points>MaxPoints then begin
874         WriteLn;
875         WriteLn('Maximum number of data points reached, aborting...');
876         Halt(ExitError);
877       end;
878       ReadLn(fIn,Data[Ang,Points],Data[IExp,Points],Data[IExpErr,Points]);
879     end;
880     Close(fIn);
881     WriteLn(Points,' points loaded');
882   end; { LoadExpData }

884   procedure LoadModelData(fn:String);
885   var
886     fIn:Text;
887     VTot:Double=0.0;
888     i:Integer;
889   begin
890     Assign(fIn,fn);
891     Write(fn,' ');
892     Reset(fIn);
893     FillChar(ModelZC1,SizeOf(ModelZC1),0);
894     Spheres:=0;
895     while (not EOF(fIn)) do begin
896       Inc(Spheres);
897       if ((Spheres>MaxSpheres) or
898         ((Spheres*Symmetry*2*MaxModels)>MaxSpheresDn)) then begin
899         WriteLn;
900         WriteLn('Too many spheres in D',Symmetry,
901           ' model, reduce the symmetry or number of spheres in C1 model! Aborting...');
902         Halt(ExitError);
903       end;
904       ReadLn(fIn,ModelZC1.SphereData[Spheres].x,ModelZC1.SphereData[Spheres].y,
905         ModelZC1.SphereData[Spheres].z,ModelZC1.SphereData[Spheres].r);
906       VTot:=VTot+Volume(ModelZC1.SphereData[Spheres].r);
907     end;
908     WriteLn(Spheres,' spheres loaded');
909     Close(fIn);
910     WriteLn('Volume of C1 model = ',VTot:12);
911     for i:=1 to Spheres do begin
912       ModelZC1.vx0:=ModelZC1.vx0+
913         (ModelZC1.SphereData[i].x*Volume(ModelZC1.SphereData[i].r)/VTot);
914       ModelZC1.vy0:=ModelZC1.vy0+
915         (ModelZC1.SphereData[i].y*Volume(ModelZC1.SphereData[i].r)/VTot);
916       ModelZC1.vz0:=ModelZC1.vz0+
917         (ModelZC1.SphereData[i].z*Volume(ModelZC1.SphereData[i].r)/VTot);
918     end;
919     WriteLn('Center of C1 model = (',ModelZC1.vx0:12,', ',ModelZC1.vy0:12,', ',
920       ModelZC1.vz0:12,')');
921   end; { LoadModelData }

923   procedure LoadModelParams(fn:String);
924   var
925     fIn:Text;
926     tdx,tdy,tdz,tda,tdb,tdc:Double;
927   begin
928     Assign(fIn,fn);
929     Reset(fIn);
930     Models:=0;
931     while (not EOF(fIn)) do begin
932       Inc(Models);
```

```
933       if ((Models>MaxModels) or ((Spheres*Symmetry*2*MaxModels)>MaxSpheresDn))
934         then begin
935         WriteLn;
936         WriteLn('Too many monomer models in D',Symmetry,
937           ' model, reduce the symmetry, number of spheres or number',
938           ' of monomer models in C1 model! Aborting...');
939         Halt(ExitError);
940       end;
941       ReadLn(fIn,tdx,tdy,tdz,tda,tdb,tdc);
942       ModelNC1[Models].vx:=tdx-ModelZC1.vx0;
943       ModelNC1[Models].vy:=tdy-ModelZC1.vy0;
944       ModelNC1[Models].vz:=tdz-ModelZC1.vz0;
945       ModelNC1[Models].a:=tda;
946       ModelNC1[Models].b:=tdb;
947       ModelNC1[Models].c:=tdc;
948     end;
949     WriteLn(Models,' monomer models loaded');
950     SpheresDn:=Spheres*Symmetry*2*Models;
951     WriteLn('D',Symmetry,' model will have ',SpheresDn,' spheres');
952     Close(fIn);
953   end; { LoadModelParams }

955   function fxy(x1,x2,y1,y2,y:Double):Double;
956   begin
957     if y1=y2 then fxy:=x1 else fxy:=(x2-x1)*(y-y1)/(y2-y1)+x1;
958   end; { fxy }

960   procedure StatusReport;
961   const ShowModel=1;
962   begin
963     WriteLn(fvars,sx.ModelData[ShowModel].cx:0:6,' ',
964       sx.ModelData[ShowModel].cy:0:6,' ',sx.ModelData[ShowModel].cz:0:6,' ',
965       sx.cv+1.0:0:6,' ',sx.ModelData[ShowModel].a+AngleOffset:0:6,' ',
966       sx.ModelData[ShowModel].b+AngleOffset:0:6,' ',
967       sx.ModelData[ShowModel].c+AngleOffset:0:6,' ',
968       CPar[Step]:0:6,' ',VPar[Step]:0:6,' ',APar[Step]:0:6);
969       //V and angles are shown as abs!
970   end; { StatusReport }

972   procedure SimmAnn;
973   var
974     ed,DeltaChi,t,Fsx,Fbx,Fnx:Double;
975     Ok,dr,NoLF:Boolean;
976     UHAcc,UHRej,FileNum,ModelCount,Weight,Good,Fail,BounceCount,
977       NewModelCount:Integer;
978     fn:String[20];
979   begin
980     sx.cv:=ModelZC1.cv;
981     for ModelCount:=1 to Models do begin
982       sx.ModelData[ModelCount].cx:=ModelNC1[ModelCount].cx;
983       sx.ModelData[ModelCount].cy:=ModelNC1[ModelCount].cy;
984       sx.ModelData[ModelCount].cz:=ModelNC1[ModelCount].cz;
985       sx.ModelData[ModelCount].a:=ModelNC1[ModelCount].a;
986       sx.ModelData[ModelCount].b:=ModelNC1[ModelCount].b;
987       sx.ModelData[ModelCount].c:=ModelNC1[ModelCount].c;
988     end;
989     Fsx:=LinFitRes.ChiSqr;
990     Fbx:=Fsx;
991     t:=t0;
992     FileNum:=0;
993     while t>=tn do begin
994       Weight:=0;
995       Ok:=False;
996       NoLF:=False;
997       Good:=0;
998       Fail:=0;
999       UHAcc:=0;
1000      UHRej:=0;
```

```
1001      BounceCount:=0;
1002      NewModelCount:=0;
1003      while not Ok do begin
1004        repeat
1005          GenerateNewModel;
1006          MakeDn(Symmetry);
1007          CalcInt;
1008          CalcChi;
1009          Fnx:=LinFitRes.ChiSqr;
1010          Inc(NewModelCount);
1011          if Fnx>=HiChi then Inc(BounceCount);
1012        until LinFitRes.ChiSqr<HiChi;
1013        Inc(Weight);
1014        Write('.');
1015        NoLF:=True;
1016        DeltaChi:=Fnx-Fsx;
1017        if SaveVars then StatusReport;
1018        if DeltaChi<=0.0 then begin
1019          Inc(Good);
1020          sx.cv:=ModelZC1.cv;
1021          for ModelCount:=1 to Models do begin
1022            sx.ModelData[ModelCount].cx:=ModelNC1[ModelCount].cx;
1023            sx.ModelData[ModelCount].cy:=ModelNC1[ModelCount].cy;
1024            sx.ModelData[ModelCount].cz:=ModelNC1[ModelCount].cz;
1025            sx.ModelData[ModelCount].a:=ModelNC1[ModelCount].a;
1026            sx.ModelData[ModelCount].b:=ModelNC1[ModelCount].b;
1027            sx.ModelData[ModelCount].c:=ModelNC1[ModelCount].c;
1028          end;
1029          Fsx:=Fnx;
1030          if Fsx<Fbx then begin
1031            Fbx:=Fsx;
1032            Inc(FileNum);
1033            Str(FileNum,fn);
1034            SavePDB('best-'+fn+'.pdb');
1035            SaveDat('best-'+fn,True);
1036            SaveMod('best-');
1037            SavePDB('best.pdb');
1038            SaveDat('best',False);
1039            SaveMod('best');
1040            SendKill(RasmolPID,SigUsr2);
1041            SendKill(GnuplotPID,SigUsr2);
1042            if NoLF then WriteLn;
1043            WriteLn('Found new ChiSqr(',LinFitRes.ChiSqr:12,
1044              '), saving results in best-',fn,'.pdb/.mod/.par/.dat/.fit');
1045            NoLF:=False;
1046          end;
1047        end else begin
1048          Inc(Fail);
1049          ed:=-DeltaChi/t;
1050          if ed<ExpLowLimit then dr:=False else dr:=ran3(gljdum)<Exp(ed);
1051          if dr then begin
1052            sx.cv:=ModelZC1.cv;
1053            for ModelCount:=1 to Models do begin
1054              sx.ModelData[ModelCount].cx:=ModelNC1[ModelCount].cx;
1055              sx.ModelData[ModelCount].cy:=ModelNC1[ModelCount].cy;
1056              sx.ModelData[ModelCount].cz:=ModelNC1[ModelCount].cz;
1057              sx.ModelData[ModelCount].a:=ModelNC1[ModelCount].a;
1058              sx.ModelData[ModelCount].b:=ModelNC1[ModelCount].b;
1059              sx.ModelData[ModelCount].c:=ModelNC1[ModelCount].c;
1060            end;
1061            if SaveVars then StatusReport;
1062            Fsx:=Fnx;
1063            Inc(UHAcc);
1064          end else Inc(UHRej);
1065        end;
1066        Ok:=(Good=GoodLimit) or (Fail=FailLimit);
1067      end;
1068      t:=at*t+bt;
```

```
1069        CPar[Step]:=CPar[Step]*CPar[ARed]+CPar[BRed];
1070        VPar[Step]:=VPar[Step]*VPar[ARed]+VPar[BRed];
1071        APar[Step]:=APar[Step]*APar[ARed]+APar[BRed];
1072        if NoLF then WriteLn;
1073        WriteLn('␣␣␣␣Time␣␣␣␣␣␣␣Temp␣␣␣␣␣␣␣␣␣␣Chi␣␣Trials
1074 ␣␣␣␣␣Bounces(%)␣␣UhAcc(%)␣␣UhRej(%)');
1075        WriteLn(PrintTime,'␣',t:10,'␣',Fbx:12,'␣␣',Weight:6,'␣␣␣␣␣␣',
1076          (BounceCount/(BounceCount+NewModelCount))*100:6:2,'␣␣␣␣',
1077          (UHAcc/(UHRej+UHAcc))*100:6:2,'␣␣␣␣',(UHRej/(UHRej+UHAcc))*100:6:2);
1078        NoLF:=False;
1079      end;
1080  end;  { SimmAnn }

1082  procedure SaveSeed;
1083  var f:Text;
1084  begin
1085    Assign(f,'start.ran');
1086    Rewrite(f);
1087    WriteLn(f,idum);
1088    Close(f);
1089  end; { SaveSeed }

1091  procedure FinalReport;
1092  var ModelCount:Integer;
1093  begin
1094    WriteLn('-------------------------------------------------',
1095      '-------------------------------------------------');
1096    WriteLn;
1097    WriteLn('␣␣␣␣␣␣␣␣FitA␣␣␣␣␣␣␣␣␣FitB␣␣␣␣␣␣␣SigFitA␣␣␣␣␣␣',
1098      'SigFitB␣␣␣␣␣␣␣ChiSqr␣␣␣␣␣␣␣␣GofFQ');
1099    WriteLn(BestFitParams.LinFitRes.FitA:12,'␣',
1100      BestFitParams.LinFitRes.FitB:12,'␣',
1101      BestFitParams.LinFitRes.SigFitA:12,'␣',
1102      BestFitParams.LinFitRes.SigFitB:12,'␣',
1103      BestFitParams.LinFitRes.ChiSqr:12,'␣',
1104      BestFitParams.LinFitRes.GofFQ:12);
1105    WriteLn;
1106    WriteLn('Monomer␣absolute␣delta␣V␣factor:␣',BestFitParams.cv:0:2);
1107    WriteLn('Monomers␣relative␣centers␣and␣angles:');
1108    for ModelCount:=1 to Models do
1109      WriteLn(ModelCount:2,'␣',BestFitParams.ModelData[ModelCount].cx:12,'␣',
1110        BestFitParams.ModelData[ModelCount].cy:12,'␣',
1111        BestFitParams.ModelData[ModelCount].cz:12,'␣',
1112        BestFitParams.ModelData[ModelCount].a:12,'␣',
1113        BestFitParams.ModelData[ModelCount].b:12,'␣',
1114        BestFitParams.ModelData[ModelCount].c:12);
1115    WriteLn;
1116    WriteLn('-------------------------------------------------',
1117      '-------------------------------------------------');
1118    WriteLn;
1119  end; { FinalReport }

1121  begin
1122    WriteLn('IDnSAXSFit␣(',Version,')');
1123    Randomize;
1124    idum:=-Random(999)-12;
1125    Write('Initializing␣RNG␣with␣seed␣=␣',idum);
1126    SaveSeed;
1127    WriteLn(',␣ran3␣=␣',ran3(idum):0:6);
1128    GnuplotPID:=GetPID('gnuplotpipe');
1129    RasmolPID:=GetPID('rasmolpipe');
1130    WriteLn('Loading␣parameters␣from␣parameters.txt...');
1131    LoadParameters;
1132    Write('Loading␣experimental␣data:␣');
1133    LoadExpData(ExpDataFN);  // Filled Data[1..4,1..Points]
1134    Write('Loading␣C1␣model␣data:␣');
1135    LoadModelData(ModelFN);  // Filled ModelZC1 cv,vxyz0,[1..Spheres],xyzr
1136    Write('Loading␣monomer␣model␣parameters:␣');
```

```
1137    LoadModelParams(ModelParamsFN); //Filled ModelNC1[1..Models]
1138    MakeDn(Symmetry);  // Filled ModelDn[1..SpheresDn]
1139    WriteLn('Writing start.pdb/.mod/.par...');
1140    SavePDB('start.pdb');
1141    SaveMod('start');
1142    WriteLn('Calculating scattering intensity...');
1143    CalcInt;  // of ModelDn, filled Data[ICalc,Points]
1144    CalcChi;  // Filled LinFitRes
1145    WriteLn('Start Chi squared =',LinFitRes.ChiSqr:12);
1146    WriteLn('Writing start.dat and start.fit...');
1147    SaveDat('start',False);
1148    WriteLn('Starting fit at ',PrintTimeDate);
1149    SendKill(GnuplotPID,SigUsr1);
1150    SendKill(RasmolPID,SigUsr1);
1151    SimmAnn;
1152    WriteLn;
1153    Close(fvars);
1154    FinalReport;
1155    WriteLn('Finished at ',PrintTimeDate);
1156    Halt(ExitOK);
1157  end.
```

B.1.2 makefile

```
1  TARG = idnsaxsfit
2  CPP = gpc
3  CPPFLAGS = -Wall -W --automake -O2 --executable-file-name

5  default : $(TARG)

7  $(TARG) : $(TARG).pas
8      $(CPP) $(CPPFLAGS) $(TARG).pas

10 fastmath : $(TARG).pas
11     $(CPP) $(CPPFLAGS) --fast-math $(TARG).pas

13 clean :
14     rm -f $(TARG).exe

16 cleandata :
17     rm -f best-*.*

19 cleanpids :
20     rm -f *.pid rdump gdump
```

B.1.3 parameters.txt

```
1  # parameters file parameters.txt
2  # for idnsaxsfit v 0.3
3  # regular text file, unix end of line

5  # Experimental data file
6  # With format Ang, IExp, IExpError or 0.0, tab or space separated

8  ExpDataFileName=rapana_new.dat
9  ErrorsInExpData=1

11 # Model data file and symmetry

13 ModelFileName=monomer_19-8-3.mod
14 ModelParameters=start-mon-8.par
15 Symmetry=5
16 Solid=0
17 SolidAmount=0.45
```

```
20 # Fit parameters=Limit Step AStepRed BStepRed
21 # Limit parameter for Angles is ignored but must be present
22 # Centers and volumes are relative

24 FitVolumes=1
25 FitAngles=1
26 FitCenters=1

28 Volumes=0.5 0.05 1.0 0.0
29 Angles=0.0 0.5 1.0 0.0
30 Centers=15.0 1.5 1.0 0.0

32 # Annealing parameters
33 # Temp=Initial Final ATempRed BTempRed

35 GoodLimit=40
36 FailLimit=400
37 Temperature=1e3 1e-2 0.9 0.0

39 # Save fitting variables at each step?

41 SaveVars=1
```

C C source codes

C.1 RasMol and gnuplot communication

C.1.1 genericpipe.c

```
1  /*
2    ver 0.2:  Added SIGUSR1 and SIGUSR2 handling, optional time delay
3    ver 0.21: Modified to reread each time the files
4    ver 0.22: Added debug
5  */

7  #include <stdio.h>
8  #include <stdlib.h>
9  #include <signal.h>
10 #include <unistd.h>
11 #include <string.h>
12 #include "genericpipe.h"

14 FILE *frasmol, *freplot, *fplot;
15 char fnp[100];
16 char fnr[100];
17 char pidfile[100]=PNAME;
18 void quit();
19 void plot();
20 void replot();

22 int main (int argc, const char *argv[]) {
23   int delta;
24   char command[255];
25 /*
26   char extension[8];
27   strcpy(pidfile,PNAME);
28   strcpy(extension,"pipe.pid");
29   strcat(pidfile,extension);
30 */
31   FILE *fpid;
32   if ((argc<3)||(argc>4)) {
33     printf("Usage: %spipe <plot_commands_file> \
34 <replot_commands_file> <delay in seconds>\n",PNAME);
35     printf("  If delay time is missing, plot is executed on SIGUSR1 and \
36 replot on SIGUSR2\n");
37     exit(EXIT_FAILURE);
38   }
39   strcpy(fnp,*(argv+1));
40   strcpy(fnr,*(argv+2));
41 //  strcpy(fn1,argv[1]);
42   if ((fplot=fopen(fnp,"r"))==NULL) {
43     printf("Cannot open plot_commands file %s!\n",fnp);
44     exit(EXIT_FAILURE);
45   }
46   if ((freplot=fopen(fnr,"r"))==NULL) {
47     printf("Cannot open replot_commands file %s!\n",fnr);
48     exit(EXIT_FAILURE);
```

```
49    }
50    fclose(fplot);
51    fclose(freplot);
52    delta=0;
53    if (argc==4) {
54      if ((delta=atoi(argv[3]))==0) {
55        printf("Time delay must be >0!\n");
56        exit(EXIT_FAILURE);
57      }
58    }
59    if (argc==3) {
60      strcat(pidfile,"pipe.pid");
61      if ((fpid=fopen(pidfile,"w"))==NULL) {
62        printf("Cannot write PID to file %s!\n",pidfile);
63        exit(EXIT_FAILURE);
64      }
65      delta=0;
66      fprintf(fpid,"%d",getpid());
67      fclose(fpid);
68      printf("Program ID is %d\n",getpid());
69    }
70    if (delta!=0) printf("Replot delay set to %d second(s)\nStarting\
71  %s, abort with ^C...\n", delta,PNAME);
72      else   printf("Waiting for signals\nStarting %s, abort with ^C...\n",PNAME);
73    signal(SIGINT, quit);
74    sprintf(command,"%s%s >%s",PPATH,PNAME,DUMP);
75  /*   strcat(command,PNAME);
76    strcat(command," >");
77    strcat(command,DUMP);*/
78    frasmol=popen(command,"w");
79    signal(SIGUSR1, plot);
80    signal(SIGUSR2, replot);
81    if (delta>0) {
82      raise(SIGUSR1);
83      sleep(delta);
84      for (;;) {
85        raise(SIGUSR2);
86        sleep(delta);
87      }
88    } else for(;;) {pause();};
89    return 0;
90  }

92  void quit() {
93    printf("SIGINT caught, exiting...\n");
94    pclose(frasmol);
95    remove(DUMP);
96    remove(pidfile);
97    exit(EXIT_SUCCESS);
98  }

100 void plot() {
101   #ifdef debug
102     printf("SIGUSR1 caught, plotting...\n");
103   #endif
104   char ts[100];
105   fplot=fopen(fnp,"r");
106   while (!feof(fplot)) {
107     fgets(ts,100,fplot);
108     fprintf(frasmol,"%s\n",ts);
109     fflush(frasmol);
110   }
111   fclose(fplot);
112 };

114 void replot() {
115   #ifdef debug
116     printf("SIGUSR2 caught, replotting...\n");
```

```
117   #endif
118   char ts[100];
119   freplot=fopen(fnr,"r");
120   while (!feof(freplot)) {
121     fgets(ts,100,freplot);
122     fprintf(frasmol,"%s\n",ts);
123     fflush(frasmol);
124   }
125   fclose(freplot);
126 };
```

C.1.2 genericpipe.h

```
1  #ifdef  gnuplot
2  #   define  PNAME   "gnuplot"
3  #   define  DUMP    "gdump"
4  #   define  PPATH   "/usr/bin/"
5  #endif

7  #ifdef  rasmol
8  #   define  PNAME   "rasmol"
9  #   define  DUMP    "rdump"
10 #   define  PPATH   "/usr/X11R6/bin/"
11 #endif
```

C.1.3 makefile

```
1  CPP = gcc
2  CPPFLAGS = -Wall -W -O2
3  PROGRAM = genericpipe
4  INSTALLDIR = ~/bin

6  default : $(PROGRAM)

8  $(PROGRAM) : genericpipe.c
9      $(CPP) $(CPPFLAGS) genericpipe.c -o gnuplotpipe -Dgnuplot
10     $(CPP) $(CPPFLAGS) genericpipe.c -o rasmolpipe -Drasmol

12 debug : genericpipe.c
13     $(CPP) $(CPPFLAGS) genericpipe.c -o gnuplotpipe -Dgnuplot -Ddebug
14     $(CPP) $(CPPFLAGS) genericpipe.c -o rasmolpipe -Drasmol -Ddebug

16 clean :
17     rm -f gnuplotpipe rasmolpipe

19 install :
20     cp gnuplotpipe $(INSTALLDIR)
21     cp rasmolpipe $(INSTALLDIR)
```

C.1.4 RasmolPlot.txt

```
1  zap
2  zap
3  zap
4  zap
5  zap
6  load pdb start.pdb
7  axes on
8  rotate x 90
9  spacefill temperature
10 backbone off
11 select *A
12 color red
```

C.1.5 RasmolReplot.txt

```
load pdb best.pdb
axes on
rotate x 90
spacefill temperature
backbone off
select *A
color red
molecule 1
zap
axes on
```

C.1.6 GnuplotPlot.txt

```
set term x11
set logscale y
a=`head -1 start.fit |awk '{print $1}'`
b=`head -1 start.fit |awk '{print $2}'`
plot 'start.dat' u 1:2:3 w errorlines t 'Experimental','start.dat' \
u 1:(a+b*$4) w l lw 2 t 'Calculated'
```

C.1.7 GnuplotReplot.txt

```
a=`head -1 best.fit |awk '{print $1}'`
b=`head -1 best.fit |awk '{print $2}'`
pt=`head -1 best.fn`
plot 'best.dat' u 1:2:3 w errorlines t 'Experimental','best.dat' \
u 1:(a+b*$4) w l lw 2 t 'Calculated'
```

Bibliography

Synchrotron radiation and SAXS theory

Baruchel, J., Hodeau, J.L., Lehmann, M.S., Regnard, J.R. & Schlenker, C., eds. (1993). *Neutron and synchrotron radiation for condensed matter studies: Theory, instruments and methods*, vol. 1. Springer-Verlag, New York.

Fanchon, E., Geissler, E., Hodeau, J., Regnard, J. & Timmins, P., eds. (2000). *Neutron and synchrotron radiation for condensed matter studies: Structure and dynamics of biomolecules*, vol. 4. Oxford University Press, New York.

Computer programming

Foley, J.D., van Dam, A., Feiner, S.K. & Hughes, J.F. (1996). *Computer graphics: principles and practice in C*. Addison-Wesley Longman Publishing Co., Boston, 2nd edn.

Press, W.H., Flannery, B.P., Teukolsky, S.A. & Vetterling, W.T. (1992). *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, New York, 2nd edn.

Hemocyanins reviews

Salvato, B. & Beltramini, M. (1990). Hemocyanins: molecular architecture, structure and reactivity of the binuclear copper active site. *Life Chem Rep*, **8**, 1–47.

van Holde, K.E. & Miller, K.I. (1995). Hemocyanins. *Adv Protein Chem*, **47**, 1–81.

References

Bannister, W.H. & Wood, E.J. (1969). Free electron model for the absorption of oxygen-bridged binuclear complexes in the near ultraviolet. *Nature*, **223**, 53–55.

Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N. & Bourne, P.E. (2000). The Protein Data Bank. *Nucl Acids Res*, **28**, 235–242.

Bernstein, F.C., Koetzle, T.F., Williams, G.J., r. Meyer EF, J., Brice, M.D., Rodgers, J.R., Kennard, O., Shimanouchi, T. & Tasumi, M. (1977). The Protein Data Bank: a computer-based archival file for macromolecular structures. *J Mol Biol*, **112**, 535–542.

Boteva, R., Severov, S., Genov, N., Beltramini, M., Filippi, B., Richelli, F., Tallandini, L., Pallhuber, M.M., Tognon, G. & Salvato, B. (1991). Biochemical and functional characterization of *Rapana thomasiana* hemocyanin. *Comp Biochem Physiol B*, **100**, 493–501.

Burmester, T. (2001). Molecular evolution of the arthropod hemocyanin superfamily. *Mol Biol Evol*, **18**, 184–195.

Burmester, T. & Scheller, K. (1996). Common origin of arthropod tyrosinase, arthropod hemocyanin, insect hexamerin, and dipteran arylphorin receptor. *J Mol Evol*, **42**, 713–728.

Cuff, M.E., Miller, K.I., van Holde, K.E. & Hendrickson, W.A. (1998). Crystal structure of a functional unit from *Octopus* hemocyanin. *J Mol Biol*, **278**, 855–870.

Debye, P. (1915). X-ray dispersal. *Ann Phys*, **46**, 809–823.

Decker, H. & Terwilliger, N. (2000). Cops and robbers: putative evolution of copper oxygen-binding proteins. *J Exp Biol*, **203**, 1777–1782.

Decker, H., Hartmann, H., Sterner, R., Schwarz, E. & Pilz, I. (1996). Small-angle X-ray scattering reveals differences between the quaternary structures of oxygenated and deoxygenated tarantula hemocyanin. *FEBS Lett*, **393**, 226–230.

Dooley, D.M., Scott, R.A., Ellinghaus, J., Solomon, E.I. & Gray, H.B. (1978). Magnetic susceptibility studies of laccase and oxyhemocyanin. *Proc Natl Acad Sci U S A*, **75**, 3019–3022.

Durstewitz, G. & Terwilliger, N.B. (1997). cDNA cloning of a developmentally regulated hemocyanin subunit in the crustacean *Cancer magister* and phylogenetic analysis of the hemocyanin gene family. *Mol Biol Evol*, **14**, 266–276.

Ellerton, H.D., Ellerton, N.F. & Robinson, H.A. (1983). Hemocyanin–a current perspective. *Prog Biophys Mol Biol*, **41**, 143–248.

Erker, W., Beister, U. & Decker, H. (2005). Cooperative transition in the conformation of 24-mer tarantula hemocyanin upon oxygen binding. *J Biol Chem*, **280**, 12391–12396.

Feigin, A.L. & Svergun, D.I. (1987). *Structure analisys by small-angle X-ray and neutron scattering*. Plenum Press, New York.

Fournet, G. (1951). Étude théorique et expérimentale de la diffusion des rayons X par les ensembles denses de particules. *Bull Soc Franc Miner Crist*, **74**, 39–113.

Free Software Foundation (1991a). GNU Compiler Collection, GCC. [Online]. Available: http://gcc.gnu.org/

Free Software Foundation (1991b). GNU Pascal - Pascal compiler of the GNU Project. [Online]. Available: http://www.gnu-pascal.org/

Freedman, T.B., Loehr, J.S. & Loehr, T.M. (1976). A resonance Raman study of the copper protein, hemocyanin. New evidence for the structure of the oxygen-binding site. *J Am Chem Soc*, **98**, 2809–2815.

Glatter, O. (1977). A new method for the evaluation of small-angle scattering data. *J Appl Cryst*, **10**, 415–421.

Glatter, O. (1980). Computation of distance distribution functions and scattering functions of models for small angle scattering experiments. *Acta Phys Austr*, **52**, 243–256.

Guinier, A. & Fournet, G. (1955). *Small angle scattering of X-rays*. Wiley, New

York.

Hartmann, H. & Decker, H. (2002). All hierarchical levels are involved in conformational transitions of the 4 x 6-meric tarantula hemocyanin upon oxygenation. *Biochim Biophys Acta*, **1601**, 132–137.

Hartmann, H., Bongers, A. & Decker, H. (2001). Small-angle neutron scattering reveals an oxygen-dependent conformational change of the immunogen keyhole limpet hemocyanin type 1 (KLH1). *Eur Biophys J*, **30**, 471–475.

Hazes, B., Magnus, K.A., Bonaventura, C., Bonaventura, J., Dauter, Z., Kalk, K.H. & Hol, W.G. (1993). Crystal structure of deoxygenated *Limulus polyphemus* subunit II hemocyanin at 2.18 Å resolution: clues for a mechanism for allosteric regulation. *Protein Sci*, **2**, 597–619.

Hellmann, N., Jaenicke, E. & Decker, H. (2001). Two types of urate binding sites on hemocyanin from the crayfish *Astacus leptodactylus*: an ITC study. *Biophys Chem*, **90**, 279–299.

Hwang, U.W., Friedrich, M., Tautz, D., Park, C.J. & Kim, W. (2001). Mitochondrial protein phylogeny joins myriapods with chelicerates. *Nature*, **413**, 154–157.

Jaenicke, E. & Decker, H. (2004). Functional changes in the family of type 3 copper proteins during evolution. *Chembiochem*, **5**, 163–169.

Kirkpatrick, S., Gelatt, C.D. & Vecchi, M.P. (1983). Optimization by simulated annealing. *Science*, **220**, 671–680.

Kitajima, N., Fujisawa, K., Tanaka, M. & Moro-oka, Y. (1992). X-ray structure of Thiolatocopper(II) complexes bearing close spectroscopic similiarities to blue copper proteins. *J Am Chem Soc*, **114**, 9232–9233.

Lang, W.H. (1988). cDNA cloning of the *Octopus dofleini* hemocyanin: sequence of the carboxyl-terminal domain. *Biochemistry*, **27**, 7276–7282.

Lang, W.H. & van Holde, K.E. (1991). Cloning and sequencing of *Octopus dofleini* hemocyanin cDNA: derived sequences of functional units Ode and Odf. *Proc Natl Acad Sci U S A*, **88**, 244–248.

Lecointre, G. & Le Guyader, H. (2001). *Classification phylogénétique du vivant*. Belin, Paris.

Leslie, A.G.W. (1992). Recent changes to the MOSFLM package for processing film and image plate data. *Joint CCP4 and ESF-EACBM Newsletters on Protein Crystallography*, **26**.

Lieb, B., Altenhein, B. & Markl, J. (2000). The sequence of a gastropod hemocyanin (HtH1 from *Haliotis tuberculata*). *J Biol Chem*, **275**, 5675–5681.

Magnus, K.A., Hazes, B., Ton-That, H., Bonaventura, C., Bonaventura, J. & Hol, W.G. (1994). Crystallographic analysis of oxygenated and deoxygenated states of arthropod hemocyanin shows unusual differences. *Proteins*, **19**, 302–309.

Markl, J. & Decker, H. (1992). Molecular structure of the arthropod hemocyanins. In *Advances in Comparative and Environmental Physiology*, vol. 13, 325–376, Springer-Verlag, Berlin.

Meissner, U., Dube, P., Harris, J.R., Stark, H. & Markl, J. (2000). Structure of a molluscan hemocyanin didecamer (HtH1 from *Haliotis tuberculata*) at 12 Å resolution by cryoelectron microscopy. *J Mol Biol*, **298**, 21–34.

Mellema, J.E. & Klug, A. (1972). Quaternary structure of gastropod haemocyanin. *Nature*, **239**, 146–150.

Menze, M., Hellmann, N., Decker, H. & Grieshaber, M. (2001). Binding of urate and caffeine to haemocyanin analysed by isothermal titration calorimetry. *J Exp Biol*, **204**, 1033–1038.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, H. & Teller, E. (1953). Equation of state calculations by fast computing machines. *J Chem Phys*, **21**, 1087–1092.

Miller, K.I., Cuff, M.E., Lang, W.F., Varga-Weisz, P., Field, K.G. & van Holde, K.E. (1998). Sequence of the *Octopus dofleini* hemocyanin subunit: structural and evolutionary implications. *J Mol Biol*, **278**, 827–842.

Mori, W., Yamauchi, O., Nakao, Y. & Nakahara, A. (1975). Spectroscopic studies on the active site of *Sepioteuthis lessoniana* hemocyanin. *Biochem Biophys*

Res Commun, **66**, 725–730.

Mouche, F., Zhu, Y., Pulokas, J., Potter, C.S. & Carragher, B. (2003). Automated three-dimensional reconstruction of keyhole limpet hemocyanin type 1. *J Struct Biol*, **144**, 301–312.

Nakamura, T. & Mason, H.S. (1960). An electron spin resonance study of copper valence in oxyhemocyanin. *Biochem Biophys Res Commun*, **3**, 297–299.

Narayanan, T., Diat, O. & Bösecke, P. (2001). SAXS and USAXS on the high brilliance beamline at the ESRF. *Nucl Instr and Meth in Phys Res A*, **467-468**, 1005–1009.

Otwinowski, Z. & Minor, W. (1997). Processing of X-ray diffraction data collected in oscillation mode. In *Methods in Enzymology: Macromolecular Crystallography, part A*, vol. 276, 307–326, Academic Press, New York.

Perbandt, M., Guthohrlein, E.W., Rypniewski, W., Idakieva, K., Stoeva, S., Voelter, W., Genov, N. & Betzel, C. (2003). The structure of a functional unit from the wall of a gastropod hemocyanin offers a possible mechanism for cooperativity. *Biochemistry*, **42**, 6341–6346.

Press, W.H., Flannery, B.P., Teukolsky, S.A. & Vetterling, W.T. (1992). *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, New York, 2nd edn.

Rayleigh, L. (1914). On the diffraction of light by spheres of small relative index. *Proc Roy Soc Lond A*, **90**, 219–225.

Riggs-Gelasco, P.J., Stemmler, T.L. & Penner-Hahn, J.E. (1995). XAFS of dinuclear metal sites in proteins and model compounds. *Coord Chem Rev*, **144**, 245–286.

Sabatucci, A. (1999). *Study of the evolutionary strategies of hemocyanins, invertebrate oxygen transport proteins*. Ph.D. thesis, University of Padova.

Salvato, B. & Beltramini, M. (1990). Hemocyanins: molecular architecture, structure and reactivity of the binuclear copper active site. *Life Chem Rep*, **8**, 1–47.

Salvato, B., Ghiretti-Magaldi, A. & Ghiretti, F. (1979). Hemocyanin of *Octopus vulgaris*. The molecular weight of the minimal functional subunit in 3 M urea. *Biochemistry*, **18**, 2731–2736.

Salvato, B., Santamaria, M., Beltramini, M., Alzuet, G. & Casella, L. (1998). The enzymatic properties of *Octopus vulgaris* hemocyanin: o-diphenol oxidase activity. *Biochemistry*, **37**, 14065–14077.

Sayle, R.A. & Milner-White, E.J. (1995). RASMOL: biomolecular graphics for all. *Trends Biochem Sci*, **20**, 374.

Solomon, E.I., Dooley, D.M., Wang, R.H., Gray, H.B., Credonio, M., Mogno, F. & Romani, G.L. (1976). Letter: Susceptibility studies of laccase and oxyhemocyanin using an ultrasensitive magnetometer. Antiferromagnetic behavior of the type 3 copper in Rhus laccase. *J Am Chem Soc*, **98**, 1029–1031.

Stoeva, S., Rachev, R., Severov, S., Voelter, W. & Genov, N. (1995). Carbohydrate content and monosaccharide composition of *Rapana thomasiana* grosse (Gastropoda) hemocyanin and its structural subunits. Comparison with gastropodan hemocyanins. *Comp Biochem Physiol B Biochem Mol Biol*, **110**, 761–765.

Stuhrmann, H.B. (1970a). Ein neues Verfahren zur Bestimmung der Oberflaechenform und der inneren Struktur von geloesten globularen Proteinen aus Roentgenkleinwinkelmessungen. *Z Phys Chem Neue Folge*, **72**, 177–198.

Stuhrmann, H.B. (1970b). Interpretation of small-angle scattering functions of dilute solutions and gases. A representation of the structures related to a one-particle scattering function. *Acta Crystallogr A*, **26**, 297–306.

Svergun, D., Barberato, C. & Koch, M.H.J. (1995). CRYSOL – a program to evaluate X-ray solution scattering of biological macromolecules from atomic coordinates. *J Appl Cryst*, **28**, 768–773.

Svergun, D.I. (1992). Determination of the regularization parameter in indirect-transform methods using perceptual criteria. *J Appl Cryst*, **25**, 495–503.

Svergun, D.I. (1997). Restoring three-dimensional structure of biopolymers from solution scattering. *J Appl Cryst*, **30**, 792–797.

Svergun, D.I. & Stuhrmann, H.B. (1991). New developments in direct shape determination from small-angle scattering. 1. Theory and model calculations. *Acta Crystallogr A*, **47**, 736–744.

Svergun, D.I., Semenyuk, A.V. & Feigin, L.A. (1988). Small-angle-scattering-data treatment by the regularization method. *Acta Crystallogr A*, **44**, 244–251.

Svergun, D.I., Volkov, V.V., Kozin, M.B. & Stuhrmann, H.B. (1996). New developments in direct shape determination from small-angles cattering. 2. Uniqueness. *Acta Crystallogr A*, **52**, 419–426.

Svergun, D.I., Volkov, V.V., Kozin, M.B., Stuhrmann, H.B., Barberato, C. & Koch, M.H.J. (1997). Shape determination from solution scattering of biopolymers. *J Appl Cryst*, **30**, 798–802.

Truchot, J.P. (1992). Respiratory function of arthropod hemocyanins. In *Advances in Comparative and Environmental Physiology*, vol. 13, 377–410, Springer-Verlag, Berlin.

Vachette, P. & Svergun, D.I. (2000). Small-angle x-ray scattering by solutions of biological macromolecules. In *Neutron and synchrotron radiation for condensed matter studies: Structure and dynamics of biomolecules*, vol. 4, chap. 11, 199–237, Oxford University Press, New York.

van Breemen, J.F., Ploegman, J.H. & van Bruggen, E.F. (1979). Structure of *Helix pomatia* oxy-beta-hemocyanin and deoxy-beta-hemocyanin tubular polymers. *Eur J Biochem*, **100**, 61–65.

van Holde, K.E. & Miller, K.I. (1995). Hemocyanins. *Adv Protein Chem*, **47**, 1–81.

van Holde, K.E., Miller, K.I. & Lang, H.W. (1992). Molluscan hemocyanins: structure and function. In *Advances in Comparative and Environmental Physiology*, vol. 13, 257–300, Springer-Verlag, Berlin.

van Holde, K.E., Miller, K.I. & Decker, H. (2001). Hemocyanins and invertebrate evolution. *J Biol Chem*, **276**, 15563–15566.

van Kuik, J.A., Kamerling, J.P. & Vliegenthart, J.F.G. (1990). *Invertebrate dioxygen carriers*. Leuven University Press, Leuven.

Voit, R., Feldmaier-Fuchs, G., Schweikardt, T., Decker, H. & Burmester, T. (2000). Complete sequence of the 24-mer hemocyanin of the tarantula *Eurypelma californicum*. Structure and intramolecular evolution of the subunits. *J Biol Chem*, **275**, 39339–39344.

Volbeda, A. & Hol, W.G. (1989). Crystal structure of hexameric haemocyanin from *Panulirus interruptus* refined at 3.2 Å resolution. *J Mol Biol*, **209**, 249–279.

Williams, T. & Kelley, C. (2005). Gnuplot homepage. [Online]. Available: http://www.gnuplot.info/

Wolfram Research, Inc. (2003). Mathematica, version 5.0. [Online]. Available: http://www.wolfram.com/

Zhu, Y., Carragher, B., Mouche, F. & Potter, C.S. (2003). Automatic particle detection through efficient Hough transforms. *IEEE Trans Med Imaging*, **22**, 1053–1062.