Journal of
Applied
Crystallography

ISSN 1600-5767

Received 1 December 2013
Accepted 1 March 2014

Two practical Java software tools for small-angle X-ray scattering analysis of biomolecules

Andreas Hofmann[a,b]* and Andrew E. Whitten[c]*

[a]Structural Chemistry Program, Eskitis Institute, Griffith University, Nathan, Queensland, Australia, [b]Faculty of Veterinary Science, University of Melbourne, Parkville, Victoria, Australia, and [c]Institute for Molecular Biosciences, University of Queensland, St Lucia, Queensland, Australia. Correspondence e-mail: a.hofmann@griffith.edu.au, a.whitten@imb.uq.edu.au

Small-angle X-ray scattering has established itself as a common technique in structural biology research. Here, two novel Java applications to aid modelling of three-dimensional macromolecular structures based on small-angle scattering data are described. *MolScat* is an application that computes small-angle scattering intensities from user-provided three-dimensional models. The program can fit the theoretical scattering intensities to experimental X-ray scattering data. *SAFIR* is a program for interactive rigid-body modelling into low-resolution shapes restored from small-angle scattering data. The program has been designed with an emphasis on ease of use and intuitive handling. An embedded version of *MolScat* is used to enable quick evaluation of the fit between the model and experimental scattering data. *SAFIR* also provides options to refine macromolecular complexes with optional user-specified restraints against scattering data by means of a Monte Carlo approach.

1. Introduction

Small-angle X-ray scattering (SAXS) has become a regularly used technique in the past decade for characterizing the structure of biological macromolecules. The fact that protein samples originally prepared for NMR or X-ray crystallographic studies are also appropriate to be subjected to SAXS experiments is one major reason for the popularity of this technique. At the same time, synchrotron facilities are increasingly establishing SAXS beamlines dedicated to structural biology, thus making the technique even more accessible.

A frequent experimental question to be addressed in structural biology is the characterization of the quaternary structure of proteins and their complexes in solution. A typical work flow for modelling such structures comprises rigid-body fitting of atomic models of individual monomers. This task is greatly aided by *ab initio* calculation of three-dimensional shapes of the scattering object, typically represented as accumulations of dummy atoms, beads or density maps (Chacón *et al.*, 1998; Svergun, 1999; Svergun *et al.*, 2001; Walther *et al.*, 2000). Using these *ab initio* shapes as guides, the individual models can in many cases be manually arranged to fill the restored volume approximately. To refine such an approximate model further, small repositioning and adjustments of the relative orientation of the individual monomers are required and the fit of the model-derived scattering intensity to the experimental data needs to be evaluated. Model adjustments can be either made manually or carried out computationally.

Here, we describe two practical Java software programs that can perform these tasks. *SAFIR* is an application that allows for the visual arrangement of protein monomers into quaternary structures. From the *SAFIR* application, the theoretical scattering intensity of a model can be computed and its fit to experimental data evaluated. This latter task is carried out by the Java program *MolScat*, which can also be used as a standalone program.

Different approaches and software tools have been developed to generate scattering intensity profiles from molecular models. *CRYSOL* is the most popular software in this context and uses multipole expansion to calculate a scattering profile based on atomic coordinates (Svergun *et al.*, 1995). In contrast, the *FoXS* server uses the Debye formula to calculate scattering profiles (Schneidman-Duhovny *et al.*, 2010), and *ORNL_SAS* performs a Monte Carlo sampling of the interatomic distances in the model (Tjioe & Heller, 2007). A different method to speed up calculation applies coarse graining by combining several atomic scatterers into a scattering unit (Grishaev *et al.*, 2005; Wriggers, 2010; Yang *et al.*, 2009).

In order to perform rigid-body modelling of multimeric complexes, any molecular graphics software can be used and the generated model, output in PDB format (Protein Data Bank; Berman *et al.*, 2000), subjected to evaluation of its small-angle scattering intensity using the software programs above. Many rigid-body modelling programs in this context focus on the computational modelling/docking aspects and rely on external graphics software for visualization and manual manipulation. *SITUS*, for example, recommends the molecular graphics program *VMD* (Humphrey *et al.*, 1996) for this purpose (http://situs.biomachina.org/tutorial_saxs.html). Similarly, *SAS_RIGID* (Meesters *et al.*, 2010) is centred around Monte Carlo computations, outsourcing the scattering intensity evaluation to *CRYSOL* or *CRYSON* (Svergun *et al.*, 1998). The program *MASSHA* offers model visualization and manipulation, but also has a built-in feature to compute the scattering intensity of the model (Konarev *et al.*, 2001).

Within our ongoing project of developing fundamental Java classes and applications for structural biology and biophysical chemistry research (Hofmann & Wlodawer, 2002), we set out to design *SAFIR*, a simple-to-use and portable Java application that aids in the modelling of quaternary protein structures using solution scattering data. During this process, it became apparent that the calculation of theoretical scattering intensities from atomic models would also need to be implemented, and we therefore developed the standalone application *MolScat*.

2. Program implementation and methods

2.1. *MolScat*: general concept

MolScat is a program to evaluate solution scattering of biological macromolecules from atomic coordinates. It can be run with terminal commands given when starting the program or through a graphical user interface. The program reads three-dimensional structures provided as PDB files and considers non-water atoms to calculate a scattering intensity curve. If experimental scattering data are provided, *MolScat* will fit the theoretical scattering curve to the experimental data. Results are provided in the form of graphical plots, an ASCII file of the theoretical scattering data (and fit with goodness-of-fit if applicable) and selected biophysical parameters.

After determining the bounding box of the model, this box is divided into a voxel grid with each voxel having a side length of 3 Å. In the grid, voxels representing the inside, surface and envelope of the protein are identified. The envelope is the first shell of unoccupied voxels around the model. From the voxel grid, the pair distance distribution function $p(r)$ for the protein is generated as a histogram and smoothed using the *KernelEstimator* class from the *WEKA* package (Hall *et al.*, 2009).

2.2. *MolScat*

To evaluate X-ray scattering, the pair distance distribution function $p(r)$ for the provided three-dimensional atomic model is generated by evaluating the electron density in each voxel (number of electrons in each voxel divided by the voxel volume, 0.027 nm^3). Water molecules in the model are automatically removed. To enable explicit corrections for excluded volume and solvation shell contrast, we have implemented a rigorous calculation of the pair distance distribution. This leads to six components of the pair distribution function, describing convolutions between the macromolecule and both envelope and solvent voxels:

$$p_1(r) = 2 \sum_{i,\text{prot}} \sum_{j,\text{prot}>i,\text{prot}} \rho(r_{i,\text{prot}})\, \rho(r_{j,\text{prot}}), \tag{1}$$

$$p_2(r) = -2\rho_\text{s} \left\{ \sum_{i,\text{prot}} \sum_{j,\text{prot}>i,\text{prot}} \left[\rho(r_{i,\text{prot}}) + \rho(r_{j,\text{prot}})\right] + \sum_{i,\text{prot}} \sum_{j,\text{env}} \rho(r_{i,\text{prot}}) \right\}, \tag{2}$$

$$p_3(r) = 2 \sum_{i,\text{prot}} \sum_{j,\text{env}} \rho(r_{i,\text{prot}})\, \rho(r_{j,\text{env}}), \tag{3}$$

$$p_4(r) = -2\rho_\text{s} \left\{ \sum_{i,\text{env}} \sum_{j,\text{env}>i,\text{env}} \left[\rho(r_{i,\text{env}}) + \rho(r_{j,\text{env}})\right] + \sum_{i,\text{prot}} \sum_{j,\text{env}} \rho(r_{j,\text{env}}) \right\}, \tag{4}$$

$$p_5(r) = 2 \sum_{i,\text{env}} \sum_{j,\text{env}>i,\text{env}} \rho(r_{i,\text{env}})\, \rho(r_{j,\text{env}}), \tag{5}$$

$$p_6(r) = 2\rho_\text{s}^2 \sum_i \sum_{j>i} 1. \tag{6}$$

Here, the subscripts i and j refer to voxels belonging to either the macromolecule (prot) or the solvent envelope (env). Hence, $\rho(r_{i,\text{prot}})$ refers to the electron density of a voxel containing atoms of the macromolecule, while $\rho(r_{i,\text{env}})$ refers to the electron density of the solvent envelope. The electron density of the bulk solvent is represented by ρ_s (default value of 334 e nm^{-3}).

The theoretical scattering intensity $I_k(q)$ is calculated by

$$I_k(q) = 4\pi \int p(r)\, [\sin(qr)/(qr)]\, \text{d}r, \tag{7}$$

(Orthaber *et al.*, 2000) for each of the six pair distance distribution functions (k = 1–6). The total scattering intensity can be calculated from the following equation which allows optimization of two scaling factors, one for the excluded volume F_prot and one for the solvation shell contrast F_env:

$$\begin{aligned} I_\text{t}(q) =& F_\text{prot}^2 I_1(q) + F_\text{prot} I_2(q) + F_\text{prot} F_\text{env} I_3(q) + F_\text{env} I_4(q) \\ &+ F_\text{env}^2 I_5(q) + I_6(q). \end{aligned} \tag{8}$$

The two scaling factors are determined by optimizing the fit of the total theoretical scattering intensity $I_\text{t}(q)$ to the experimental data $I_\text{e}(q)$. The scaling of theoretical to experimental scattering intensities and an intensity background are calculated by linear regression:

$$I_\text{e}(q) = \text{scale}\, I_\text{t}(q) + \text{background}. \tag{9}$$

The goodness of fit between the two data sets is evaluated using the χ value as defined by Svergun *et al.* (1995):

$$\chi = \left(N^{-1} \sum \left\{ \left[I_\text{e}(q) - \text{scale}\, I_\text{t}(q) - \text{background} \right] / \sigma_\text{e}(q) \right\}^2 \right)^{1/2}, \tag{10}$$

where N is the number of experimental data points included in the fit and $\sigma_\text{e}(q)$ is the experimental error. The final scattering intensity data are then calculated by spline interpolation to yield data points for each angular momentum tabulated in the user-provided experimental data file.

2.3. SAFIR

SAFIR (small-angle scattering data fitting with rigid bodies) is an application to fit small-angle scattering data with rigid-body objects, with the main purpose of modelling oligomeric structures of biological macromolecules. In the design of the program, a clear emphasis has been on an intuitive interface and ease of usage. Three-dimensional atomic protein models are therefore rendered as C$^\alpha$ traces and individual monomers are automatically coloured differently. Multimeric models can be established by loading individual monomers or by loading a PDB file with monomers being recognized by their chain identifier.

The program allows loading of individual protein monomers or oligomeric structures, which can be displayed and oriented as rigid bodies in the embedded *Jmol* molecular graphics viewer (http://jmol.sourceforge.net/). A shape object restored from small-angle scattering data can also be displayed, enabling the user to arrange the protein molecules to fit. For the loaded model, the small-angle scattering can be evaluated and compared with experimental scattering data using an embedded version of *MolScat*. Using molecular viewer features inherited from *Jmol*, the screen representation of the structures can be adjusted using the mouse. Modification of the position and orientation of one or more individual monomers is achieved by pressing the arrow keys on the number pad.

Other molecular graphics features included in *SAFIR* comprise an alignment tool, clash check and visualization. The structural alignment algorithm is based on inertia axes, thus allowing for superposition of high- and low-resolution models. The algorithm follows the concept introduced by Svergun and colleagues (Kozin & Svergun, 2001), which is based on a distance measure first introduced for polyhedron matching (Bloch *et al.*, 1993).

Rigid-body refinement of the loaded model has been implemented by means of a Monte Carlo approach which uses the fit between the model-derived and experimental scattering data as a target function. The implemented protocol runs through the following steps:

(1) Initial values for χ and R_g (radius of gyration; `RgStart`) are calculated for the starting model.

(2) A random movement of all rigid bodies activated by the user is achieved by a translation vector and three rotations (one each around the *x*, *y* and *z* axes). For this purpose, six positive random numbers are generated per rigid body to make up the translation vector and the three rotations. Another six random numbers per rigid body determine whether any of the components should be positive or negative. One further random number is required that provides a seed for the random number generation in the next cycle.

(3) The theoretical scattering of the new model is evaluated using *MolScat*. The current R_g is stored as `RgNow`. If the user has specified distance restraints, these are evaluated and a penalty of 0.1 is applied to the *MolScat*-derived χ value per violated restraint.

(4) If the χ value improves compared with the previous cycle, a clash analysis is performed on the current model if requested by the user. If more than the tolerated number of clashes are observed, the new model is discarded and the step counted as unproductive ('dead cycle'). Otherwise the model is kept and subjected to a new cycle. If the χ value does not improve compared with the previous cycle, the move is accepted with a probability that is proportional to $\exp(-1/T)$, where T is the current 'temperature' of the model (see below).

(5) If the number of unproductive cycles exceeds the user-set number of dead cycles, the shift sizes are decreased to the new value of `coolingFactor` × `oldValue`. At the same time, the number of dead cycles is increased to 1.2 times the current value. If the miminum shift size has been reached, or the number of unproductive cycles has reached the maximum number of final cycles specified by the user, the procedure will exit. Otherwise, a new cycle is started at step (2).

The algorithm accepts a bad move ($\chi_{\rm now} \geq \chi_{\rm previous}$) in step (4) above with a probability

$$p = \texttt{toleranceFactor} \exp(-d/T). \quad (11)$$

The `toleranceFactor` is a user-provided variable (default value of 0.4) and d is calculated as

$$d = \chi_{\rm now} - \chi_{\rm previous} + |\texttt{RgNow} - \texttt{RgStart}|. \quad (12)$$

2.4. Availability

Both programs make use of and extend Java classes previously developed in our laboratory (Hofmann & Wlodawer, 2002; Weeratunga *et al.*, 2012). They are available as standalone compiled Java applications from the Program Collection for Structural Biology and Biophysical Chemistry (PCSB) project home page at http://www.structuralchemistry.org/pcsb/. The *MolScat* API includes methods that enable interfacing with other Java applications and may thus also be useful to developers. The applications are freely available to academic users. For download, users will be asked for their name, institution and e-mail address. The source code is available from the authors upon request.

3. Results

3.1. Specific consequences of the voxel concept

As a consequence of the allocation of electron density into voxels, the radius of gyration R_g of the non-solvated (dry) model of lysozyme is larger than that of the solvated model (Table 1). Owing to binning of electrons into voxels of 27 Å^3 volume, the centre of mass of the binned electron density may be further from the centre of the macromolecule than the actual atom to which the electrons belong. This effect may be of particular importance at the periphery of a macromolecule where there is a lower packing density of atoms, and its impact on the overall electron-density distribution will be more pronounced in smaller molecules. Accordingly, a comparatively large R_g value is observed for the non-solvated model.

Clearly, by considering the excluded volume ($F_{\rm prot}$) and envelope contrast ($F_{\rm env}$), this effect is corrected for, and very sensible R_g values are obtained for the solvated models. In the case of lysozyme, the factor correcting for the excluded volume is close to 1 and, concomitantly, the correction factor for the electron density of the solvent layer is reduced to counteract these effects.

Conceptually, there is thus a caveat in the interpretation of $F_{\rm prot}$ and $F_{\rm env}$. Although the definition of these parameters is clear, we feel that these factors also help to correct for inadequacies in the modelling procedure.

3.2. Benchmarking of *MolScat*

Four protein systems with published small-angle scattering data have been used to compare the performance of *MolScat* with that of other generally available programs. The results are listed in Table 1 and demonstrate that *MolScat* computes scattering intensities from protein models similar to those obtained using other software. Importantly, the quality of fit between the computed and experi-

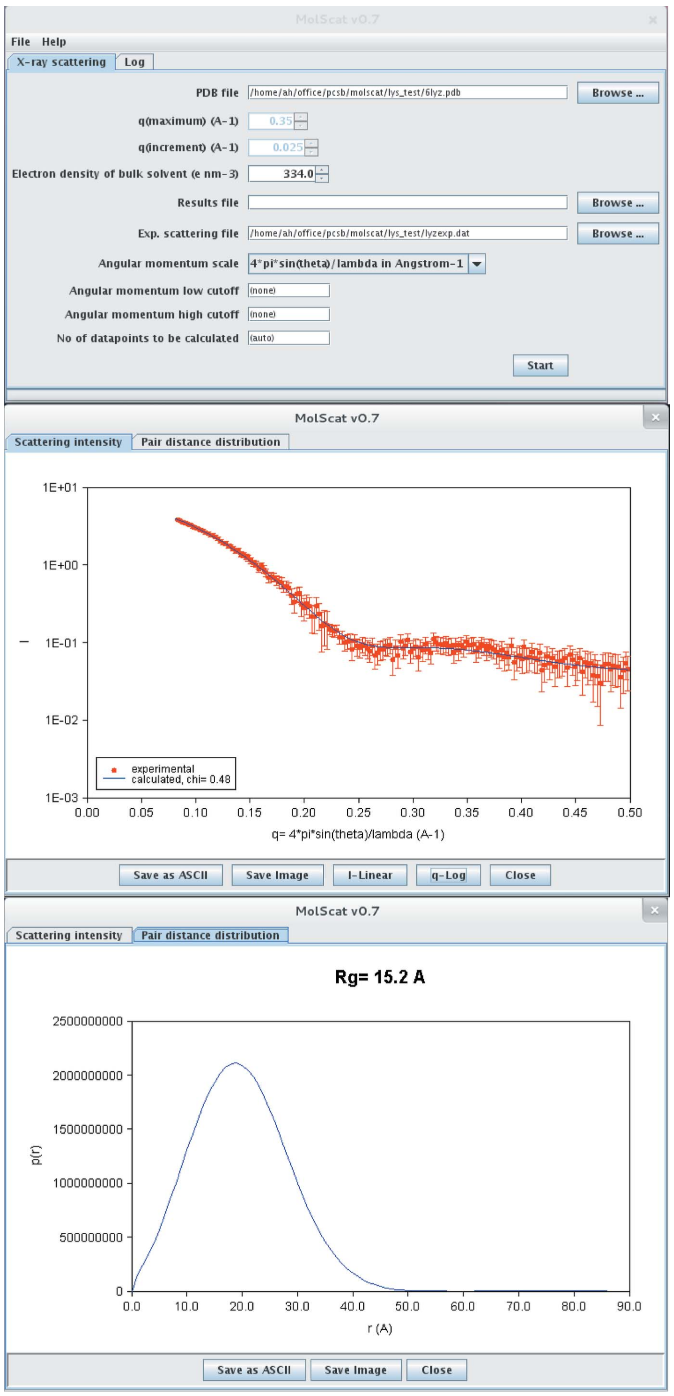
Table 1
Comparison of results obtained with different programs for fitting SAXS data.

All calculations were carried out on a Linux PC (Intel i7-2620M QuadCore, 7.7 GB RAM; Fedora Core 16.x86_64) and considered the solvation model provided by each program. *CRYSOL* (Svergun *et al.*, 1995) was obtained from http://www.embl-hamburg.de/biosaxs/crysol.html. *FoXS* and *Sastbx* are web services at http://modbase.compbio.ucsf.edu/foxs/ and http://sastbx.als.lbl.gov/cgi-bin/index.html, respectively.

	MolScat	*CRYSOL*	*FoXS*	*Sastbx* intensity
Example 1: Lysozyme (Svergun *et al.*, 1995)				
Computation time (s)	1.6	0.45	–	–
Goodness-of-fit χ	0.52	0.45	0.45	0.45
R_g of dry model (Å)	16.3	14.0	–	–
R_g of solvated model (Å)	15.4	15.0	14.0	–
Volume of dry model (Å^3)	13 716	17 350	–	–
Volume of solvated model (Å^3)	13 063	17 410	–	–
Contrast factors	1.05, 0.20	–	1.01, 0.59	–
Example 2: VILIP-1 dimer (Wang *et al.*, 2011)				
Computation time (s)	7.9	0.61	–	–
Goodness-of-fit χ	5.2	4.3	3.0	4.4
R_g of dry model (Å)	29.6	27.6	–	–
R_g of solvated model (Å)	29.8	28.7	28.2	–
Volume of dry model (Å^3)	40 500	52 000		–
Volume of solvated model (Å^3)	67 500	53 190	–	–
Contrast factors	0.60, 1.04	–	1.04, 3.09	–
Example 3: 14-3-3β dimer (Hu *et al.*, 2012)				
Computation time (s)	17	0.74	–	–
Goodness-of-fit χ	1.9	1.4	2.0	1.0
R_g of dry model (Å)	30.0	28.2	–	–
R_g of solvated model (Å)	30.2	30.4	28.2	–
Volume of dry model (Å^3)	50 463	63 090	–	–
Volume of solvated model (Å^3)	48 060	64 760	–	–
Contrast factors	1.05, 1.62	–	1.05, 4.00	–
Example 4: Glucose isomerase (Whitten, unpublished data)				
Computation time (s)	64	1.0	–	–
Goodness-of-fit χ	0.6	0.35	0.31	0.38
R_g of dry model (Å)	34.2	31.8	–	–
R_g of solvated model (Å)	33.0	33.5	31.7	–
Volume of dry model (Å^3)	170 019	213 100	–	–
Volume of solvated model (Å^3)	161 922	216 800	–	–
Contrast factors	1.05, 0.2	–	1.05, 0.16	–

mental scattering intensities is highly similar for all algorithms; an example is shown in Fig. 1. Among the tested algorithms, only *CRYSOL* is a standalone program and it therefore provides the only comparison for computing time. The four examples show that the *MolScat* calculations are slower than those of *CRYSOL*, with an exponential increase in computing time as the size of the protein system increases.

The longer computing time required by *MolScat* may result partly from the methodology chosen here. Furthermore, we have placed an emphasis on the conceptual design, and some improvements may be possible when optimizing the source code for speed. However, a substantial contribution to computing time arises from the Java programming language itself, which is known to be less time efficient than languages such as Fortran or C (Amedro *et al.*, 2008; Ashby, 2003).



PDB file: /home/ah/office/pcsb/molscat/lys_test/6lyz.pdb
q(maximum) (A-1): 0.35
q(increment) (A-1): 0.025
Electron density of bulk solvent (e nm-3): 334.0
Results file:
Exp. scattering file: /home/ah/office/pcsb/molscat/lys_test/lyzexp.dat
Angular momentum scale: 4*pi*sin(theta)/lambda in Angstrom-1
Angular momentum low cutoff: (none)
Angular momentum high cutoff: (none)
No of datapoints to be calculated: (auto)

Figure 1
Screenshot of a *MolScat* calculation fitting the lysozyme SAXS data provided by *CRYSOL* (Svergun *et al.*, 1995). (Top) The graphical user interface and parameters used for the calculation. Alternatively, the program can be invoked with terminal commands. (Middle) A plot of the fit of the model to the experimental data. This window can be suppressed by the user if invoking the program from the terminal. (Bottom) The pair distance distribution function of the model, calculated by the algorithm outlined in the text, is plotted. The graphs can be saved as binary images or ASCII data.

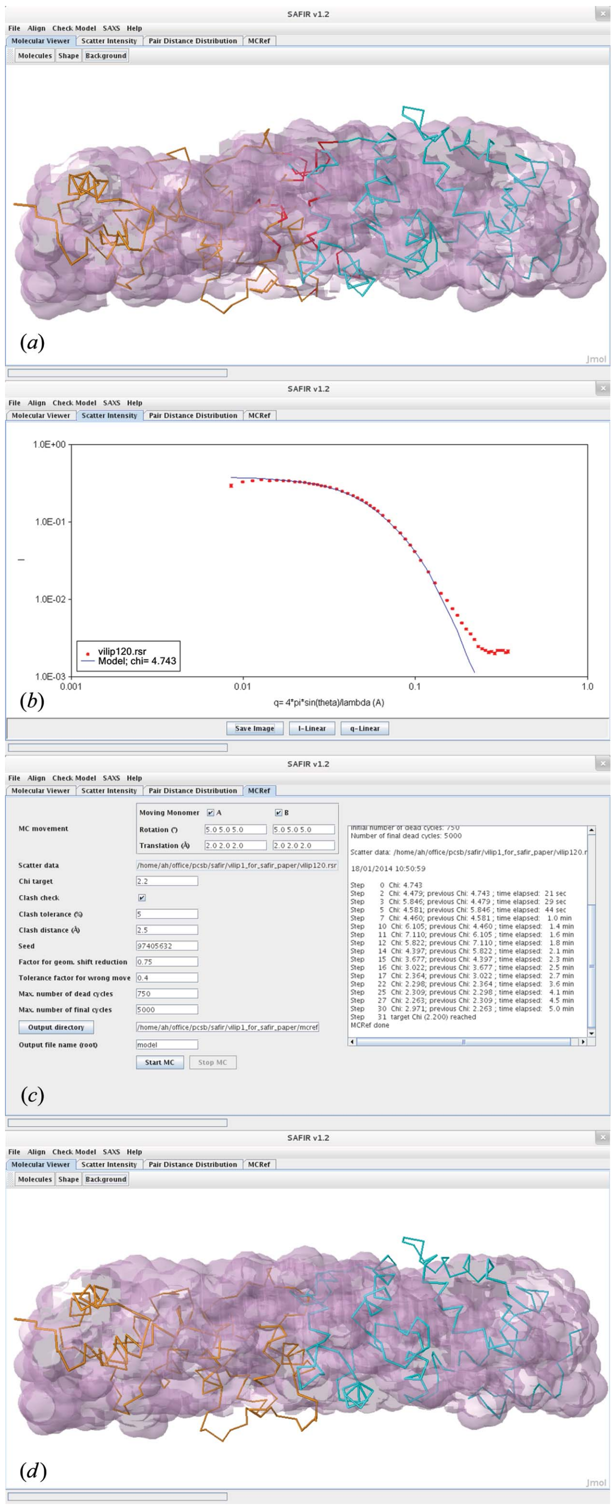
3.3. Rigid-body fitting with *SAFIR*

For illustration, we have re-worked the rigid-body fitting of the VILIP-1 homodimer published previously (Wang *et al.*, 2011). VILIP-1 belongs to the family of neuronal calcium sensor (NCS) proteins, and *GASBOR* (Svergun *et al.*, 2001) was used for *ab initio* shape restoration from SAXS data obtained from VILIP-1 in the presence of calcium and under reducing conditions. The shape obtained was distinctly different from the dimer models of other NCS proteins, suggesting that, although these proteins share a similar overall fold, they may have different molecular mechanisms. The VILIP-1 dimer model proposed in a previous modelling study (Li *et al.*, 2011) was used as the initial model and superimposed on the *GASBOR* shape in *SAFIR* (Fig. 2*a*) using the in-built superposition algorithm based on inertia axes. The model shows reasonable agreement with the SAXS data collected at a protein concentration of 12 mg ml^{-1} (Fig. 2*b*). This fit was evaluated using the *MolScat* implementation in *SAFIR* using two mouse clicks. Manual adjustments of the model yielded varying changes in the goodness-of-fit parameter χ but no significant overall improvement. Thus, the model was subjected to computational rigid-body fitting by the Monte Carlo algorithm outlined above (Fig. 2*c*). In less than 5 min of computation time, a dimer conformation close to the proposed model but with a significantly improved fit to the SAXS data was obtained (Fig. 2*d*). Distance restraints can be added before starting the computational refinement but have not been used in this example.

4. Conclusions

With *SAFIR*, we present a novel interactive modelling program that is tailored for rigid-body modelling applications and low-resolution structural data from SAXS. Similar to *MASSHA* (Konarev *et al.*, 2001), our software combines several features of low-resolution rigid-body modelling with SAXS data into one application, and we have put a strong emphasis on intuitive use. Through the built-in molecular graphics capability provided by *Jmol*, model handling and inspection are highly interactive and intuitive, and build on the experiences users have accumulated from other common molecular graphics programs such as *O* (Jones *et al.*, 1991) and *COOT* (Emsley & Cowtan, 2004). Manipulation of models is accessibly and conveniently done using the arrow keys on the number pad, omitting intermittent steps to change from viewing to transformation mode. The fit of the present model to the experimental data can be evaluated with a mouse click. Computational refinement of a model is possible in the current version of the software by means of a Monte Carlo procedure, which is a commonly chosen method for this type of refinement (Meesters *et al.*, 2010).

Within the concept of our Java PCSB project (Hofmann & Wlodawer, 2002), it is one of our main goals to have an intuitive, easy-to-use and portable application. We therefore had to implement an algorithm to compute SAXS intensities from molecular models. This


SAFIR v1.2
File Align Check Model SAXS Help
Molecular Viewer Scatter Intensity Pair Distance Distribution MCRef
Molecules Shape Background
(a)
Jmol
1.0E+00
1.0E-01
1.0E-02
1.0E-03
0.001
0.01
0.1
1.0
q= 4*pi*sin(theta)/lambda (A)
vilip120.rsr
Model; chi= 4.743
(b)
Save Image
I-Linear
q-Linear
(c)
Start MC
Stop MC
(d)

MC movement	Moving Monomer	✓ A / ✓ B
	Rotation (°)	5.0 5.0 5.0 / 5.0 5.0 5.0
	Translation (Å)	2.0 2.0 2.0 / 2.0 2.0 2.0
Scatter data	/home/ah/office/pcsb/safir/vilip1_for_safir_paper/vilip120.rsr	
Chi target	2.2	
Clash check	✓	
Clash tolerance (%)	5	
Clash distance (Å)	2.5	
Seed	97405632	
Factor for geom. shift reduction	0.75	
Tolerance factor for wrong move	0.4	
Max. number of dead cycles	750	
Max. number of final cycles	5000	
Output directory	/home/ah/office/pcsb/safir/vilip1_for_safir_paper/mcref	
Output file name (root)	model	

```
Initial number of dead cycles: 750
Number of final dead cycles: 5000

Scatter data: /home/ah/office/pcsb/safir/vilip1_for_safir_paper/vilip120.r

18/01/2014 10:50:59

Step     0  Chi: 4.743
Step     2  Chi: 4.479; previous Chi: 4.743 ; time elapsed:  21 sec
Step     3  Chi: 5.846; previous Chi: 4.479 ; time elapsed:  29 sec
Step     5  Chi: 4.581; previous Chi: 5.846 ; time elapsed:  44 sec
Step     7  Chi: 4.460; previous Chi: 4.581 ; time elapsed:   1.0 min
Step    10  Chi: 6.105; previous Chi: 4.460 ; time elapsed:   1.4 min
Step    11  Chi: 7.110; previous Chi: 6.105 ; time elapsed:   1.6 min
Step    12  Chi: 5.822; previous Chi: 7.110 ; time elapsed:   1.8 min
Step    14  Chi: 4.397; previous Chi: 5.822 ; time elapsed:   2.1 min
Step    15  Chi: 3.677; previous Chi: 4.397 ; time elapsed:   2.3 min
Step    16  Chi: 3.022; previous Chi: 3.677 ; time elapsed:   2.5 min
Step    17  Chi: 2.364; previous Chi: 3.022 ; time elapsed:   2.7 min
Step    22  Chi: 2.298; previous Chi: 2.364 ; time elapsed:   3.6 min
Step    25  Chi: 2.309; previous Chi: 2.298 ; time elapsed:   4.1 min
Step    27  Chi: 2.263; previous Chi: 2.309 ; time elapsed:   4.5 min
Step    30  Chi: 2.971; previous Chi: 2.263 ; time elapsed:   5.0 min
Step    31  target Chi (2.200) reached
MCRef done
```

Figure 2
Screenshots from a rigid-body fitting task using *SAFIR*. (*a*) The dimer model was loaded, and individual molecules were automatically recognized from their chain identifiers and coloured differently. Red indicates clashes identified by the clash check (*Check Model* menu). Molecules can be transformed (translation/rotation) separately or in groups according to the user's choice (*Molecules* button in the tool bar). The *ab initio* shape is shown as magenta spheres. (*b*) Experimental SAXS data, shown as red dots. The fit of the current model in the *Molecular Viewer* panel (blue line) can be evaluated by the embedded version of *MolScat*. (*c*) Computational rigid-body fitting was carried out using the input provided in the *MC Ref* panel. All parameter fields are populated with default values and can be changed by the user. (*d*) A view of the final model obtained after 31 steps of Monte Carlo refinement using the parameters shown in part (*c*).

application, *MolScat*, is part of the *SAFIR* modelling program but can also be used on its own.

The results generated by *MolScat* are in agreement with those of other available programs; the main difference at this stage is the longer computation time. For the frequently used approach of carrying out one-off calculations, we do not consider the longer computation time a significant problem, since even large protein systems are still processed in well under 1 min. However, time is certainly a more significant aspect in repeated executions of *MolScat*, such as for example in the Monte Carlo refinement, a feature also implemented in *SAFIR*. Given the computing power of modern CPUs and the fact that the automated refinement is carried out only for improvement of the pre-oriented model provided by the user, we do not feel that the computing time has a major impact on the benefit of this program for the user. However, in future versions of this software we will address this issue and work towards improvement of the calculation speed, *e.g.* by implementing lookup-table optimization (Wilcox *et al.*, 2011).

Additionally, future work on these programs will include implementation of neutron scattering in *MolScat*, as well as more specialized molecular modelling options in *SAFIR*, such as symmetry restraints, mixtures of oligomeric species, and generation and handling of different conformations.

AH and AEW invented and designed the algorithms, tested the programs and wrote the manuscript. AH wrote the software and manuals. Conflict of interest: none declared.

Research in the laboratory of AH is funded by the National Health and Medical Research Council (grant Nos. 1002898 and 1044022) and the Australian Research Council (grant Nos. LE120100071 and DP140100599). AEW was supported by an NHMRC Peter Doherty Fellowship (grant No. 569864). Parts of this research were undertaken on the SAXS/WAXS beamline at the Australian Synchrotron, Victoria, Australia.

References

Amedro, B., Bodnartchouk, V., Caromel, D., Delbé, C., Huet, F. & Taboada, G. L. (2008). *Current State of Java for HPC.* Rapport Technique No. 0353. Institut National de Recherche en Informatique et en Automatique, Sophia Antipolis, France. http://proactive.inria.fr/userfiles/file/papers/ProActiveJavaStatusforHPC.pdf
Ashby, J. V. (2003). *Comparison of C, Fortran and Java.* STFC Rutherford Appleton Laboratory, Didcot, UK. http://www.stfc.ac.uk/CSE/randd/arc/25040.aspx
Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalow, I. N. & Bourne, P. E. (2000). *Nucleic Acids Res.* **28**, 235–242.
Bloch, I., Maitre, H. & Minoux, M. (1993). *Pattern Recognit. Image Anal.* **3**, 137–149.
Chacón, P., Morán, F., Díaz, J. F., Pantos, E. & Andreu, J. M. (1998). *Biophys. J.* **74**, 2760–2775.
Emsley, P. & Cowtan, K. (2004). *Acta Cryst.* D**60**, 2126–2132.
Grishaev, A., Wu, J., Trewhella, J. & Bax, A. (2005). *J. Am. Chem. Soc.* **127**, 16621–16628.
Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. & Witten, I. H. (2009). *SIGKDD Explorations*, **11**, 10–18.
Hofmann, A. & Wlodawer, A. (2002). *Bioinformatics*, **18**, 209–210.
Hu, S. H., Whitten, A. E., King, G. J., Jones, A., Rowland, A. F., James, D. E. & Martin, J. L. (2012). *PLoS One*, **7**, e41731.
Humphrey, W., Dalke, A. & Schulten, K. (1996). *J. Mol. Graphics*, **14**, 33–38.
Jones, T. A., Zou, J.-Y., Cowan, S. W. & Kjeldgaard, M. (1991). *Acta Cryst.* A**47**, 110–119.
Konarev, P. V., Petoukhov, M. V. & Svergun, D. I. (2001). *J. Appl. Cryst.* **34**, 527–532.
Kozin, M. B. & Svergun, D. I. (2001). *J. Appl. Cryst.* **34**, 33–41.
Li, C., Pan, W., Braunewell, K. H. & Ames, J. B. (2011). *J. Biol. Chem.* **286**, 6354–6366.

Meesters, C., Pairet, B., Rabenhorst, A., Decker, H. & Jaenicke, E. (2010). *Comput. Biol. Chem.* **34**, 158–164.

Orthaber, D., Bergmann, A. & Glatter, O. (2000). *J. Appl. Cryst.* **33**, 218–225.

Schneidman-Duhovny, D., Hammel, M. & Sali, A. (2010). *Nucleic Acids Res.* **38**, W540–W544.

Svergun, D. I. (1999). *Biophys. J.* **76**, 2879–2886.

Svergun, D., Barberato, C. & Koch, M. H. J. (1995). *J. Appl. Cryst.* **28**, 768–773.

Svergun, D. I., Petoukhov, M. V. & Koch, M. H. (2001). *Biophys. J.* **80**, 2946–2953.

Svergun, D. I., Richard, S., Koch, M. H., Sayers, Z., Kuprin, S. & Zaccai, G. (1998). *Proc. Natl Acad. Sci. USA*, **95**, 2267–2272.

Tjioe, E. & Heller, W. T. (2007). *J. Appl. Cryst.* **40**, 782–785.

Walther, D., Cohen, F. E. & Doniach, S. (2000). *J. Appl. Cryst.* **33**, 350–363.

Wang, C. K., Simon, A., Jessen, C. M., Oliveira, C. L., Mack, L., Braunewell, K. H., Ames, J. B., Pedersen, J. S. & Hofmann, A. (2011). *PLoS One*, **6**, e26793.

Weeratunga, S. K., Hu, N. J., Simon, A. & Hofmann, A. (2012). *BMC Bioinformatics*, **13**, 201.

Wilcox, C., Strout, M. M. & Bieman, J. M. (2011). *Scientific Programming*, **19**, 213–229.

Wriggers, W. (2010). *Biophys Rev.* **2**, 21–27.

Yang, S., Park, S., Makowski, L. & Roux, B. (2009). *Biophys. J.* **96**, 4449–4463.