D+: Software for High-Resolution Hierarchical Modeling of Solution X-Ray Scattering from Complex Structures

AVI GINSBURG,^{a,b} TAL BEN-NUN,^{a,b,c} ROI ASOR,^{a,b} ASAF SHEMESH,^{a,b}

LEA FINK,^{a,b} ROEE TEKOAH,^{a,b} YEHONATAN LEVARTOVSKY,^{a,b}

DANIEL KHAYKELSON^{a,b} AND URI RAVIV ^{a,b*}

^a*Institute of Chemistry, The Hebrew University of Jerusalem, Edmond J. Safra Campus, Givat Ram, 9190401, ^bCenter for Nanoscience and Nanotechnology, The Hebrew University of Jerusalem, Edmond J. Safra Campus, Givat Ram, 9190401, Jerusalem, Israel, and ^cSchool of Computer Science and Engineering, The Hebrew University of Jerusalem, Edmond J. Safra Campus, Givat Ram, 9190401 Jerusalem, Israel. E-mail: uri.raviv@mail.huji.ac.il*

(Received 0 XXXXXXXX 0000; accepted 0 XXXXXXXX 0000)

Abstract

In this paper, we present our new computer program, D+, which uses the reciprocal-grid (RG) algorithm to efficiently compute X-ray scattering curves from solutions of complex structures at high-resolution. Structures are defined in hierarchical trees in which subunits can be represented by geometric or atomic models. Repeating subunits can be docked into their assembly symmetries, describing their locations and orientations in space. The scattering amplitude of the entire structure can be calculated by computing the amplitudes of the basic subunits on 3D reciprocal-space grids, moving up in the hierarchy, calculating the RGs of the larger structures, and by repeating this

process for all the leaves and nodes of the tree. For very large structures, a Hybrid method can be used to avoid numerical artifacts. In the Hybrid method, only grids of smaller subunits are summed and used as subunits in a direct computation of the scattering amplitude. D+ can accurately analyze both small- and wide-angle solution X-ray scattering data. We present how D+ applies the RG algorithm, accounts for rotations and translations of subunits, processes atomic models, accounts for the contribution of the solvent as well as the solvation layer of complex structures in a scalable manner, writes and accesses RGs, interpolates between grid points, computes numerical integrals, enables the use of scripts to define complicated structures, applies fitting algorithms, accounts for several coexisting uncorrelated populations, and accelerates computations using GPUs. D+ may also account for different X-ray energies to analyze anomalous solution X-ray scattering data. An accessory tool that can identify repeating subunits in a protein data bank (PDB) file of a complex structure is provided. The tool can compute the orientation and translation of repeating subunits needed for exploiting the advantages of the RG algorithm in D+. In addition, a python wrapper is also available, enabling more advanced computations and integration of D+ with other computational tools. Finally, we present a large number of tests and compare the results of D+ with other programs when possible and demonstrate the use of D+ to analyze solution scattering data from dynamic microtubule structures with different protofilament number. D+ and its source code are freely available for academic users and developers.

1. Introduction

Most involved biomolecular complexes cannot be crystallized and therefore can only be investigated in their native solution conditions. The challenge of structural biophysics is to determine the high-resolution structure of large self-assembled complexes, made

of many subunits, in their biologically relevant solution conditions. Solution small- and wide-angle X-ray scattering (SAXS/WAXS) methods are one of the important label-free and highly sensitive bulk methods for investigating the structure and interactions between complex molecular constructs (Rädler *et al.*, 1997; Koltover *et al.*, 1998; Schilt *et al.*, 2016; Dvir *et al.*, 2014; Dvir *et al.*, 2013; Chung *et al.*, 2015; Chung *et al.*, 2016; Lotan *et al.*, 2016; Ojeda-Lopez *et al.*, 2014; Moshe *et al.*, 2013; Saper *et al.*, 2012; Steiner *et al.*, 2012; Szekely *et al.*, 2011*a*; Szekely *et al.*, 2011*b*; Nadler *et al.*, 2011; Choi *et al.*, 2009; Choi *et al.*, 2016; Wong *et al.*, 2000; Deek *et al.*, 2013; Beck *et al.*, 2010; Kornreich *et al.*, 2016; Shaharabani *et al.*, 2016; Ginsburg *et al.*, 2017; Asor *et al.*, 2017; Fink *et al.*, 2017*a*). With modern synchrotron facilities the temporal and spatial resolution of these methods has been greatly improved (Ginsburg *et al.*, 2016; Kler *et al.*, 2012).

In solution X-ray scattering experiments, an X-ray beam is focused onto a solution containing structures with random orientations. As the intensities of the scattered photons are recorded rather than their amplitudes, the phase information is lost. Azimuthally integrating the intensities yields 1*D* curves of the scattered intensity as a function of q , which is the magnitude of the elastic momentum transfer (or scattering) vector, $\vec{q}$. $q = 4\pi/\lambda \cdot \sin \theta$, where λ is the X-ray wavelength, and θ is half the scattering angle. Owing to the lack of phase and 3D information (resulting from the free rotation of molecules in solution) data analysis is challenging. Challenges include addressing the contribution to the scattering signal from individual objects and from their arrangement in space, accounting for the contribution of the solvating shell, thermal fluctuations, and involve computational and model fitting aspect (see for example (Louzon *et al.*, 2017)). Multiple computer programs and algorithms have been developed over the past years for analyzing the scattering data from soluble macromolecules or oligomers (Virtanen *et al.*, 2011; Svergun *et al.*, 1995; Petoukhov

& Svergun, 2005; Bardhan *et al.*, 2009; Koutsioubas & Perez, 2013; Schneidman-Duhovny *et al.*, 2010; Schneidman-Duhovny *et al.*, 2013; Poitevin *et al.*, 2011; Knight & Hub, 2015; Franke *et al.*, 2015; Hura *et al.*, 2009; Franke & Svergun, 2009; Spinozzi *et al.*, 2014; Sarje *et al.*, 2014; Grant, 2018; Curtis *et al.*, 2012; Wright & Perkins, 2015; Gumerov *et al.*, 2012; Watson & Curtis, 2013; Ravikumar *et al.*, 2013; Petoukhov *et al.*, 2012; Schneidman-Duhovny *et al.*, 2016; Grudin *et al.*, 2017).

The program *X+* was our first attempt to address the challenges when analyzing more complicated and larger self-assembled structures (Székely *et al.*, 2010; Ben-Nun *et al.*, 2010; Ben-Nun *et al.*, 2016). Similar type of programs were developed by others (Pedersen *et al.*, 2013; Förster *et al.*, 2010; Ilavsky & Jemian, 2009). *X+* can model multilayer single geometry-based structures that may also be in Space-Filling lattices. The geometries include rectangular cuboids, a stack of layered structures, multiple spherical shells, concentric hollow cylindrical or distorted cylindrical structures, or a series of coaxial shifted helical structures. Each layer or subunit has an electron density profile that can be uniform, Gaussian, or a sum of hyperbolic tangents (Ben-Nun *et al.*, 2016). The features of *X+* include phase fitting algorithms to obtain lattice parameters and peak indices, model resolution function, and sample polydispersity. The range of models that *X+* can compute is limited. Geometrical models may fit low-angle X-ray scattering data (corresponding to relatively low-resolution information in real-space). High-quality solution X-ray scattering data at ultra-low to wide angles (Möller *et al.*, 2016), however, can be obtained using modern synchrotron facilities, high solute concentrations, proper background measurement protocols in flow-cell setups (Ginsburg *et al.*, 2016), and online size exclusion chromatography (Graewert *et al.*, 2015; Pérez & Koutsioubas, 2015).

D+ is our new computer program, which is designed to accurately compute the solution X-ray scattering curves from supramolecular structures by docking repeating

subunits into their assembly symmetry. Structures can be defined in a hierarchical manner using a data tree structure (Fig. 1). Subunits may include geometric shapes (like the models of $X+$), or atomic models when available. Atomic models are presented in protein data bank (PDB) files, containing a list of atoms and the coordinates of their real-space location. The assembly symmetry describes the position and orientation of repeating subunits. The assembly symmetry can be defined in D+ by providing lattice parameters, position, and orientation of each subunit at the graphic user interface (GUI), by uploading a file containing the information, or by writing a script, or using Lua programming language, which computes these parameters. D+ has a python application programming interface (API) that can be used to define sophisticated assembly symmetries. Subunits can be added and grouped together at any node in the hierarchical data tree structure to form a new and more involved subunit. At any level of structural complexity, identical copies of any subunit may be shifted and/or rotated in any way. The level of complexity of the elements and the entire structure can be as high as needed. Solvation layers can be computed for any assembly. For each structure or subunit, a finite domain size is directly defined or fitted to data. Advanced features including thermal fluctuations and intermolecular interactions can also be taken into account by using the python wrapper of D+ or by writing Lua scripts inside D+, which can extend, modify, or control the arrangements of subunits. The python API of D+ can also be used to integrate D+ with advanced computations or simulations, as demonstrated (Louzon *et al.*, 2017)).

In our earlier publication (Ginsburg *et al.*, 2016), we described the reciprocal-space grid (RG) hierarchical algorithm for computing solution X-ray scattering intensity curves from complex structures. We then showed results obtained with the algorithm, discussed its limits, and presented ways to address them. In particular, we showed the algorithm is slower than alternative algorithms (Svergun *et al.*, 1995) for small struc-

tures like a soluble protein. The efficiency of the RG algorithm, however, dramatically increases as the number of repeating subunits increases.

In this paper, we focus on the implementation of the RG algorithm in D+. We describe the 3D spherical grid, its mapping with a single index, how we access the mapped grid, and how we interpolate between grid points. The Direct and Hybrid methods, which should be used for very large structures, are explained. We then describe the numerical integration methods used in D+ as well as other algorithms and features that were implemented in the program, including how D+ processes atomic models in vacuum or in solution and accounts for hydrogen atoms and solvation layers of complex structures in a scalable manner, computes numerical integrals, applies fitting algorithms, analyzes anomalous X-ray scattering data, accounts for several coexisting uncorrelated populations, and accelerates computations using GPUs. Finally, we demonstrate how D+ was tested and cross validated, and how solution X-ray scattering data analysis is performed with D+. When possible, we compare D+ with other programs.

2. Applied theory

The goal of D+ is to evaluate the following solution scattering equations:

$$F(\vec{q}) = -r_0 \int \Delta\rho(\vec{r}) e^{i\vec{q}\cdot\vec{r}} d\vec{r} \quad (1)$$

$$I(q) = \langle I(\vec{q}) \rangle_{\Omega_q} = \frac{1}{4\pi} \int_{\Omega_q} |F(\vec{q})|^2 d\Omega_q, \quad (2)$$

where F is the scattering amplitude, $\vec{r}$ is the position vector in real-space, $\Delta\rho(\vec{r})$ is the electron density contrast with respect to the medium as a function of $\vec{r}$, and $r_0 = 2.82 \cdot 10^{-5} \text{Å}$ is the Thomson scattering length, which in D+ is set to 1, unless anomalous scattering option is used (see section 5.10). $I(q)$ is the orientation-averaged

scattering intensity in solution. Ω_q is the reciprocal-space solid-angle. Eq. 2 describes structures in solution that are uniformly distributed in all orientations.

The process used by D+ to compute the scattering intensity from a hierarchical assembly (Fig. 1) is described in our previous paper (Ginsburg *et al.*, 2016) and can be summarized as follows:

1. Define a 3D RG, which is essentially a lookup table of the scattering amplitudes at each scattering vector, $\vec{q}$.
2. Compute the amplitude of the lowest node (leaf) for each $\vec{q}$ point in the grid.
3. For each level of hierarchy, compute the amplitudes in the grid using the already computed lower level's amplitudes. Use cubic spline interpolation to compute values between precomputed points on the lower level grid. Save the new amplitude and discard the old.
4. Use the highest level's amplitudes to compute the scattering intensities at each point.
5. Orientation average the intensity of points on the grid with the same $|\vec{q}|$.

In the case of large or elongated assemblies, a Hybrid method, which combines direct amplitude computations with RG, should be used and can be summarized as follows:

1. Define an RG and compute the amplitude of the lowest node (leaf) for each $\vec{q}$ point in the grid.
2. For each level of hierarchy up to a predetermined level, compute the amplitude using the lower level's amplitude. For $\vec{q}$ values between precomputed points, use cubic spline interpolations.

3. At higher hierarchy levels no grids are computed anymore. Instead, the highest level of computed RG is used as a subunit leaf.
4. Repeat the last three steps for all the other leaves.
5. Flatten all the remaining higher level symmetry nodes by applying them recursively on their subunits until the tree is depth 1 (i.e., root and leaves). For each leaf, the flattening is done by determining the number of identical subunits of the highest level computed, their translations and rotations. Directly compute the scattering intensity using multiple lookups (depending on the number of subunits) of the scattering amplitude.
6. Compute orientation average by repeating the last step for each random orientation of the highest hierarchy (root) level.

In the following we present a rigorous and detailed description of how the RG algorithm is applied in D+.

2.1. Definitions and conventions

2.1.1. Coordinates $\vec{q}$ is the momentum transfer vector (or the scattering vector) given in reciprocal-space by: $\vec{q} = (q_x, q_y, q_z) = (q, \theta_q, \phi_q)$, in Cartesian and spherical coordinates, respectively. The position vector, $\vec{r}$, in real-space, is: $\vec{r} = (x, y, z) = (r, \theta, \phi)$ in the corresponding coordinates.

2.1.2. Rotation convention The convention used for rotations is the following Tait-Bryan angles (Goldstein *et al.*, 2001):

$$\mathbf{A}(\alpha, \beta, \gamma) = \mathbf{A}_x(\alpha) \cdot \mathbf{A}_y(\beta) \cdot \mathbf{A}_z(\gamma) =$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \cdot \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \cdot \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} \cos \beta \cos \gamma & -\cos \beta \sin \gamma & \sin \beta \\ \cos \alpha \sin \gamma + \cos \gamma \sin \alpha \sin \beta & \cos \alpha \cos \gamma - \sin \alpha \sin \beta \sin \gamma & -\cos \beta \sin \alpha \\ \sin \alpha \sin \gamma - \cos \alpha \cos \gamma \sin \beta & \cos \gamma \sin \alpha + \cos \alpha \sin \beta \sin \gamma & \cos \alpha \cos \beta \end{bmatrix}$$

where the rotation matrix $\mathbf{A}_i$ rotates a column vector $\vec{v}$ by an angle a_i about the i th axis, where $i \in \{x, y, z\}$ and $a_i \in \{\alpha, \beta, \gamma\}$. The rotation starts about the z -axis, followed by a rotation about the y -axis and finally about the x -axis. The rotated vector is then given by $\mathbf{A}\vec{v}$.

2.2. Rotation and translation in real- and reciprocal-space

The RG algorithm is efficient when a structure contains identical objects with different positions and orientations. We therefore need to define the relation between rotation/translation of the objects in real-space and the effect of these operations in reciprocal-space, at which scattering amplitudes are evaluated. Given an isolated object in real-space, $\vec{r}$, with an electron density contrast, $\Delta\rho(\vec{r})$ with respect to its surrounding, its scattering form-factor amplitude is given by Eq. 1. When the object is translated by a vector $\vec{R}$ and then rotated by a rotation matrix $\mathbf{A}^{-1}$, its electron density contrast is given by $\Delta\rho[\mathbf{A}^{-1}(\vec{r} - \vec{R})]$ and the form factor of the translated and rotated object is

$$\tilde{F}(\vec{q}) = -r_0 \int \Delta\rho[\mathbf{A}^{-1}(\vec{r} - \vec{R})] \exp(i\vec{q} \cdot \vec{r}) d\vec{r}.$$

We define the vector $\vec{s} \equiv \mathbf{A}^{-1} (\vec{r} - \vec{R})$ and then $\vec{r} = \mathbf{A}\vec{s} + \vec{R}$. As $\det \mathbf{A} = 1$, $d\vec{s} = d\vec{r}$. Substituting in $\tilde{F}(\vec{q})$ gives:

$$\begin{aligned} \tilde{F}(\vec{q}) &= -r_0 \int \Delta\rho[\vec{s}] \exp \left[i\vec{q} \cdot (\mathbf{A}\vec{s} + \vec{R}) \right] d\vec{s} \\ &= -r_0 \exp \left(i\vec{q} \cdot \vec{R} \right) \int \Delta\rho[\vec{s}] \exp \left[i\vec{q} \cdot (\mathbf{A}\vec{s}) \right] d\vec{s} \\ &= -r_0 \exp \left(i\vec{q} \cdot \vec{R} \right) \int \Delta\rho[\vec{s}] \exp \left[i\vec{q}\mathbf{A} \cdot \vec{s} \right] d\vec{s} \\ &= -r_0 \exp \left(i\vec{q} \cdot \vec{R} \right) \int \Delta\rho[\vec{s}] \exp \left[i\mathbf{A}^{-1}\vec{q} \cdot \vec{s} \right] d\vec{s}. \end{aligned}$$

The last equation holds because rotating the vector $\vec{s}$, given by $(\mathbf{A}\vec{s})$, or rotating the vector $\vec{q}$ by the same amount in the opposite direction, given by $(\mathbf{A}^{-1}\vec{q})$, yields the same phase (or scalar product): $\vec{q} \cdot (\mathbf{A}\vec{s}) = \mathbf{A}^{-1}\vec{q} \cdot \vec{s}$, as the magnitude of the two new vectors and the angle between them remain unchanged upon rotation (Papoulis, 1968). Note that because $\mathbf{A}$ is a rotation matrix $\mathbf{A}^{-1} = \mathbf{A}^T$. We therefore get that:

$$\tilde{F}(\vec{q}) = \exp \left(i\vec{q} \cdot \vec{R} \right) \cdot F \left(\mathbf{A}^{-1}\vec{q} \right). \quad (3)$$

D+ uses this relation when objects are translated and rotated.

2.3. Docking

In real-space, the electron density of a complex structure made of $K_{j,m}$ identical repeats of subunit j , whose orientation is m , is given by

$$\Delta\rho_{ob}^{j,m}(\vec{r}) \otimes \sum_{k=1}^{K_{j,m}} \delta \left(\vec{r} - \vec{R}_{j,m,k} \right).$$

$\Delta\rho_{ob}^{j,m}(\vec{r})$ is the subunit electron density contrast, $\otimes$ is the convolution operation, and the collection of delta functions are describing the subunit centers of mass, $\vec{R}_{j,m,k}$. In reciprocal $\vec{q}$ -space, this convolution becomes a multiplication of the two contributions. The scattering amplitude, $F_j(\vec{q})$, of an object j is given by the Fourier Transform of its electron density contrast, $\Delta\rho_{ob}^j(\vec{r})$, relative to its surroundings. The square of

this amplitude, $|F_j(\vec{q})|^2$, is the form-factor of the scattering object. The lattice sum is associated with the relative real-space arrangement of those objects.

If copy k of object j is shifted by $\vec{R}_{j,m,k}$ and rotated by a rotation matrix $\mathbf{A}_{j,m}$ with respect to its principal axes, its electron density contrast is given by

$$\Delta\rho_j \left[\mathbf{A}_{j,m}^{-1} \left(\vec{r} - \vec{R}_{j,m,k} \right) \right]$$

and its contribution to the scattering amplitude is given by the Fourier transform of the electron density contrast, which according to Eq. 3 is:

$$F_j \left(\mathbf{A}_{j,m}^{-1} \vec{q} \right) \cdot \exp \left(i \vec{q} \cdot \vec{R}_{j,m,k} \right).$$

For large assemblies, with repeating subunits (identical or not) that are shifted, rotated, and docked onto one another (Fig. 1), the scattering amplitude is:

$$F(\vec{q}) = \sum_{j=1}^J \sum_{m=1}^{M_j} \left[F_j \left(\mathbf{A}_{j,m}^{-1} \vec{q} \right) \cdot \exp \left(i \vec{q} \cdot \vec{R}_{j,m} \right) \right]. \quad (4)$$

J is the number of different types of objects, which can either be geometry-based or atomic-based models, taken, for example, from PDB files. M_j is the number of instances (rotations and translations) of object type j , given by rotation matrices $\mathbf{A}_{j,m}$ and real-space translation vectors $\vec{R}_{j,m}$. The total number of subunits, n_s , is therefore $\sum_{j=1}^J M_j$.

Another way to write Eq. 4 is:

$$F(\vec{q}) = \sum_{j=1}^J \sum_{m=1}^{M_j^u} \left[F_j \left(\mathbf{A}_{j,m}^{-1} \vec{q} \right) \cdot \sum_{k=1}^{K_{j,m}} \exp \left(i \vec{q} \cdot \vec{R}_{j,m,k} \right) \right], \quad (5)$$

where M_j^u is the number of unique orientations of object of type j , given by the rotation matrices $\mathbf{A}_{j,m}$. $K_{j,m}$ is the number of real-space translations of object j with orientation $\mathbf{A}_{j,m}$. Eq. 5 is used in the Direct and Hybrid methods, discussed in §3.2.

In §3 we describe our implementation of the RG algorithm. Numerical integration methods are explained in §4. We then explain how our atomic (§5) and geometric §6 models are computed. Ways to account for repeating subunits are explained in §7.

Uncorrelated mixed structures and key features of D+ are described in §8 and §9, respectively. Accessory tools are presented in §10. Usage examples and validation tests are provided in §12.

3. Implementation of the RG algorithm

3.1. Reciprocal grids

In our earlier paper (Ginsburg *et al.*, 2016), we provided a detailed analysis and demonstration of the RG algorithm and its efficiency. We showed that the efficiency considerably increases when the structure contains an increasing number of identical repeats of subunits. The core principle of the RG algorithm is to compute the scattering amplitudes of an object on a 3D grid (or lookup table) containing G_q points in reciprocal $\vec{q}$ -space and to store that computation for later use. When moving up in the hierarchy of the data tree structure (Fig. 1), the precomputed grid can be used to compute the scattering amplitudes of the larger structures, containing copies of the smaller object (subunit) at various locations and orientations.

Whereas the geometry of the RG can be arbitrary (Cartesian, for example), we chose to implement spherical geometry owing to the following distinct advantages. First, as the area of interest is determined by the magnitude of the scattering vector, q , using a Cartesian grid would result in “wasted” corners, which contain about half the grid points (see Supporting Online Materials, SOM, §1. Second, in order for the method to work well on GPUs, we needed a bijective relation between an index m (representing the position in the 1D array onto which the grid is mapped) and the spherical coordinates q , θ_q , and ϕ_q . Finally, orientation average in reciprocal space becomes simpler as we can fix q and average over θ_q and ϕ_q . To this end, we designed the following spherical shell based grid.

The grid origin is stored at $m = 0$. Surrounding the origin we position N evenly

spaced shells, with a spacing of $\eta_q = \frac{q_{\max}}{N}$, where $q_{\max}$ is the largest q . Note that in the GUI of D+ the parameters **Grid Size**, corresponds to $2N$, hence it should be an even number. Each shell represents the θ_q - ϕ_q plane of a given radius q . To attain the bijective relation, D+ uses a modified version of a 3D semi-uniform spherical grid, resulting in a nonuniform grid. In a semi-uniform 3D spherical grid, on the i th shell, at $q_i = i\eta_q$, half the circumference is πq_i . Hence, there are

$$J_i = \left\lceil \frac{\pi q_i}{\eta_q} \right\rceil + 1 = \lceil \pi i \rceil + 1$$

evenly spaced points along the polar axis, θ_q , where $i \in \{0, 1, \dots, N, N+1, N+2, N+3\}$. Note that for interpolation reasons (see §3.1.3), three additional shells are added to the grid, hence $i_{\max} = N+3$ rather than N . The j th polar angle on the i th shell is $\theta_q^{i,j} = \frac{j\pi}{J_i-1}$, where $j \in \{0, \dots, (J_i-1)\}$. On the j th polar angle of the i th shell, the azimuthal circumference is $2\pi q_i \sin \theta_q^{i,j}$, hence there are

$$K_{i,j} = \left\lceil \frac{2\pi q_i \sin \theta_q^{i,j}}{\eta_q} \right\rceil = \left\lceil 2\pi i \sin \left(\frac{j\pi}{J_i-1} \right) \right\rceil$$

evenly spaced points along the azimuthal axis, ϕ_q . The k th azimuthal angle of the j th polar angle on the i th shell is then given by $\phi_q^{i,j,k} = \frac{2\pi k}{K_{i,j}}$, where $k \in \{0, \dots, (K_{i,j}-1)\}$. In D+, however, to obtain simple bijective relations, the expressions for J_i and $K_{i,j}$ were changed to $J_i = id_\theta + 1$ and $K_{i,j} = id_\phi$. We chose $d_\theta = 3$ and $d_\phi = 6$ instead of π and $2\pi \sin \left(\frac{j\pi}{J_i-1} \right)$, used in the corresponding relations. The grid is nonuniform and the i th shell contains $id_\phi \cdot (id_\theta + 1)$ points in its θ_{q_i} - ϕ_{q_i} plane, which is approximately twice the density in the corresponding semi-uniform 3D spherical grid. The resulting total number of grid points is similar to that of the corresponding 3D Cartesian grid, however, all the grid points are within the sphere (see SOM §1).

Within each plane, the values are arranged in a ϕ_q -major storage order (i.e. two neighboring ϕ_q values (with the same θ_q) will be adjacent in memory, where two θ_q values with the same ϕ_q will be distanced by $6i$ other values). The total number of

reciprocal grid points, G_q^i , inside i spheres is therefore:

$$\begin{aligned} G_q^i &= \sum_{n=1}^i (d_\theta d_\phi n^2 + d_\phi n) \\ &= \frac{(3 + d_\theta) d_\phi i}{6} + \frac{(1 + d_\theta) d_\phi i^2}{2} + \frac{d_\theta d_\phi i^3}{3} \end{aligned}$$

With this, we can find the relation between any index m and three indices used to indicate the discrete values of q , θ_q , and ϕ_q (i , j , and k , respectively). First, note that for all positive i

$$\frac{d_\theta d_\phi i^3}{3} < G_q^i < \frac{d_\theta d_\phi (i+1)^3}{3},$$

hence

$$i = \left\lfloor \left(\frac{3m}{d_\theta d_\phi} \right)^{\frac{1}{3}} \right\rfloor, \quad (6)$$

unless

$$m > G_q^{i-1}$$

in which case

$$i = \left\lfloor \left(\frac{3m}{d_\theta d_\phi} \right)^{\frac{1}{3}} \right\rfloor + 1.$$

The remainder is

$$R_i = m - G_q^{i-1} - 1.$$

The last two indices are

$$j = \left\lfloor \frac{R_i}{i d_\phi} \right\rfloor = R_i \operatorname{div} (i d_\phi)$$

and

$$k = R_i - i \cdot j \cdot d_\phi = R_i \bmod (i \cdot j \cdot d_\phi).$$

It should be noted that using floating point numbers to compute $\text{pow}\left(\frac{3m}{d_\theta d_\phi}, \frac{1.0}{3.0}\right)$ to calculate i can be inaccurate. Therefore, a modified stable algorithm is used in D+ for integer values (Warren, 2012).

Given a point in reciprocal space $\{q, \theta_q, \phi_q\}$ and the values of $q_{\max}$ and N of the spherical grid, we can find the corresponding point array that is below (or equal to) the given point. Its indices would be:

$$i = \left\lfloor \frac{qN}{q_{\max}} \right\rfloor, \quad (7)$$

If $i = 0$, then $j = 0$, and $k = 0$. For $i > 0$:

$$j = \left\lfloor \frac{id_\theta \theta_q}{\pi} \right\rfloor, \quad (8)$$

and

$$k = \left\lfloor \frac{id_\phi \phi_q}{2\pi} \right\rfloor. \quad (9)$$

If, however, $k = id_\phi$ then we reset $k = 0$. We can then compute the index m . If $i = 0$ then $m = 0$. For $i > 0$:

$$m = G_q^{i-1} + i \cdot j \cdot d_\phi + k + 1. \quad (10)$$

The additions of $id_\theta + 1$ and id_ϕ in the expressions for j and k , respectively, ensure that small negative angles (owing to floating point arithmetic) are accounted for correctly.

3.1.1. Hollow Grids If the minimal q value, $q_{\min}$, is not at the origin, ($q_{\min} > 0$), a hollow grid, which covers the q -range of interest, $q_{\text{range}} = q_{\max} - q_{\min}$, should be used. The spacing between shells is $\eta_q = \frac{q_{\text{range}}}{N}$, where now N is the number of shells in the hollow grid (N is half the **Grid Size**). The number of shells in the full grid (from $q_{\min} = 0$ to $q_{\max}$), would be

$$N_{\text{full}} = \left\lceil \frac{q_{\max}}{\eta_q} \right\rceil.$$

The full and hollow grids may not coincide, to address this and to allow interpolations close to $q_{\min}$ and $q_{\max}$, three shells are added below $q_{\min}$ and above $q_{\max}$. Hence the modified $q_{\min}$ is $q'_{\min} = q_{\min} - 3\eta_q$. Following Eqs. 7, 8, and 9 we get:

$$i'_{\min} = \left\lfloor \frac{q'_{\min} \cdot N_{\text{full}}}{q_{\max}} \right\rfloor.$$

Because the first point in the grid is $\{q, \theta_q, \phi_q\} = \{q'_{\min}, 0, 0\}$, we get:

$$j'_{\min} = 0$$

and

$$k'_{\min} = 0.$$

In the full grid, the index of the first point in the hollow grid, is then given by Eq. 10:

$$\begin{aligned} m^{q'_{\min}} &= G_q^{i'_{\min}-1} + i'_{\min} \cdot j'_{\min} \cdot d_{\phi} + k'_{\min} + 1 = \\ &= \frac{(3 + d_{\theta}) d_{\phi} i'_{\min}}{6} + \frac{(1 + d_{\theta}) d_{\phi} i'^2_{\min}}{2} + \frac{d_{\theta} d_{\phi} i'^3_{\min}}{3} + 1 \end{aligned}$$

The hollow grid, is built exactly as the full grid is built, but it contains only the amplitudes that start from $m^{q'_{\min}}$. The grid points are located at $q_i = i\eta_q$, where $i \in \{i'_{\min}, \dots, N_{\text{full}} + 3\}$, $\theta_q^{i,j} = \frac{j\pi}{id_{\theta}}$, where $j \in \{0, \dots, id_{\theta}\}$ and $\phi_q^{i,j,k} = \frac{2\pi k}{id_{\phi}}$, where $k \in \{0, \dots, id_{\phi}\}$. In the hollow grid, the indexes, m , of the amplitudes are offset by $m^{q'_{\min}}$, with respect to the indexes of same amplitudes in the full grid, given by Eq. 10

$$m_{\text{Hollow Grid}} = m_{\text{Full Grid}} - m^{q'_{\min}}.$$

3.1.2. Grid density As explained in our earlier paper (Ginsburg *et al.*, 2016), to get accurate results, the grid density should be sufficiently high. To find the minimal number of grid points, we need to know the distance, L , which is the maximum between the diameter of the sphere that envelopes the structure and distance of the most distant point in the structure from the origin $(0, 0, 0)$. According to the Nyquist-Shannon

sampling rate (Shannon, 1949; Beaulieu, 2002), accurate representation using a reciprocal grids requires that the grid density satisfies $\Delta q \approx \frac{2\pi}{L}$ (Ginsburg *et al.*, 2016). The number of grid points, G , is therefore:

$$G \approx \left[(2q_{\max})^3 - (2q_{\min})^3 \right] \cdot \left(\frac{L}{2\pi} \right)^3. \quad (11)$$

D+ has a **Suggest Parameters** tool that gets $q_{\max}$, and the coordinates of the most distant point in the structure, with respect to the origin $(0, 0, 0)$, and returns the **Grid Size** parameter (or $2N$) required for D+:

$$2N = \left(\left\lfloor \frac{(q_{\max} - q_{\min}) \cdot L + 3}{10} \right\rfloor + 1 \right) \cdot 10. \quad (12)$$

where N is the number of shells in the spherical grid. In the current version, $q_{\min}$ is assumed to be zero. Eq. 12 returns a multiple of 10.

It is faster to compute the same structure after placing its center of mass at the origin. G will then depend only the dimension of the object itself rather than the size of the object plus the length of its translation vector (see Eq. 11).

3.1.3. Interpolation As we often need amplitude values that do not necessarily fall out exactly on a precomputed RG point, we need to interpolate for most values. Cubic spline interpolation is carried out serially, first ϕ_q , then θ_q , and finally q (Fig. 2). As the ϕ_q values are periodic and evenly spaced, all the n derivatives (D_i) for a given q_i and $\theta_q^{i,j}$ can be computed at once by solving (Bartels *et al.*, 1998):

$$\begin{bmatrix} 4 & 1 & & & 1 \\ 1 & 4 & 1 & & \\ & 1 & 4 & \ddots & \\ & & \ddots & \ddots & 1 \\ 1 & & & 1 & 4 \end{bmatrix} \cdot \begin{bmatrix} D_0 \\ D_1 \\ \vdots \\ D_{n-2} \\ D_{n-1} \end{bmatrix} = \begin{bmatrix} 3(F_1 - F_{n-1}) \\ 3(F_2 - F_0) \\ \vdots \\ 3(F_{n-1} - F_{n-3}) \\ 3(F_0 - F_{n-2}) \end{bmatrix},$$

where F_i are the amplitudes at each grid point and $n = K_{i,j}$. This is done once, upon computation of the RG and the derivatives are saved for later use. All subsequent

(q, θ) cubic splines are calculated on the fly from four evenly spaced $\vec{q}$ points, two above $((\vec{q}_{a_1}, F_{a_1}), (\vec{q}_{a_2}, F_{a_2}))$ and two below $((\vec{q}_{b_1}, F_{b_1}), (\vec{q}_{b_2}, F_{b_2}))$ the desired $\vec{q}$ point (where $\vec{q}_{a_2}$ is closer to $\vec{q}$ than $\vec{q}_{a_1}$ and $\vec{q}_{b_1}$ is closer to $\vec{q}$ than $\vec{q}_{b_2}$). At each interpolation, all the $\vec{q}$ points differ only in one component whereas the other two components are fixed. The amplitudes at the four points are calculated using splines from previous interpolations, using the following to calculate the derivatives.

$$\begin{bmatrix} 2 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} D_{b_1} \\ D_{b_2} \\ D_{a_1} \\ D_{a_2} \end{bmatrix} = \begin{bmatrix} 3(F_{b_2} - F_{b_1}) \\ 3(F_{a_1} - F_{b_1}) \\ 3(F_{a_2} - F_{b_2}) \\ 3(F_{a_2} - F_{a_1}) \end{bmatrix}$$

gives

$$\begin{bmatrix} D_{b_1} \\ D_{b_2} \\ D_{a_1} \\ D_{a_2} \end{bmatrix} = \frac{1}{15} \begin{bmatrix} -19F_{b_1} + 24F_{b_2} - 6F_{a_1} + F_{a_2} \\ -7F_{b_1} - 3F_{b_2} + 12F_{a_1} - 2F_{a_2} \\ 2F_{b_1} - 12F_{b_2} + 3F_{a_1} + 7F_{a_2} \\ -F_{b_1} + 6F_{b_2} - 24F_{a_1} + 19F_{a_2} \end{bmatrix}.$$

The amplitude, F at $\vec{q}$ values between points $\vec{q}_{b_2}$ and $\vec{q}_{a_1}$ are calculated as

$$F(\vec{q}) = F_{b_2} + D_{b_2}t + [3(F_{a_1} - F_{b_2}) - 2D_{b_2} - D_{a_1}]t^2 + [2(F_{b_2} - F_{a_1}) + D_{b_2} + D_{a_1}]t^3$$

where

$$t = \frac{|\vec{q} - \vec{q}_{b_2}|}{|\vec{q}_{a_1} - \vec{q}_{b_2}|}$$

A crude error estimate of the cubic interpolation is determined by the RG density, $\approx \eta_q^3$, which dictates an error of order $\mathcal{O}(\eta_q^4)$ (Sonneveld, 1969).

Three additional shells beyond $q_{\max}$ are added to the grid, so that interpolations can be performed all the way up to $q_{\max}$. When the required values fall between the first and the second shells, the grid amplitudes that are used for interpolations are taken from the origin ($q = 0$), the first, the second, and the third shells. When the required values fall between the origin and the first shell, the grid amplitudes that are used for interpolations, are taken from the first and second shells, the origin, and the points on

the first shell, whose azimuthal and polar angles are closest to

$$(\pi + \phi_q) \mod 2\pi,$$

and

$$(\pi - \theta_q).$$

3.2. The Direct and Hybrid methods

For large self-assembled structures the scattering amplitude becomes highly oscillatory in $\vec{q}$ -space. As a result the required RG density (or RG size, G_q) becomes too high (Shannon, 1949; Beaulieu, 2002). D+ has two ways to overcome this issue: the Direct and the Hybrid methods. For both methods, the number of computations or lookups can be reduced by identifying identically oriented objects and for each only multiplying by the phase factor associated with its unique position (Eq. 5).

3.2.1. The Direct Method In the Direct method, no RG is used. Instead, the amplitude is directly computed at each one of the required points in $\vec{q}$ -space. Each geometry or PDB object is identified and all its copies (orientations and locations) in space are collected. The intensity is then computed as:

$$I_N(q) = \frac{1}{N} \sum_{i=1}^N \left| \sum_{j=1}^J \sum_{m=1}^{M_j^u} F_j \left(\mathbf{A}_{j,m}^{-1} \vec{q}_i \right) \cdot \sum_k^{K_{j,m}} \exp \left(i \vec{q}_i \cdot \vec{R}_{j,m,k} \right) \right|^2 \quad (13)$$

where J is the number of different objects (or the total number of different leaves, see Fig. 1). M_j^u is the number of unique rotations $\mathbf{A}_{j,m}$ of object j . $K_{j,m}$ is the number of copies of object j with orientation m that were translated in real-space by vectors $\vec{R}_{j,m,k}$. Orientation average is done according to Eq. 17. The total number of directly calculated subunits is

$$n_s = \sum_{j=1}^J \sum_{m=1}^{M_j^u} K_{j,m}.$$

3.2.2. The Hybrid Method In the Hybrid method, which is implemented in D+, RGs are combined with the Direct method. All geometries and PDB files (in other words, all the leaves in the hierarchical data tree structure, see Fig. 1) are calculated to grids. Grids of assembly symmetries are computed up to predetermined nodes in the hierarchical data tree structure (Fig. 1). Once grids of assembly symmetries are computed, their children are discarded and not used. Assembly symmetries at higher hierarchy levels continue as in the Direct method (Eq. 13) but use the last computed RG for looking up values once per copy of those structures at each one of the required points in $\vec{q}$ -space.

The evaluation is the same as in the Direct method, except that if RGs are computed, $F_j \left(\mathbf{A}_{j,m}^{-1} \vec{q}_i \right)$ is retrieved from the grid, using interpolations (see §3.1.3), and not directly computed. In other words, RGs are computed for all leaves; grids may be calculated for internal nodes and if they are then the node's leaves are discarded and the internal node is treated as a leaf. The new number of leaves, J , in the data tree structure does not include the discarded grids. The Hybrid method can retain near atomic resolution while significantly reducing the computational times. A more comprehensive analysis of the Hybrid method is provided in our earlier publication (Ginsburg *et al.*, 2016). Validation tests of the Direct and Hybrid methods are shown in SOM §9.1 and §9.2.

4. Numerical integration methods

To compute the scattering intensity of the assembled structure the squared norm of the final amplitude (or the amplitude of the root in the data tree structure, Fig. 1) is calculated. In solution, we have to average over all the orientations of the structure in $\vec{q}$ -space, hence the scattering intensity is:

$$I(q) = \frac{\int_0^{2\pi} d\phi_q \int_0^\pi d\theta_q |F(\vec{q})|^2 \sin \theta_q}{\int_0^{2\pi} d\phi_q \int_0^\pi d\theta_q \sin \theta_q}. \quad (14)$$

where θ_q and ϕ_q are the polar- and azimuthal-angles in reciprocal space, respectively (see §2.1.1). The actual integration is done numerically using one of the methods below.

Some numerical integration methods require generation of random numbers. Pseudo-random numbers were generated by the Mersenne Twister algorithm that has a period of $2^{19937} - 1$ (instead of 2^{32} in the `rand()` function of C++). Validation tests of the integration methods used in D+ are shown in SOM §9.1 and §9.2.

4.0.1. Classic Monte-Carlo integration To compute Eq. 14, random ϕ_q^i and θ_q^i angles should be generated from which the vectors $\vec{q}_i = (q, \theta_q^i, \phi_q^i)$ can be generated. The orientation average:

$$\begin{aligned} I_N(q) &= \frac{\sum_{i=1}^N |F(\vec{q}_i)|^2 \sin(\theta_q^i)}{\sum_{i=1}^N \sin(\theta_q^i)} \\ &= \frac{\pi}{2N} \sum_{i=1}^N |F(\vec{q}_i)|^2 \sin(\theta_q^i), \end{aligned}$$

needs to be computed until N is large enough, between the limits $N_{\min}$ and $N_{\max}$, so that for each q we get:

$$\left| 1 - \frac{I_N(q)}{I_f(q)} \right| \leq \varepsilon, \quad (15)$$

where ε is a small number that defines our error. We chose $f = N - p \cdot k$ where $k \in \{1, 2, 3, 4\}$, where $p = 100$ for CPU computations and 8192 for GPU computations. In practice, computations are often done up to $N_{\min}$, and $I_{N_{\min}}$ is saved. Each time the number of iterations has increased by p from the previously saved curve, another curve is saved to a matrix, which can contain up to 4 curves. After the first 4 curves are saved, D+ checks if Eq. 15 is satisfied. If it is, the computation ends, if not the number of iterations is increased by an additional p iterations and the last curve is saved. The curves are saved on a first in - first out basis. Each time a new curve is saved, the saved intensity curves are used to check if Eq. 15 is satisfied.

To uniformly sample the polar and azimuthal angles we use:

$$\begin{aligned}\phi_q^i &= 2\pi \cdot u \\ \theta_q^i &= \cos^{-1}(2 \cdot v - 1)\end{aligned}\tag{16}$$

where u and v are random variates in the $[0, 1]$ range (Weisstein, 2015). The intensity is then computed by:

$$I_N(q) = \frac{1}{N} \sum_{i=1}^N |F(\vec{q}_i)|^2\tag{17}$$

4.0.2. VEGAS: an adaptive Monte-Carlo integration As with numerical integrations in general, integrating the Fourier space intensity does not always converge quickly. In some cases (see later) using an adaptive method results in quicker convergence. To that end, we have included two adaptive integration methods in D+.

In 1978, Lepage (Peter Lepage, 1978) introduced an effective method for a biased sampling for the Monte-Carlo integration that its calculation-cost grows linearly with the dimension of the integral. The integration domain is divided into bins. After each step of N evaluations, the bins are resized so that the variance within each bin is roughly the same. This method leads to a faster convergence when certain areas of the integration domain fluctuate faster than others. In the case of scattering data, objects that are long in one or two dimensions (and shorter in the remaining dimensions) have this characteristic and then adaptive integration method can be helpful. The VEGAS method is implemented for the GPU in D+.

Implementation We divide the integration space from Eq. 16 (u and v) into N_b bins, each with a not necessarily equal volume V_{bin}^k . For each bin, the same number of intensity points, ip , are evaluated. The variance in each bin is calculated with the following algorithm.

Per bin, the difference between the i th (unit offset, $i = 1, 2, \dots, ip$) evaluation and

the previous mean is:

$$\Delta_i = V_{\text{bin}}^k \cdot I_i - \langle M \rangle_{i-1}$$

The i th mean is then

$$\langle M \rangle_i = \frac{\Delta_i}{i} + \langle M \rangle_{i-1}$$

and the i th variance is

$$\sigma_i^2 = \Delta_i^2 \left(\frac{i-1}{i} \right) + \sigma_{i-1}^2$$

In the case where there are multiple q values calculated at once between two shells (see §9.2.3)

$$\sigma_i^2 = \sum_q^{N_q} \sigma_{i,q}^2,$$

where N_q is the number of q points between the shells. The boundaries are then adjusted such that the variance in the variance is minimized. Convergence is determined by Eq. 15.

4.0.3. Adaptive Gauss-Kronrod An alternative integration method that can be used in D+ is the adaptive Gauss-Kronrod quadrature algorithm. The integral is evaluated as a weighted sum at selected points. The Gauss Kronrod method used is a 7-point Gauss rule with a 15-point Kronrod rule (G7,K15) (Kronrod, 1964; Laurie, 1997). The error estimate is given by the relative difference between the two:

$$\left| 1 - \frac{G_7}{K_{15}} \right| < \epsilon \quad (18)$$

Note that the **Integration Iterations** in the GUI, is in fact the maximum recursion depth and should be in the order of 10 or 15. This method is implemented only for the CPU. It should be used for structures that have between one and three large dimensions.

5. Atomic models

To calculate the scattering amplitude from a molecule, we take its PDB file representation. Atom identification is based on characters 77,78 of every ATOM/HETATM entry (PDB v2.0 and greater) (Berman *et al.*, 2014). If an older file format is used, D+ attempts to identify the atom based on characters 13,14 (no charge is assumed in this case) (Bernstein *et al.*, 1977). Other deviations from the standard PDB file format should be amended (or removed from the file) before loading the file into D+. The formal charge is taken from characters 79,80 (for example, Mg²⁺). The formal charge changes the atomic scattering amplitude and can therefore be important (see §5.3). The coordinates of the j th atom are read into $\vec{r}_j$ and an index that represents the type of atom/ion is saved. There is an index for each of the 209 atoms and ions listed in the International Tables for Crystallography (Hamilton, 1974; Marsh & Slagle, 1983).

5.1. Centering

As the center of mass of many PDB files is not at the origin, there is an option to find the center of mass and translate the entire object in the opposite direction so that the origin coincides with the center of mass, $\vec{r}_{c.m.} = \frac{\sum_j \vec{r}_j \cdot m_j}{\sum_j m_j}$, where m_j is the atomic mass of the j th atom. The vector $\vec{r}_{c.m.}$ is then subtracted from the coordinate of each atom. Centering the molecule is enabled by default in D+, but can be disabled before adding a PDB file. Objects that are off centered have an inherent phase in their scattering amplitude. Hence, it is better to compute the same structure after its center of mass is at the origin (see §3.1.2).

5.2. Computing the scattering amplitude from solvated atomic models

The solution scattering amplitude from PDB structures can be computed by using a combination of the following contributions. There are two options to evaluate the

scattering amplitude. One option uses Dummy Atom Gaussian spheres to approximate the volume of excluded solvent by the atoms:

$$F(\vec{q}) = a \cdot F_{\text{mol}}^v(\vec{q}) - \rho_0 \cdot f_{\text{Excluded Solvent}}^{\text{Dummy Atom}}(\vec{q}) + (\rho_{\text{Solvation Layer}} - \rho_0) \cdot F_{\text{Solvation Layer}}(\vec{q}). \quad (19)$$

Alternatively, the volume of excluded solvent can be taken into account as a collection of voxels:

$$F(\vec{q}) = a \cdot F_{\text{mol}}^v(\vec{q}) - \rho_0 \cdot F_{\text{Excluded Solvent}}^{\text{Voxel}}(\vec{q}) + (\rho_{\text{Solvation Layer}} - \rho_0) \cdot F_{\text{Solvation Layer}}(\vec{q}). \quad (20)$$

F_{mol}^v is explained in §5.3 (Eq. 22), $f_{\text{Excluded Solvent}}^{\text{Dummy Atom}}$ in §5.5 (Eq. 27), $F_{\text{Excluded Solvent}}^{\text{Voxel}}$ in §5.9 (Eq. 29), and $F_{\text{Solvation Layer}}$ in §5.8 (Eq. 31). a is equal to 1 unless **Solvent Only** is indicated in D+, in which case $a = 0$. If a is set to 0 and, in addition, the mean electron density of the solvent, ρ_0 , is set to 0, we get the contribution to the scattering amplitude only from the solvation layer. By loading a PDB and computing Eq. 19 using finite value for $\rho_{\text{Solvation Layer}}$ and ρ_0 , the solvent contribution to the scattering is computed based on the radii taken from (Fraser *et al.*, 1978; Slater, 1964; Svergun *et al.*, 1995). The volume of excluded solvent and solvation layer are then computed based on the chosen radii.

If $\rho_{\text{Solvation Layer}}$ is set to 0, when a PDB file is loaded, D+ does not add any solvation layer. If the same PDB is reloaded and only the contribution from the solvation layer is computed (by setting $a = 0$, $\rho_0 = 0$, and $\rho_{\text{Solvation Layer}}$ to the mean electrons density contrast in the solvation layer, with respect to the bulk solvent), any other method for calculating the solvation layer can be applied and its contribution can be added to the contributions of the vacuum and dummy atom terms. Splitting the computation in this mode has another advantage. Fitting the value of the solvation layer contrast ($\rho_{\text{Solvation Layer}} - \rho_0$) becomes faster, as it is equivalent to setting $\rho_{\text{Solvation Layer}} = 1$ and fitting the scale factor of the amplitude, and hence does not require a new grid computation for the voxels of the solvation layer. In SOM §10, we demonstrate

how D+ can be used to compute the contribution to the scattering amplitude from solvation layers of complex structures in a scalable manner. The approach is based on computing the amplitude of the solvated isolated subunits and correcting for spatial overlaps between solvated subunits when in the complex. The corrections are based on the amplitudes from solvated isolated pairs of subunits within the complex and the solvated isolated subunits. The corrected solvated subunits are then docked into the complex structure. If the same subunit has different solvation overlap corrections then its original assembly symmetry should be divided into smaller assembly symmetries, based on the different solvation overlap corrections. More details and examples are given in SOM §10.

5.3. PDB (*in vacuo*)

The scattering amplitude of an atom j , in units of $-r_0$, is calculated using the 5 Gaussian approximation atomic form factor expression (Als-Nielsen & McMorrow, 2011; Grudin *et al.*, 2017):

$$f_j^0(q) = \sum_{k=1}^4 a_k \cdot \exp\left(-b_k \left(\frac{|\vec{q}|}{4\pi}\right)^2\right) + c. \quad (21)$$

a_k, b_k , and c are the Cromer-Mann coefficients (Hamilton, 1974; Marsh & Slagle, 1983), given in units of the Thomson scattering length, $r_0 = 2.82 \times 10^{-5}$ Å. Note that in D+, however, r_0 is set to 1.

Given a point in reciprocal space $\vec{q}$, and a list of atoms and their coordinates (as in PDB files), the scattering amplitude of the entire molecular structure, containing n atoms, is given by:

$$F_{\text{mol}}^v(\vec{q}) = \sum_{j=1}^n f_j^0(q) \cdot \exp(i\vec{q}\vec{r}_j) \quad (22)$$

where $\vec{r}_j$ is the location in real-space of the j th atom.

5.4. Displaced solvent from atomic models

The displaced solvent contribution may be calculated in multiple ways (Fedorov *et al.*, 1972; Pavlov & Fedorov, 1983; Fraser *et al.*, 1978; Svergun *et al.*, 1995; Park *et al.*, 2009; Bardhan *et al.*, 2009; Poitevin *et al.*, 2011; Koutsioubas & Perez, 2013; Schneidman-Duhovny *et al.*, 2013; Knight & Hub, 2015; Grudin *et al.*, 2017). One way is to subtract Gaussian dummy-atoms localized at the center of each atom (Fraser *et al.*, 1978; Svergun *et al.*, 1995; Förster *et al.*, 2008; Schneidman-Duhovny *et al.*, 2010; Schneidman-Duhovny *et al.*, 2013; Grudin *et al.*, 2017); another way is to determine the displaced volume and shape using voxels (Pavlov & Fedorov, 1983; Bardhan *et al.*, 2009; Virtanen *et al.*, 2011). Both methods can be used in D+. Using the python API of D+ and computer simulations, more accurate methods can be applied (see, for example, (Knight & Hub, 2015)).

5.5. Solvent as Gaussian dummy-atoms

The mean atomic volume $V_m = \sum_j V_j$ and mean atomic radius $r_m = \left(\frac{3}{4\pi} V_m\right)^{1/3}$ are computed based on the list of atoms or atomic groups in the PDB file. $V_j = \frac{4\pi}{3} r_j^3$ is the approximated volume of excluded solvent by the j th atom (or atomic group) and r_j is the published experimental atomic radius of the j th atom (or atomic group) (Fraser *et al.*, 1978; Svergun *et al.*, 1995; Grudin *et al.*, 2017). When absent, the r_j values were replaced by empirical radii (Slater, 1964). To match the radii used in CRY SOL (ATSAS 2.8.2), we modified the carbon and nitrogen radii to be 1.577 Å and 0.8414 Å, receptively.

A Gaussian dummy-atom is placed at the center of each atom in the PDB file. The electron density of the Gaussian sphere of atom j is:

$$\rho_j(\vec{r}) = \rho_0 \left(\frac{r_j}{r_m}\right)^3 \cdot \exp\left[-\left(\frac{9\pi}{16}\right)^{\frac{1}{3}} \frac{|\vec{r}|^2}{r_m^2}\right] = \rho_0 \frac{V_j}{V_m} \cdot \exp\left[-\pi V_m^{-\frac{2}{3}} |\vec{r}|^2\right]$$

where ρ_0 is the mean electron density of the bulk solvent (for example, $\rho_0^{\text{water}} =$

334 e/nm³). The Gaussian electron density profile is normalized so that the total number of excluded electrons, which is given by $\int_{d\vec{r}^3} \rho_j(\vec{r}) d\vec{r}^3 = \rho_0 \cdot V_j$, is equivalent to that of a uniform sphere of volume V_j . The reason for using the mean atomic radius, r_m (or the mean atomic volume V_m), in the exponent is to be able to uniformly adjust (and account for) the volume of excluded solvent throughout the entire structure (using Eq. 24). V_m can be slightly varied to better fit experimental data. The scattering amplitude contribution of the Gaussian dummy atom j is then:

$$F_j^d(\vec{q}) = \rho_0 \cdot \frac{V_j}{V_m} \cdot \int_0^{2\pi} d\phi_r \int_0^\pi d\theta_r \int_0^\infty \exp \left[- \left(\frac{9\pi}{16} \right)^{\frac{1}{3}} \frac{r^2}{r_m^2} \right] \cdot e^{i\vec{q} \cdot \vec{r}} \cdot r^2 \sin \theta_r dr$$

The result depends only on the magnitude of the atomic radius (or the atomic volume, V_j), owing to the spherical symmetry of $\rho_j(\vec{r})$, and is given by:

$$F_j^d(q) = \rho_0 \cdot V_j \cdot \exp \left[- \frac{V_m^{2/3} \cdot q^2}{4\pi} \right]. \quad (23)$$

This expression reproduces the calculations of CRY SOL (ATSAS 2.8.2), which has been extensively used and shown to adequately fit experimental data (Förster *et al.*, 2008; Schneidman-Duhovny *et al.*, 2010; Schneidman-Duhovny *et al.*, 2013; Svergun *et al.*, 1995; Grudinin *et al.*, 2017).

D+ accepts both positive and negative values of bulk solvent electron density, ρ_0 . Negative ρ_0 is nonphysical. However, when $\rho_0 > 0$ the contribution of F_j^d is subtracted from other amplitudes (see Eq. 25). Negative ρ_0 values can only be used to added the contribution of the excluded solvent, F_j^d (or Eq. 30), to other amplitudes. ρ_0 is also subtracted from the the electron density of the solvation layer (see Eqs. 19, 20, or 31). If there is a solvation layer, negative ρ_0 means that the magnitude of ρ_0 will be added to the electron density of the solvation layer.

To uniformly adjust V_m , each F_j^d is multiplied by:

$$C_1(q) = c_1^3 \cdot \exp \left[- \frac{V_m^{2/3} \cdot q^2 (c_1^2 - 1)}{4\pi} \right]. \quad (24)$$

The default value of c_1 is 1, corresponding to $C_1(q) = 1$. By slightly varying the value of c_1 , the mean volume of excluded solvent by the atoms is adjusted to better fit experimental data, as done in FoXS (Förster *et al.*, 2008; Schneidman-Duhovny *et al.*, 2010; Schneidman-Duhovny *et al.*, 2013), or CRY SOL (Svergun *et al.*, 1995), or Pepsi-SAXS (Grudin *et al.*, 2017). The contribution of atom j to the scattering amplitude in solution is then:

$$f_j^s(q) = f_j^0(q) - C_1(q) \cdot F_j^d(q). \quad (25)$$

The solution scattering amplitude from a molecule, given a list of n atoms, whose coordinates are $\vec{r}_i$, is:

$$F_{\text{mol}}^s(\vec{q}) = \sum_{j=1}^n f_j^s(q) \cdot \exp(i\vec{q}\vec{r}_j), \quad (26)$$

and

$$\rho_0 \cdot f_{\text{Excluded Solvent}}^{\text{Dummy Atom}} = C_1(q) \cdot \sum_{j=1}^n F_j^d(q) \cdot \exp(i\vec{q}\vec{r}_j) \quad (27)$$

5.6. Atomic groups

The PDB files may contain atomic groups. The type of atomic group is determined by the 14th, 15th, and sometimes 16th columns (labeled as CA, CB, or CG, for example, corresponding to C_α , C_β , or C_γ) and the 18th to 20th columns that contain the type of amino acid or nucleic acid. This combination determines the type of atomic group (CH, CH₂, or CH₃ in the above example). Other atomic groups are listed in Table 1 in (Svergun *et al.*, 1995). If water molecules are included in a PDB file, D+ will compute their contribution to the scattering curve.

5.7. Implicit and explicit hydrogen atoms

If hydrogen atoms are included or added to the PDB file (Chen *et al.*, 2010), D+ will explicitly compute their contribution to the scattering amplitude. The amplitude of the hydrogen atoms is computed based on their position, as listed in the PDB file. In that case, only the contribution from the heavy atom of each atomic group is taken into account (to avoid double counting of hydrogen atoms).

If, however, no hydrogen atoms are included in the PDB file, the contribution of hydrogen atoms will be implicitly taken into account using a five Gaussian approximation (Eq. 21), whose coefficients were obtained as follows. Firstly, the amplitude in vacuum of each type of atomic group was computed by explicitly adding the hydrogen atoms (Chen *et al.*, 2010) to the heavy atom of the atomic group. The result is a sum of Gaussian functions. Secondly, the coefficients of the five Gaussian approximation that best fitted each atomic group amplitude were found (see Table 1), stored, and used when needed. Other combination of coefficients, however, may also fit (Grudin *et al.*, 2017).

Solvent subtraction is done according to Eq. 23, using the atomic group radii from Table 1 in (Svergun *et al.*, 1995), with the following adjustments to match the radii used in CRY SOL (ATSAS 2.8.2). The atomic group radius, R_{AG} , was obtained by summing the heavy atom volume, V_{HA} , and hydrogen atom volumes, $n_H \cdot V_H$, where n_H is the number of hydrogen atoms in the atomic group and V_H is the volume of a hydrogen atom. The atomic group volume is therefore: $V_{AG} = V_{HA} + n_H \cdot V_H$, and its radius is: $R_{AG} = \left(\frac{3V_{AG}}{4\pi}\right)^{\frac{1}{3}}$, in agreement with (Svergun *et al.*, 1995). Fig. S3 and the figures in SOM §9.2 show that the results of D+ are in agreement with those of CRY SOL (ATSAS 2.8.2) for atomic models in solution with either implicit or explicit hydrogen atoms.

5.8. Voxalized solvent

In this method, we equally divide the space occupied by the molecule into voxels of a predetermined size, v (whose default value is $v = 0.2$ nm, which is smaller than used by Pepsi-SAXS (Grudin *et al.*, 2017)). For each voxel, we determine whether it contains an atom (or part of one) or not by applying the following algorithm.

1. **Allocate space.** The occupied space is defined by going over every atom i , taking its center coordinates $\left(\mathbf{c}^i = \{c_x^i, c_y^i, c_z^i\}\right)$, adding and subtracting the atomic radius (r^i) to each coordinate, and selecting the minimum and maximum in each of the three dimensions, $\min(c_j^i - r^i)$ and $\max(c_j^i + r^i)$, respectively, where $j \in \{x, y, z\}$. The atomic radii used can be selected (by the user) from the dummy atoms (Svergun *et al.*, 1995), van der Waals (Bondi, 1964; Mantina *et al.*, 2009), empirical (Slater, 1964), or calculated radii (Clementi *et al.*, 1967). Once the range is determined, it is expanded by $t_h = \max[r^{\text{probe}}, \delta]$, where r^{probe} is the **Probe Radius** and δ is the solvation thickness. The minimum and maximum locations are stored as

$$j_{\min} = \min(c_j^i - r^i) - (t_h + 2v)$$

and

$$j_{\max} = \left\lfloor \frac{\max(c_j^i + r^i) + (t_h + v)}{v} \right\rfloor v + v,$$

for each j . The space is then allocated, divided into

$$\left\lceil \left[\prod_j (j_{\max} - j_{\min}) \right] / v^3 \right\rceil$$

voxels of volume v^3 , and labeled as non-occupied (0 or blue in Fig. 3).

2. **Mark all the voxels that contain atoms.** For each atom, i , the indices of the lowest neighbor are calculated by $v_j^i = \left\lfloor (c_j^i - j_{\min}) / v \right\rfloor$, for each j . Each voxel within r^i of c_j^i is marked as an atom voxel (1 or green in Fig. 3).

3. **Mark solvent accessible layers.** Around the lowest neighbor of each atom, v_j^i , each voxel within $2 + \lfloor (r^i + r^{\text{probe}})/v \rfloor$ voxels for each j is tested. Each voxel within $r^i + r^{\text{probe}}$ of c_j^i is marked as solvation (2 or red in Fig. 3) if it was previously non-occupied (0 or blue in Fig. 3). The addition of r^{probe} to each atom allows us to simulate a spherical probe that would determine the solvent accessibility as in (Pavlov & Fedorov, 1983).
4. **Outer solvent.** If the user selects **Fill Holes** then any enclosed unoccupied voxels should be marked as excluded volume. In order to differentiate between the enclosed and outer unoccupied voxels, a 3D queue-based flood-fill algorithm (explained in (Heckbert, 1990), for example) labels the outer solvent as 3 (or light blue in Fig. 3). Any remaining 0 (or blue) voxels will be considered “holes” and marked as 1 (or green in Fig. 3). If the user did not select **Fill Holes**, then all 0 (or blue) voxels are marked as 3 (or light blue in Fig. 3).
5. **Mark solvation layer.**

Find all voxels that are marked as bulk solvent (3 or light blue) and neighbor a voxel marked as solvation (2 or red). All red voxels within $\lfloor r^{\text{probe}}/v \rfloor$ voxels of them are marked as 4 (or pink in Fig. 3).

If $r^{\text{probe}} > \delta$ then part of the pink (or 4) layer must be marked as light blue (3). As such, we find all the voxels that are marked pink (4) and neighbor light blue (3). All pink voxels within $\lfloor r^{\text{probe}} - \delta / v \rfloor$ voxels of them are marked as light blue (3).

If $r^{\text{probe}} < \delta$ then the pink (4) layer must be expanded by the difference. As such, we find all the voxels that are marked pink (4) and neighbor light blue (3). All light blue voxels within $\lfloor \delta - r^{\text{probe}} / v \rfloor$ voxels of them are marked as pink (4).

To ensure we did not accidentally change a green (or 1) to pink (or 4), we repeat

step 2.

6. Reduce voxels to irregular boxes. To shorten the computation time, when possible, voxels are binned into larger irregular rectangular boxes.

The scattering amplitude of the j th voxel (or box) of dimensions, ω_j, τ_j , and μ_j , whose center is at $\vec{r}_j^{\text{Voxel}}$ is (Székely *et al.*, 2010):

$$f_j^{\text{Voxel}}(\vec{q}) = \frac{8}{q_x \cdot q_y \cdot q_z} \sin\left(\frac{q_x \omega_j}{2}\right) \sin\left(\frac{q_y \tau_j}{2}\right) \sin\left(\frac{q_z \mu_j}{2}\right) \cdot \exp\left(i \cdot \vec{q} \cdot \vec{r}_j^{\text{Voxel}}\right) \quad (28)$$

The total scattering amplitude of the excluded voxels is then a sum over the green voxels:

$$F_{\text{Excluded Solvent}}^{\text{Voxel}}(\vec{q}) = \sum_{j \in \{\text{Green Voxels}\}} f_j^{\text{Voxel}}(\vec{q}) \quad (29)$$

and the scattering amplitude of the molecule in the solution is:

$$F_{\text{mol}}^{sv}(\vec{q}) = F_{\text{mol}}^v(\vec{q}) - \rho_0 \cdot F_{\text{Excluded Solvent}}^{\text{Voxel}}(\vec{q}). \quad (30)$$

5.9. Solvation layer

To determine the scattering amplitude of the solvation layer, we continue and expand upon the method in §5.4. The contribution of the solvation layer to the scattering amplitude is computed as the sum over the scattering amplitudes from the collection of voxels comprising that layer (pink voxels). The scattering amplitude of the solvation layer voxels is:

$$F_{\text{Solvation Layer}}(\vec{q}) = \sum_{j \in \{\text{Pink Voxels}\}} f_j^{\text{Voxel}}(\vec{q}). \quad (31)$$

The scattering amplitude of the solvated molecule is:

$$F_{\text{Solvated Molecule}}(\vec{q}) = F_{\text{mol}}^s(\vec{q}) + (\rho_{\text{Solvation Layer}} - \rho_0) \cdot F_{\text{Solvation Layer}}(\vec{q}), \quad (32)$$

where $F_{\text{mol}}^{sv}(\vec{q})$ may be used instead of $F_{\text{mol}}^s(\vec{q})$.

By default, D+ sets $r^{\text{probe}} = 0.14 \text{ nm}$, which corresponds to the radius of a water molecule (Richards, 1977). $\rho_{\text{Solvent Layer}}$, in units of $e \cdot \text{nm}^{-3}$, is set to 0. When $\rho_{\text{Solvent Layer}} = 0$, D+ ignores the contribution from the solvation layer. Other values should be assigned by the user as relevant. SOM §9.3 shows validation tests of the solvation layer of atomic models, by creating atomic models that form hollow or filled spherical structures. The scattering curves from the solvated atomic models were then compared with the scattering curves from the corresponding geometric models to which solvation layers were added. In Figs. 5, 6, and S3, and in SOM §10 there are examples in which the contribution of the solvation layer was taken into account.

5.10. Anomalous scattering

In modern synchrotrons, the energy of the X-ray photons can be varied. If scattering experiments are repeated at multiple wavelengths and the structure contains atoms whose resonant scattering terms, f' and f'' , respond to the energy scan, additional structural information can be derived. When modeling atomic models with anomalous scattering the scattering amplitude from an atom j in units of $-r_0$, in vacuum, is given by (Als-Nielsen & McMorrow, 2011):

$$f_j^{va}(q, \lambda) = f_j^0(q) + f_j'(\lambda) + if_j''(\lambda). \quad (33)$$

Therefore, one needs to provide input in the form of $f_j'(\lambda)$ and $f_j''(\lambda)$ for all the atoms whose resonant scattering terms have a significant contribution to the scattering signal in the relevant conditions. In D+, this is done via a text file loaded when loading an atomic model. D+ uses the input without any sanity checks, so it is the users responsibility to ensure that proper values are used. In particular, note that in D+, r_0 is set to 1 when $f_j^0(q)$ is computed. Input is an $\{f_j', f_j''\}$ pair with either a list of the atom indices in the PDB file or an ion type (Eq. 33 will be applied to all ions of

that type). This type of input may be obtained from X-ray fluorescence spectra and the computer program CHOOCH (Evans & Pettifer, 2001). See the User's Manual of D+ for more details and usage example.

The anomalous scattering amplitude in solution from atom j is:

$$f_j^{sa}(q, \lambda) = f_j^{va}(q, \lambda) - F_j^d(q)$$

and from a molecule:

$$F_{\text{mol}}^{as}(\vec{q}, \lambda) = \sum_{i=1}^n f_i^{sa}(q, \lambda) \cdot \exp(i\vec{q} \cdot \vec{r}_i).$$

5.11. *DebyeCalculator*

DebyeCalculator is a tool for computing the solution scattering intensity from atomic models using Debye formula (Debye, 1915):

$$I^c(q) = \sum_i^n \sum_k^n f_i^c(q) \cdot f_k^c(q) \frac{\sin(q \cdot r_{ik})}{q \cdot r_{ik}}. \quad (34)$$

Debye formula applies for spherically symmetric scatterers, which is the case for atoms. $f_i^c(q)$ and $f_k^c(q)$ are the atomic form factors in the relevant experimental conditions (vacuum, solution, or anomalous, $c \in \{v, s, va, sa\}$) of the i th and k th atoms, respectively, and r_{ik} is the distance between the atoms i and j . The advantage of the Debye approach is that orientation average is computed analytically hence the accuracy is better than any of the other methods, which numerically compute orientation average. *DebyeCalculator* is a separate tool that is provided with D+ and computes the scattering intensity using Eq. 34. *DebyeCalculator* runs on both CPU and GPU.

In both D+ and *DebyeCalculator*, the resonant scattering terms are given as a text file, which is loaded when loading an atomic model (see §5.10). Both programs use the input as is, so it is the users responsibility to ensure that proper values are used. In particular, note that r_0 is set to 1 in both programs when $f_j^0(q)$ values are computed.

The results of *DebyeCalculator* and D+ are similar when using sufficient integration iterations and grid density in D+. Comparison tests between CRY SOL (Svergun *et al.*, 1995), *DebyeCalculator*, and D+ are presented in SOM §9.2.

6. Geometric form-factors

Leaves in the hierarchical tree structure (Fig. 1) may also be geometric models. D+ computes the scattering amplitude of uniform rectangular cuboids, multiple uniform spherical shells, concentric uniform or Gaussian hollow cylinders, as well as helices with a circular cross section (Székely *et al.*, 2010). Each layer or shape can have uniform electron density contrast with respect to the solvent. In *X+* (Ben-Nun *et al.*, 2010; Ben-Nun *et al.*, 2016), which is a single-geometry software, the form-factor of infinite flat slabs or infinitely long cylinders can be computed. In contrast, D+ can add different (geometric or atomic) models to one another. As adding infinite models is nonphysical, only models with finite dimensions can be computed in D+. In SOM §9.1, the geometric models in D+ were tested and compared with the equivalent models in *X+* (Ben-Nun *et al.*, 2010; Ben-Nun *et al.*, 2016). To cross validate the accuracy of the geometric models of D+, the scattering curves from the geometric models were compared with their corresponding atomic models. The atomic models contained oxygen atoms that were randomly packed into each of the geometric shapes until they filled the shape. The adequate agreement between the models is presented in SOM §9.3.

7. Structure-factor

Hierarchical data tree structures often include assembly-symmetries, which describe the arrangement in space (shifts and rotations) of identical subunits. The contribution of assembly-symmetries to the scattering amplitude (structure-factor) may be done in the following ways.

7.1. Space-Filling symmetry

If identical subunits are only shifted with respect to one another, as in a finite primitive Bravais lattice, Space-Filling symmetries can be defined. In 3D, Space-Filling symmetries of subunit j are defined by selecting the x-y plane and using the following vectors. $\vec{A}_1^j$ of length a^j along the x direction and $\vec{A}_2^j$ of length b^j in the same plane. The angle between the two vectors is γ^j . We can then add a third vector $\vec{A}_3^j$ defined by its length c^j and two more angles α^j and β^j , where the former is between the vectors $\vec{A}_1^j$ and $\vec{A}_3^j$, and the latter is between the vectors $\vec{A}_2^j$ and $\vec{A}_3^j$. The angles should satisfy the conditions that the sum of any pair of angles is larger (or equal) than the third angle and that $\sin \gamma^j \neq 0$. D+ checks that both conditions are satisfied. If the sum of two angles equals the third angle, or if $\sin \gamma^j = 0$, the symmetry is 2D. In this case, the correct usage of the D+ is to project the 2D symmetry on the x-y plane. The projection is done by providing the values of a^j , b^j , and γ^j , set both α^j and β^j to be 90, and setting the number of repeating subunits in the third (z) direction to be 1. For 3D Space-Filling symmetry, the real-space basis vectors are given by:

$$\vec{A}_1^j = (a^j, 0, 0),$$

$$\vec{A}_2^j = (b^j \cos \gamma^j, b^j \sin \gamma^j, 0),$$

$$\vec{A}_3^j = \left(c^j \cos \alpha^j, \frac{c^j \cdot t^j}{\sin \gamma^j}, \frac{c^j \cdot B^j}{\sin \gamma^j} \right),$$

where:

$$t^j = \cos \beta^j - \cos \alpha^j \cos \gamma^j$$

and

$$B^j = \sqrt{\sin^2 \gamma^j - \sin^2 \gamma^j \cos^2 \alpha^j - (t^j)^2}.$$

D+ checks that $\sin^2 \gamma^j - \sin^2 \gamma^j \cos^2 \alpha^j - (t^j)^2 > 0$. In real-space, the unit cell vectors $\vec{A}_h$ can then be rotated by a rotation matrix $\mathbf{O}_j$, so that the final unit cell vectors are: $\vec{a}_h^j = \mathbf{O}_j \vec{A}_h^j$, where $h \in \{1, 2, 3\}$. If substructure j has a Space-Filling symmetry with N_h^j repeating subunits in the $\hat{a}_h^j$ directions, the Space-Filling scattering amplitude, F_{sf}^j is:

$$F_{\text{sf}}^j(\vec{q}) = F_j \left(\mathbf{O}_j^{-1} \vec{q} \right) \cdot \exp \left(i \vec{q} \cdot \vec{T}_j \right) \cdot SF^j(\vec{q}),$$

where F_j is the scattering amplitude of repeating subunit j that was rotated by matrix $\mathbf{O}_j$ and shifted by a vector $\vec{T}_j$ and the structure-factor, $SF^j(\vec{q})$, is given by:

$$\begin{aligned} SF^j(\vec{q}) &= \sum_{n_1^j=0}^{(N_1^j-1)} \sum_{n_2^j=0}^{(N_2^j-1)} \sum_{n_3^j=0}^{(N_3^j-1)} \exp \left(i \vec{q} \cdot \vec{R}_{n_1^j, n_2^j, n_3^j}^j \right) \\ &= \prod_{h=1}^3 \frac{1 - \exp \left(-i N_h^j \vec{q} \cdot \vec{a}_h^j \right)}{1 - \exp \left(-i \vec{q} \cdot \vec{a}_h^j \right)} \end{aligned}$$

where

$$\vec{R}_{n_1^j, n_2^j, n_3^j}^j = n_1^j \vec{a}_1^j + n_2^j \vec{a}_2^j + n_3^j \vec{a}_3^j$$

are the lattice vectors and n_h^j are integers.

7.2. Manual and Scripted symmetries

If the orientation $\mathbf{A}_k^j$ and location $\vec{R}_k^j$ of each of n identical subunits of type j are different, D+ enables manual positioning of each repeating subunit. The scattering amplitude, F_{man}^j , is then:

$$F_{\text{man}}^j(\vec{q}) = \sum_{k=1}^n F_j \left(\mathbf{A}_k^{j-1} \vec{q} \right) \cdot \exp \left(i \vec{q} \cdot \vec{R}_k^j \right)$$

The positions and orientations can be calculated from a Lua script, taken from a docking list (DOL) file, or input manually in the GUI. Each structure-factor contribution may then be recursively used in a hierarchical model (Fig. 1). Lattice sum validation tests are shown in SOM §9.1 and §9.2. Additional examples can be found in the User’s Manual, and in `./Example Files/`, `./LuaScripts`, and in the `./Tutorials/` directories, where D+ is installed.

8. Mixtures of uncorrelated populations

The scattering intensity from a mixture of multiple, uncorrelated populations is equal to the weighted sum of their intensities, where each weight is the corresponding population molar fraction (Als-Nielsen & McMorrow, 2011; Spinozzi *et al.*, 2014):

$$I(q) = \sum_{i=1}^{N_P} S_i I_i(q). \quad (35)$$

N_P is the number of populations. $S_i \geq 0$ and I_i are the normalized mass fraction and intensity of the i th population, correspondingly.

To simplify this model for optimization, we impose an implicit normalization constraint

$$\sum_{i=1}^{N_P} S_i = 1.$$

The weighted sum (Eq. 35) is then multiplied by the total mass of the measured sample. The motivation behind this modification is to:

1. Disallow physically incoherent S_i weights (for example, negative values).
2. Simplify the total mass parameter as a scale constant multiplier. This causes the derivatives of $I(q)$ in Eq. 35 (used when fitting, see §9.1) to be faster to compute.

To formulate the new model, we define $S \equiv \sum |S_i|$ and $\sigma_i \equiv |S_i| \cdot S^{-1}$. The new model is thus given by:

$$I(q) = S \sum_{i=1}^{N_P} \sigma_i I_i(q). \quad (36)$$

Eq. 36 is equivalent to Eq. 35.

9. Important features

In our implementation of the above algorithms, D+ employs several computational components. Below, we highlight some of the features that make D+ computationally efficient and versatile.

9.1. Fitting

Whereas all of the above sections deal with generating a single evaluation given a set of structural parameters, the purpose of this feature is to allow the fitting of a structural model to experimental data. To fit models to data, we incorporated Ceres-Solver (Agarwal *et al.*, 2016) to solve the least-squares problem:

$$\min_{\mathbf{x}} \quad \frac{1}{2} \sum_i \rho_i \left(\|f_i(x_{i_1}, \dots, x_{i_k})\|^2 \right)$$

$$\text{Subject to: } l_j \leq x_j \leq u_j$$

where x_i are the model mutable variables. In our case, the sum over i is just a single term (in Ceres-Solver terminology, there is only one residual block). $f_i(\cdot)$ is a cost-function that computes the residuals of the objective function (and calculates the Jacobian when asked). Before fitting, the cost-function is initialized with the experimental scattering intensity $I_{\text{experimental}}(q_i)$, an object that computes the expected scattering intensity $I_{\text{model}}(q_i)$ (as explained in the sections above), and a functor that evaluates the residuals. The residuals, f_i , can either be the normal $I_{\text{experimental}} - I_{\text{model}}$, a ratio

$$1 - \left(\frac{I_{\text{experimental}}}{I_{\text{model}}} \right)^{\pm 1},$$

where the $\pm$ is chosen so that the residual is non-negative, or a logarithmic residual,

$$\left| \log \left(\frac{I_{\text{experimental}}}{I_{\text{model}}} \right) \right|.$$

$\rho_i(\cdot)$ is a loss-function that can be chosen by the user. The complete standard set of loss-functions (trivial, Huber, soft L1, Cauchy, etc.) from Ceres-Solver, is available in the GUI of D+. Cauchy, for example, provides a robust curve fitting as it can deal with outliers in the data. The method with which Ceres-Solver tries to minimize $\rho_i(\cdot)$ can also be chosen from several methods including BFGS, LBFGS, Levenberg-Marquardt, or Dogleg. When using Levenberg-Marquardt, upper and lower bound constraints for each mutable fitting parameter are also supported by Ceres-Solver. To deal with noisy function evaluation (owing to Monte Carlo based orientation averaging), we use the Ridders' adaptive numeric differentiation method in Ceres-Solver (Ridders, 1982).

The variables that control the fitting algorithms are **Step Size** that sets a limit on the fraction by which mutable parameters can be changed, **Iterations** - the maximum number of fitting attempts, **Convergence** - the cost function cutoff standard, and **Der eps**, which regulates the value of **Step Size** based on the change in the value of the cost function. D+ can find the *local* minimum for a plausible set of initial guess model parameters. Finding the initial guess should be done by several **Generate** iterations. Alternatively, one can use the python API of D+ to generate the scattering curves and do the optimization or fitting using the available fitting algorithms of Python (see §9.3 and the examples in the manual), which may also include *global* fitting algorithms.

Fitting with certain options (for example, the thickness of a solvation layer, as explained in §5.4) can take a significant amount of time (hours). When this is the case, it is better to use educated guesses. For small solvated structures (like soluble proteins), fitting with CRY SOL or FoXS is faster than D+ as no grids are computed in those programs. Examples and validation tests of the fitting algorithms are shown in Fig. 5 and SOM §9.4.

9.2. Parallel computation and optimizations

Even using RGs to speed up the computations, computations are relatively long and can take anywhere from a second or so to several hours. We employed multiple methods to reduce the computation time.

9.2.1. Precalculation Often, there are calculations that are repeated many times that can be cached and retrieved from memory, saving computation time. For example, when calculating the scattering amplitudes of a given θ_q - ϕ_q plane, all the atomic form factors for each ion/atom are exactly the same with the contribution differing only by the phase. Additionally, if the atoms' amplitudes are sorted before the computations, the calculation can be done once per ion/atom type, improving some branch prediction or even eliminating it entirely.

When computing the amplitude of assembly symmetries, multiple copies of identically oriented substructures are identified and listed together (Eq. 5). This can significantly reduce the number of lookups to memory during the amplitude computations, speeding them up in an almost linear manner. This is particularly important when using the Direct or Hybrid methods for orientation averaging, as the number of amplitude evaluations is usually much higher than the simple amplitude computations (Eq. 13).

9.2.2. Parallelization The computations of the amplitude RGs are “embarrassingly parallel” meaning that for each point on the grid the exact same calculation is performed, differing only in $\vec{q}$. Modern graphical processing units (GPUs) are geared towards quickly computing tasks of this type. Speedups, compared to a regular CPU processor, can be between one to four orders of magnitude. The speedup depends on the choice of hardware, the algorithm, and the implementation (Ben-Nun *et al.*, 2015; Rubin

et al., 2015).

For most of the grid operations (leaf and symmetry amplitude calculation, orientation averaging) a GPU kernel was written using CUDA. The chosen layout of the grid minimizes the number of memory accesses by allowing $\vec{q}$ to be calculated from a single index (m in §3.1). Memory accesses tend to be the bottleneck of GPU kernels and can even make a GPU computation slower than a CPU. Implementations on the CPU are all parallelized using OpenMP.

One of the most commonly calculated functions is $\exp(i\vec{q} \cdot \vec{r})$. In atomic models, for example, there are many atoms situated in different points in space. By arranging $\vec{r}$ as an array of `structs`, this can be vectorized to theoretically get up to $\times 4 - \times 8$ speedups using SSE or AVX. We used the Eigen library for the vectorization of the trigonometric functions (Guennebaud *et al.*, 2010).

9.2.3. Batch integration Our RG is arranged in concentric layers. In order to evaluate the amplitude at a point between layers, we first interpolate multiple times in the four layers surrounding that point (two on either side). During the numerical orientation averaging, many points need to be evaluated. In order to minimize the number of accesses to the RG, the final interpolation (along the q direction) is used to evaluate the amplitude at all the desired q values (Fig. 2). Thus, if there is an experimental measurement at every multiple of 0.0125 nm^{-1} and the RG is spaced every 0.1 nm^{-1} then D+ gets the amplitude of eight points at once. The convergence criteria is simply that all the points are individually converged. This results approximately in a $8\times$ speedup in this example.

9.3. Interfaces

D+ comes with a full featured, user-friendly GUI for Windows based on the .NET framework. This allows interactive construction of models and helps the user gain understanding how all parameters affect the structure using visualization as well as the scattering intensity. See Fig. 4.

The actual calculation unit is separate from the GUI and can be compiled on both Windows and Linux. Communication between the GUI and the separated backend is done via a JSON and can be easily set up behind a webserver in a scalable manner. In addition to a standalone backend, there is a Python wrapper to D+ (see the User's Manual for details about the python API of D+) allowing easy integration into many other models, simulations, or advanced tools to perform more involved analyses, including, for example, expressing model parameters by rigorous physically-based link functions, accounting for polydispersity in various ways, or apply various fitting algorithms (Spinozzi *et al.*, 2014). The amplitude of the grid can be exported and used in many different ways, for example, by multiplying it externally by any lattice sum or structure factor function, and loading the resulting grid amplitude back into D+ for orientation average.

10. Accessory tools

10.1. PDBUnit

An accessory software tool for automatically finding the orientation and translations (assembly symmetry) of repeating subunits in a complex supramolecular structure is also available. The input of the tool is:

1. the complete structure represented by a PDB file;
2. the structure of a subunit, represented by a smaller PDB file, at a specific orien-

tation (that we shall call the “original” orientation);

3. the tolerance, given by the maximum root-mean-squared-displacement (RMSD) value within which repeating subunits can be considered similar.

The program reads the PDB file of the complete structure and finds all the instances of the subunit. If subunits are split at different locations in a PDB file, the parts should be combined into complete subunits before using the tool. For each instance the rotation and translation with respect to the original subunit are computed. The tool exports the assembly symmetry of each kind of subunit as a docking list (DOL) text file, which is the required input for D+ in order to compute the entire structure using the RG algorithm (Ginsburg *et al.*, 2016). More details about the tool are provided in SOM §3.

In some cases, before using D+, it is helpful to find the principal axes of an atomic structure (using the algorithm explained in SOM §4 in order to align subunits with respect to one another.

10.2. Suggest Parameters

Suggest Parameters is a tool that gets as an input, $q_{\max}$, and the x , y , and z coordinates of the point, P , which is most distant from the origin in the structure for which grids are going to be used. In the current version, $q_{\min}$ is assumed to be zero. If grids are used for the entire structure, P should be the most distant point in the entire structure. However, when the Hybrid method is used, P should be the most distant point in the part of the structure for which grids are used.

The tool computes the distance, L , of the most distant point from the origin. It can then compute, using Eq. 12, the suggested **Grid Size** for computations that use a grid.

The tool also suggests an integration method and integration parameters, like the

number of **Integration Iterations** that should be used in D+. If the diameter of the sphere that envelopes the structure is larger than L , the x , y , and z coordinates should be such that the resulting L will be that diameter. In that case, the integration parameter may not be optimal. Fitting method and fitting parameters are also suggested. In any event, the suggested parameters are only a guide or a first approximation. In practice, higher values might be more appropriate for optimal results.

11. Validation tests

In our earlier paper (Ginsburg *et al.*, 2016), several validation tests of the RG algorithm used in D+ and comparison with other programs, when relevant, were presented. Here we significantly increased the number of tests to cover a much wider range of computational options in D+. SOM §9 shows several dozen tests and cross validation tests that D+ successfully passed. Both atomic and geometric models were examined and compared with other programs (Svergun *et al.*, 1995; Ben-Nun *et al.*, 2010). Similar structures were computed in several different ways in D+ and compared with one another.

Different combinations of ways for:

1. Defining assembly symmetry: Space-Filling, Scripted, Manual, or using a DOL file
2. Computing the scattering amplitude using: Grids, the Direct, or the Hybrid methods.
3. Orientation average integration methods: Monte Carlo, Vegas Monte Carlo, or Gauss-Kronrod.
4. Processing computations though CPU or GPU.

were compared within D+ and then with Debye formula (using *DebyeCalculator*), CRY SOL, or X+ as relevant and D+ returned all the expected results (Svergun *et al.*, 1995; Ben-Nun *et al.*, 2010; Ben-Nun *et al.*, 2016). Atomic models were compared with their corresponding geometric models. In particular, the solvation layers of atomic models forming spherical or hollow spherical structures were tested and compared with the equivalent geometric models. The fitting algorithms of D+ were also tested (SOM §9.4) and gave adequate results.

12. Usage Examples

Usage examples can be found in our recent publications (Ginsburg *et al.*, 2017; Louzon *et al.*, 2017; Asor *et al.*, 2017). To further validate and demonstrate the use of D+, we measured and analyzed the scattering curve from lysozyme and microtubule (MT) in solution. Lysozyme is a soluble stable protein that has been well investigated and serves as a validation of atomic models in D+ by comparing the results of D+ with both experimental data and other software (Knight & Hub, 2015; Schneidman-Duhovny *et al.*, 2010; Svergun *et al.*, 1995).

MT are made up of $\alpha\beta$ -tubulin heterodimers that dynamically assemble into hollow nanotubes composed of straight protofilaments. Tubulin rings may also form under certain conditions (Diaz *et al.*, 1994). MT dynamics are facilitated by hydrolysis of Guanosine-5'-triphosphate (GTP). High-resolution cryo transmission electron microscopy (cryo-TEM) structures of dynamic MT were recently published (Alushin *et al.*, 2014; Hyman *et al.*, 1995). In the cryo-TEM studies, the structure of MT with 14 protofilaments was analyzed. There is, however, a distribution of MT protofilament number that vary with conditions (Raviv *et al.*, 2007; Andreu *et al.*, 1994; Pierson *et al.*, 1978; Wade *et al.*, 1990; Cueva *et al.*, 2012; Burton *et al.*, 1975; Andreu *et al.*, 1992; Choi *et al.*, 2009; Chrétien & Wade, 1991).

High-resolution synchrotron X-ray scattering from solution of dynamic MT (in the presence of 4 mM GTP and in the absence of stabilizing agents), GMPCPP-stabilized MT, and Taxol-stabilized MT were measured and analyzed (Ginsburg *et al.*, 2016; Ginsburg *et al.*, 2017). D+ was used to compute the expected scattering curves in solution, by docking the atomic model of tubulin dimer (Alushin *et al.*, 2014) onto a 3-start left handed helical lattice (Mandelkow *et al.*, 1986) where the atomic dimer model, the pitch (12.195 nm), the radius (11.940 nm), and dimer orientations were derived from PDB ID 3J6F (Alushin *et al.*, 2014), using algorithms explained in (Ginsburg *et al.*, 2016; Ginsburg *et al.*, 2017) and in SOM §2. In PDB ID 3J6F, the number of protofilaments was fixed at 14. We then extended our earlier analysis (Ginsburg *et al.*, 2016) and took into account MT structures containing 12, 13, 14, and 15 protofilaments (Ginsburg *et al.*, 2017; Chrétien & Wade, 1991). We determined the radii, the pitch, and the distribution of protofilament number that best fit the scattering data from a solution containing dynamic MT. However, the solvation layer was not taken into account. Here we extended the earlier analysis (Ginsburg *et al.*, 2016; Ginsburg *et al.*, 2017) by taking into account the solvation layer of the entire MT. Initially, the scattering amplitude of the solvated subunits was computed. We then subtracted the scattering amplitude from the various overlap regions of the solvation layers, as explained in SOM §10.2. D+ is the only software that can compute the atomic model of long solvated MT, hence in this case, the validation was done by comparing with experimental data. The approach for solvation layer computations was also tested within D+ in SOM §9.3, for the case of atomic spherical and hollow spherical models, and for the case of a tubulin ring, (SOM §10.1), which is sufficiently small so that the solvation layer of the entire structure can be computed from a PDB file of the complete tubulin ring. The usage examples discussed in this section, and additional examples can be found in `./Example Files/` and in the `./Tutorials/` directories, where D+ is

installed.

12.1. Materials and Methods

Materials Lyophilized powder of lysozyme protein from chicken egg white was purchased from Sigma-Aldrich (catalog number L6876-1G, Israel). Lysozyme was dissolved in 50 mM sodium acetate buffer that was adjusted to $\text{pH} = 4.5$ at protein concentration of $20 \text{ mg} \cdot \text{mL}^{-1}$.

Tubulin was purified from porcine brains by three polymerization/depolymerization cycles. The first cycle was at low salt (Farrell & Wilson, 1984; Ringel & Horwitz, 1987) and the other two cycles were performed in a high-molarity buffer (Castoldi & Popov, 2003).

Methods Solution X-ray scattering measurements were performed using energy of 10 keV at ID02 beamline (headed by T. Narayanan) in ESRF (Grenoble), at P12 EMBL BioSAXS beamline (D. Svergun) in PETRA III (DESY, Hamburg), and at SWING beamline (J. Perez) in Soleil synchrotron (Gif-sur-YVETTE), or in our in-house spectrometer, using energy of 8 keV. Detailed experimental descriptions of these setups were provided elsewhere (David & Pérez, 2009; Blanchet *et al.*, 2015; Van Vaerenbergh *et al.*, 2016; Nadler *et al.*, 2011; Louzon *et al.*, 2017; Fink *et al.*, 2017b). The intensity frames were normalized to the intensity of the transmitted beam, azimuthally averaged (Hammersley, 2016), and background-subtracted as explained below, in SOM §5, and in our earlier publication (Ginsburg *et al.*, 2016).

Dynamic MT was obtained and measured as explained (Ginsburg *et al.*, 2017). Briefly, 0.2 mM tubulin in BRB80 buffer (80 mM 1,4- piperazinediethanesulfonic acid, 1 mM MgCl_2 , 1 mM EGTA, adjusted to $\text{pH} = 6.9$ with KOH) with 4 mM GTP were incubated at 36°C for 30 min. An aliquot of the resulting dynamic MT solution was measured in a flow-cell, followed by centrifugation at 20,800 g, at 36°C for 30 min

and a MT pellet was obtained. The supernatant, which contained coexisting small tubulin assemblies, was measured through the same spot in the flow-cell capillary (Fig. S2A). The scattering curve of the supernatant was then subtracted from the scattered intensity of the MT sample (Fig. S2B). The scattering curves from the MT solution and the scattering curve from the supernatant are presented in Fig. S2.

Hardware architecture Computations and tests were performed on Windows computers with an Intel Core i5 3470 3.2 GHz CPU and an NVIDIA GeForce GTX Titan GPU card, or Intel Core i5 4590 3.3 GHz CPU and an NVIDIA GeForce GTX Titan Black GPU card. GeForce GTX 1070, GTX 970, and GTX 960 cards have also passed our tests. The backend was tested via a JSON on a Linux computer with an Intel Core i7-5930K 3.5 GHz CPU and a GeForce GTX 980 GPU card as well as on a Tesla P100 card.

12.2. Results and discussion

Fig. 5 shows background subtracted solution scattering curve from $20 \text{ mg} \cdot \text{mL}^{-1}$ lysozyme protein solution. The computed scattering curves were based on PDB ID 1LYZ, to which hydrogen atoms were added using PyMOL (Schrödinger, 2015), and computed by D+ and CRY SOL (Svergun *et al.*, 1995). The two programs can adequately fit the experimental data and agree with each other in most of the q -range. Fig. S3 shows a more detailed comparison between the programs, where solvent is always subtracted as dummy atoms (Eq. 26). In Fig. S3A the original PDB is used. Hydrogen atoms are then added to the PDB (Fig. S3B). Finally, a solvation layer is taken into account, using similar solvation layer thickness and electron density contrast (Fig. S3C). If H atoms are explicitly added, they are taken into account like any of the other heavier atoms. If H atoms are not added, their contribution is implicitly approximated by modifying the relevant heavy atom volume of excluded solvent to effectively account

for the added hydrogen atoms. Solvation layers are computed in each program in a different way. Therefore the results slightly differed. Fig. 5, however, shows that after small fitting adjustments both programs can adequately fit the experimental data. The difference between models with implicit and explicit H atoms is negligible. In SOM §9.4 similar tests were repeated with ubiquitin and RNase proteins, whose scattering data are available at <https://www.sasbdb.org/data/SASDAG2/>.

Solution X-ray scattering curves from dynamic MT were measured according to the protocol presented in §12.1 and Fig. S2. A more detailed explanation and verification of the protocol is provided in our earlier paper (Ginsburg *et al.*, 2017). Fig. 6 presents the background-subtracted scattering data from a solution of dynamic MT, which reveals features throughout the measured q -range ($q_{\max} = 7 \text{ nm}^{-1}$). The computed scattering curves presented in Fig. 6 are based on atomic MT models, obtained by docking the tubulin dimer structure from PDB ID 3J6F (Alushin *et al.*, 2014) into the MT 3-start left-handed helical lattice. The solvation layer of the tubulin dimer was added to each dimer, and the overlapping solvation layers were subtracted to avoid double counting, as explain in SOM §10.2. The broken curve in Fig. 6 is based on a model of MT with 14 protofilaments, where each protofilament contained 48 tubulin dimers along the long MT axis. To compute the scattering curve from this type of long structure the Hybrid method (presented in §3.2) is required. No other software can compute this size of atomic model (without the solvation layer and even more so when the solvation layer is added), hence the validation is only against experimental data.

The radius and pitch of a single MT helical lattice model that best fitted the experimental data were 11.9 nm and 12.214 nm, respectively, in agreement with PDB 3J6F (Alushin *et al.*, 2014), which contains nine dimers in a similar symmetry (see Fig. S6 and red curve in Fig. 6).

The solid blue curve in Fig. 6 is based on a fit to a linear combination of similar

atomic MT models with radii of 10.2, 11.05, 11.9, and 12.75 nm, corresponding to 12, 13, 14, and 15 protofilaments, respectively. The pitch in all the models remained 12.214 nm. The mass fraction distribution of the different models, which significantly improved the fit to the data (broken vs. solid curve in Fig. 6), was 20.5 ± 3 and 79.5 ± 3 , for MT with 13, and 14 protofilaments, respectively. This distribution is close to that of an earlier TEM data (Chrétien & Wade, 1991).

Recently, we have used D+ to analyze solution X-ray scattering data (Eisenberg *et al.*, 2017; Asor *et al.*, 2017; Ginsburg *et al.*, 2017; Chung *et al.*, 2017). In particular, we have analyzed solutions of SJW1660 flagellar filaments (Louzon *et al.*, 2017). The data were fitted to models, which took into account the atomic structure of the flagellin subunits. The analysis revealed the exact helical arrangement and the superhelical twist of the flagellin subunits within the filaments. Under osmotic stress, the filaments formed two-dimensional hexagonal bundles. Monte Carlo simulations and D+ were used to determine the fluctuations in the position of the filaments within the bundles, which were consistent with the experimental structure-factor data. The analysis revealed the elastic parameters of the filaments and was validated against osmotic stress data (Louzon *et al.*, 2017).

13. Performance Analysis

In our earlier paper (Ginsburg *et al.*, 2016), we compared the computation time of CRY SOL (ATSAS 2.8.2) with that of D+ for atomic models, based on Eq. 22, which assumes that the electron density (ED) of the surrounding solution is zero (as in vacuum). In this paper, we compared the computation time in solution (Eq. 26) and when solvation layers are taken into account (Eq. 32). We created a series of atomic models of tubulin assemblies. Tubulin dimers were gradually added along the 3-start left handed MT lattice, starting from a single dimer (PDB ID 3J6F) and up to 700

dimers. Intensity curves were computed in D+ and CRY SOL between $q_{\min} = 0$ and $q_{\max} = 5 \text{ nm}^{-1}$, using **Solvent** ED of 334 e/nm^3 , and $c_1 = 1$ (Eq. 24). Above 42 dimers, however, CRY SOL could not obtain accurate (relative error of 5% or less) results even when the number of spherical harmonics expansion terms, lm , was set to its maximum value (99). Fig. 7A shows the computation time, C_t , of the models, in aqueous solution - using Eq. 26, as a function of the number of tubulin dimers, N_{Dimers} . Fig. 7B compares the computation time of the same models, when the solvation layer of each of the structures was also taken into account. The solvation layer was computed for each structure, using Eq. 32 and the subunit overlap correction methods explained and demonstrated in SOM §10.2. D+ could accurately compute all the structures, which contained between 1 and 700 tubulin dimers (larger structure are also possible). In CRY SOL, meaningful results were obtained up 6 dimers (open square symbols). Larger structures (between 7 and 42 dimers) contained voids, hence the results of CRY SOL were inaccurate (and hence where indicated by open gray symbols).

The results show that for a single dimer in solution, the algorithm of CRY SOL is faster than that of D+, both with and without solvation layer. As the size of the tubulin assembly increases the computation of D+ is significantly shorter with and without solvation layers.

We note, however, that fitting to data of a single dimer or even few dimers, when taking into account the solvation layer, will be significantly slower in D+, as successive evaluations of the hydration layer parameters require computations of new grids, whereas in CRY SOL this is not the case. D+ should be used for larger structures.

It is of interest to note that the performance of D+ is sublinear in the number of dimers (Fig. 7C) when the number of tubulin dimers is smaller than 300. the computation time per dimer increases for larger structures because in the MT model the structure is long and hence the orientation average start to be longer on the CPU that

uses Monte Carlo. The adaptive (VEGAS) Monte Carlo used with the GPU helps to reduce the time and keep the performance sublinear in the number of dimers up to 700 dimers.

14. Conclusions

In this paper, we introduced our state-of-the-art solution X-ray scattering data analysis software D+. Nearly any complex structure can be modeled in a hierarchical manner and its solution scattering amplitude and intensity curve can be efficiently computed, using the reciprocal grid algorithm. 3D grids containing the scattering amplitudes from atomic and/or geometric model subunits are computed and used to calculate the scattering amplitudes from structures containing many subunit repeats. When the scatterers are large, a Hybrid method is used, in which grids are computed up to a predetermined point in the hierarchical data tree and used as subunits in Direct amplitude computations of the larger structures. D+ can also take into account the X-ray photon energy and thereby be used to compute anomalous X-ray scattering curves from solutions of complex structures at high-resolution. D+ has a script editor through which many different kinds of involved models can be defined and computed. The contribution of the solvation layer of complex and large structures can also be taken into account in a scalable manner. The Python wrapper of D+ can be used to compute more advanced models and integrate D+ with various computational tools, modeling, or computer simulations and thereby reveal the underlying structural biophysics that leads to the observed scattering data. D+ has been extensively tested and compared with other software or data and its self-consistency was thoroughly examined (see SOM §9). D+ and its source code are freely available for academic users.

15. Acknowledgments

We are grateful for many helpful discussions with D. Harries. We also thank I. Ringel, D. Svergun, and M. Petoukhov for helpful discussions. L. Sapir and I. Ringel are thanked for helping with some of the scattering data collection. ID02 beamline at ESRF synchrotron (T. Narayanan and his team), SWING beamline at Soleil Synchrotron (J. Perez and his team), and P12 EMBL BioSAXS Beamline in PETRA III at DESY, Hamburg (D. Svergun and his team) are acknowledged as the data presented in the paper were acquired there. We also thank I. Zandbank, D Witty, and Y. Segal (The Research Software Company). This project was supported by the Israel Science Foundation (656/16), the United States - Israel Binational Science Foundation (2016311), the Israel Ministry of Science, the Israel Ministry of Economy, and the Institute of Chemistry of the Hebrew University of Jerusalem. We thank the Safra, Wolfson, and Rudin Foundations for supporting our laboratory.

Table 1. *The coefficients for the five Gaussian approximation for implicit hydrogen atomic groups.*

Atomic Group	a_1	b_1	a_2	b_2	a_3	b_3	a_4	b_3	c
CH	0.894937	55.7145	0.894429	4.03158	3.78824	24.8323	3.14683e-6	956.628	1.42149
CH ₂	1.61908	52.1451	2.27205	24.6589	2.1815	24.6587	1.9254e-3	152.165	1.92445
CH ₃	12.5735	38.7341	-0.456658	-6.28167	5.71547	54.955	-11.711	47.898	2.87762
NH	5.06991e-3	108.256	2.03147	14.6199	1.82122	14.628	2.06506	35.4102	2.07168
NH ₂	3.00872	28.3717	0.288137	63.9637	3.39248	3.51866	2.03511	28.3675	0.269952
NH ₃	0.294613	67.4408	6.48379	29.1576	5.67182	0.54735	6.57164	0.547493	-9.02757
OH	-2.73406	22.1288	9.66263e-3	94.3428	6.64439	13.9044	2.67949	32.7607	2.39981
SH	-127.811	7.19935	62.5514	12.1591	160.747	1.88979	2.34822	55.952	-80.836

Fig. 1. Modeling a supramolecular assembly in a hierarchical manner. Each *Assembly Symmetry* may contain multiple children. Children can either be *Assembly Symmetries* or *Subunits*. A *Subunit* represents a Protein Data Bank (PDB) file or a geometric model. Internal nodes consist of *Assembly Symmetries*, whereas each leaf must be a *Subunit*. There may be an arbitrary number of hierarchy levels and nodes in each level.

Fig. 2. Illustration of the interpolations used in D+. To compute a specific $\vec{q} = (q_i, \theta_q^i, \phi_q^i)$ point (indicated in red), we first interpolate the ϕ_q values in the grid (the orange points are showing a few grid points), which are periodic (along the broken circle) and evenly spaced. We get the green points that are laying in the (q, θ_q, ϕ_q^i) plane (solid thick black curves), shown in the figure. We then use the green points to interpolate the θ_q values and obtain the blue points, which are on the $(q, \theta_q^i, \phi_q^i)$ line (solid thin line). Finally, we use the blue points to interpolate at the required q values (red points). The interpolation is done at once for all the red points that are located between adjacent blue points.

Fig. 3. Stages in determining voxelized solvent and solvation. Here, a slice of C60 is shown as an example of the stages. First the required space is determined, allocated, and marked blue (or 0). Second, all voxels within the atomic radius of each atom are marked as excluded volume (green or 1). Third, an additional **Probe Radius** is added around each atom, marking only blue voxels as red (2). The fourth step is to determine the interfaces between the solvent and the molecule. Depending on how the user chooses to treat “trapped” volume (Fill Holes or not), this is either a trivial swap of blue to light blue (No Fill Holes), or a 3D flood-fill in order to differentiate between inner and outer interfaces. In the case of Fill Holes, the blue voxels that are not changed to light blue are marked as excluded volume (green, 1). The fifth stage marks all red voxels within one **Probe Radius** of the interface between light blue (3) and red (2) as pink (4). In the sixth stage, the solvation thickness is completed from the interface between pink (4) and light blue (3). The final stage marks the remaining red voxels (2) as excluded volume (green, 1). The gray scale bar in the rightmost figure corresponds to 1 nm.

Fig. 4. A screenshot of the D+ GUI. The pane positions are all customizable. The user can visualize the structure being modeled alongside its SAXS scattering profile against data.

Fig. 5. Solution X-ray scattering data and models of Lysozyme. Azimuthally integrated background-subtracted scattering intensity as a function of the magnitude of the scattering vector, q , from solutions of $20 \text{ mg} \cdot \text{mL}^{-1}$ Lysozyme in 50 mM sodium acetate buffer, adjusted to $\text{pH } 4.5$, at ambient room temperature. Measurements were performed in SWING beamline (Soleil synchrotron). The data at $q < 1 \text{ nm}^{-1}$ were taken from measurements performed at P12 EMBL BioSAXS Beamline in PETRA III (DESY, Hamburg) measured under similar solution conditions but at lysozyme concentration of $2.5 \text{ mg} \cdot \text{mL}^{-1}$. The solid black curve represents the scattering data. The other curves were computed based on PDB ID 1LYZ. Hydrogen atoms were either implicitly taken into account or explicitly added to the PDB file by PyMol (Schrödinger, 2015). The broken scattering curves were computed by CRY SOL (ATSAS 2.8.2) (Svergun *et al.*, 1995), where the maximum order of harmonics was 50 and the order of Fibonacci grid was 18. The solid curves were computed by D+. In both D+ and CRY SOL the bulk water electron density was set to $334 \text{ e} \cdot \text{nm}^{-3}$ and the contribution of the solvation layer was taken into account. In D+, the grid size was 80, orientation average was computed by Monte Carlo integration, using Mersenne Twister algorithm. The solvent **Voxel Size** was 0.05 nm , the solvent **Probe Radius** was 0.14 nm and **Dummy Atoms** were used to account for the volume of excluded solvent (see §5.5 and Eq. 27). The fitting algorithm used the Normal Residuals cost function and the Trivial Loss function, with 20 iterations, **Step Size** of 0.01, **Convergence Tolerance** of 0.01, and **Der eps** 0.1 (see §9.1). In the implicit model, the solvation thickness was 0.2987 nm and its electron density was $337.6 \text{ e} \cdot \text{nm}^{-3}$. The excluded solvent parameter, c_1 , in Eq. 24 was 1.014, and $R^2 = 0.998$. In the explicit model, c_1 was 1.012, the solvation thickness was 0.301 nm , its electron density was $337.5 \text{ e} \cdot \text{nm}^{-3}$, and $R^2 = 0.9977$. In CRY SOL, in both models, the solvation thickness was 0.3 nm and the electron density of the solvation layer was $339 \text{ e} \cdot \text{nm}^{-3}$. In the implicit model $\chi^2 = 33.53$ and in the explicit model $\chi^2 = 35.96$. Figs. S34 and S35, in SOM §9.4 show two other examples.

Fig. 6. Solution X-ray scattering data and models of dynamic solvated microtubule (MT). Azimuthally integrated background-subtracted scattering intensity (obtained as explained in SOM §5 and Fig. S2, as a function of q , from solutions of 0.2 mM tubulin that was polymerized in the presence of 4 mM GTP. The sample was polymerized and measured at 36 °C. The black curve represents the scattering data measured at SWING beamline at Soleil synchrotron. Similar data were obtained at ID02 beamline at ESRF. The broken red curve is based on an atomic MT model with 14 protofilaments, each containing 48 tubulin dimers along the long MT axis. The tubulin dimers were arranged in a 3-start left-handed helical lattice. The radius and pitch that best fitted the data were 11.9 nm and 12.214 nm, respectively, in agreement with PDB 3J6F (Alushin *et al.*, 2014). A solvation layer was added to the entire MT structure, using a **Probe Radius** of 0.14 nm and the solvation overlap correction method explained in SOM §10.2. The shell was 0.28 nm thick and its electron density was 364 e/nm³. The solid blue curve is the computed scattering curve, which is based on a fit to a linear combination of similar solvated atomic MT models with radii of 10.2, 11.05, 11.9, and 12.75 nm, corresponding to 12, 13, 14, and 15 protofilaments. The pitch in all the models remained 12.214 nm. A Gaussian resolution function with a full width at half maximum of 0.01 nm⁻¹, was applied to all individual models forming the blue curve. The larger inset shows the low q -range on an expanded scale. The mass fraction distribution of the population that best fit the data is shown at the smaller inset. Intensity curves were computed on an NVIDIA Titan GPU, using the Hybrid method.

Fig. 7. Performance analysis of D+ and CRY SOL (ATSAS 2.8.2), using atomic models of tubulin assemblies. **A.** Computation time, C_t , of the atomic models of tubulin assemblies in solution, as a function of the number of tubulin dimers, N_{Dimers} . Solid black squares correspond to computations done by CRY SOL, where the minimum number (leading to the shortest computation time) of spherical harmonics expansion terms, lm , used for each N_{Dimers} are given in Fig. S4. The solid blue diamonds correspond to the upper limit of the computation time of D+ on a single CPU processor. D+ uses all the available CPU processors in parallel, whereas CRY SOL uses a single CPU processor. To properly compare the two programs, we multiplied the actual CPU time of D+ by the number of CPU processors, providing an upper limit on the computation time of D+, using a single CPU processor. The red solid circles, correspond to the computation time of D+ on a Titan Black GPU. For structures with more than one dimer, the Hybrid method of D+ was used with the following parameters: **Grid size**: 60, **Convergence**: 10^{-4} , **Integration Iterations**: 10^7 , **Integration Method**: Monte Carlo (Meresenne Twister) on CPU and Adaptive (VEGAS) Monte Carlo on GPU. **B.** C_t as a function of N_{Dimers} of the same atomic models when the solvation layer of each model was also taken into account. Open symbols match the solid symbols in **A** with similar shapes. The gray open squares corresponds to the computation times of CRY SOL. The results, however, started to be inaccurate, as the structures contained voids in which the solvation layers are not taken into account by CRY SOL. In D+, the computations of the solvation layers were done using the correction method explained in SOM §10, and the following parameters: **Solvent Probe Radius**: 0.14 nm, **Solvation Thickness**: 0.28 nm, **Outer Solvent ED**: $364 e/nm^3$, the **Fill Holes** option was applied, and the **Solvent Voxel Size** was 0.2 nm (which is smaller than used by other programs (Grudin *et al.*, 2017; Svergun *et al.*, 1995)). A much smaller voxel size of 0.05 nm was also tested and the results are shown in Fig. S5 **C.** The normalized computation time per dimer, given by $C_t/N_{\text{Dimers}} \cdot C_t^{\text{Dimer}}$, as a function of N_{Dimers} , where C_t^{Dimer} is the computation time of a single tubulin dimer. Solid symbols correspond to the symbols in **A** and open symbols corresponds to the symbols in **B**. The insets show parts of the graphs on expanded scales.

References

- Agarwal, S., Mierle, K. & Others, (2016). Ceres solver. <http://ceres-solver.org>.
- Als-Nielsen, J. & McMorrow, D. (2011). *Elements of modern X-ray physics*. John Wiley & Sons.
- Alushin, G. M., Lander, G. C., Kellogg, E. H., Zhang, R., Baker, D. & Nogales, E. (2014). *Cell*, **157**(5), 1117–1129.
- Andreu, J., Bordas, J., Diaz, J., de Ancos, J. G., Gil, R., Medrano, F. J., Nogales, E., Pantos, E. & Towns-Andrews, E. (1992). *Journal of molecular biology*, **226**(1), 169–184.
- Andreu, J. M., Díaz, J. F., Gil, R., De Pereda, J., de Lacoba, M. G., Peyrot, V., Briand, C., Towns-Andrews, E. & Bordas, J. (1994). *Journal of Biological Chemistry*, **269**(50), 31785–31792.
- Asor, R., Ben-nun Shaul, O., Oppenheim, A. & Raviv, U. (2017). *ACS Nano*, **11**(10), 9814–9824.
- Bardhan, J., Park, S. & Makowski, L. (2009). *Journal of applied crystallography*, **42**(5), 932–943.
- Bartels, R., Beatty, J. & Barsky, B. (1998). *An Introduction to Splines for Use in Computer Graphics and Geometric Modelling*, chap. Hermite and cubic spline interpolation. Morgan Kaufmann.
- Beaulieu, N. C. (2002). *Proceedings of the IEEE*, **90**(2), 276–279.
- Beck, R., Deek, J., Jones, J. B. & Safinya, C. R. (2010). *Nature materials*, **9**(1), 40–46.
- Ben-Nun, T., Asor, R., Ginsburg, A. & Raviv, U. (2016). *Israel Journal of Chemistry*, **56**, 622–628.
- Ben-Nun, T., Ginsburg, A., Székely, P. & Raviv, U. (2010). *Journal of Applied Crystallography*, **43**(6), 1522–1531.
- Ben-Nun, T., Levy, E., Barak, A. & Rubin, E. (2015). In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, p. 19. ACM.
- Berman, H. M., Kleywegt, G. J., Nakamura, H. & Markley, J. L. (2014). *Journal of computer-aided molecular design*, **28**(10), 1009–1014.
- Bernstein, F. C., Koetzle, T. F., Williams, G. J., Jr., E. F. M., Brice, M. D., Rodgers, J. R., Kennard, O., Shimanouchi, T. & Tasumi, M. (1977). *Journal of Molecular Biology*, **112**(3), 535 – 542.
- Blanchet, C. E., Spilotros, A., Schwemmer, F., Graewert, M. A., Kikhney, A., Jeffries, C. M., Franke, D., Mark, D., Zengerle, R., Cipriani, F. *et al.* (2015). *Journal of Applied Crystallography*, **48**(2), 431–443.
- Bondi, A. (1964). *The Journal of physical chemistry*, **68**(3), 441–451.
- Burton, P., Hinkley, R. & Pierson, G. (1975). *The Journal of cell biology*, **65**(1), 227–233.
- Castoldi, M. & Popov, A. V. (2003). *Protein. Expr. Purif.* **32**(1), 83–88.
- Chen, V. B., Arendall, III, W. B., Headd, J. J., Keedy, D. A., Immormino, R. M., Kapral, G. J., Murray, L. W., Richardson, J. S. & Richardson, D. C. (2010). *Acta Crystallographica Section D*, **66**(1), 12–21.
- Choi, M., Raviv, U., Miller, H., Gaylord, M., Kiris, E., Ventimiglia, D., Needleman, D., Kim, M., Wilson, L., Feinstein, S. *et al.* (2009). *Biophysical journal*, **97**(2), 519–527.
- Choi, M. C., Chung, P. J., Song, C., Miller, H. P., Kiris, E., Li, Y., Wilson, L., Feinstein, S. C. & Safinya, C. R. (2016). *Biochimica et Biophysica Acta (BBA)-General Subjects*.
- Chrétien, D. & Wade, R. H. (1991). *Biology of the Cell*, **71**(1-2), 161–174.
- Chung, P. J., Choi, M. C., Miller, H. P., Feinstein, H. E., Raviv, U., Li, Y., Wilson, L., Feinstein, S. C. & Safinya, C. R. (2015). *Proceedings of the National Academy of Sciences*, **112**(47), E6416–E6425.
- Chung, P. J., Song, C., Deek, J., Miller, H. P., Li, Y., Choi, M. C., Wilson, L., Feinstein, S. C. & Safinya, C. R. (2016). *Nature Communications*, **7**.
- Chung, P. J., Song, C., Miller, H. P., Li, Y., Raviv, U., Choi, M. C., Wilson, L., Feinstein, S. C. & Safinya, C. R. (2017). In *Methods in cell biology*, vol. 141, pp. 155–178. Elsevier.

- Clementi, E., Raimondi, D. & Reinhardt, W. (1967). *The Journal of chemical physics*, **47**(4), 1300–1307.
- Cueva, J. G., Hsin, J., Huang, K. C. & Goodman, M. B. (2012). *Current biology*, **22**(12), 1066–1074.
- Curtis, J. E., Raghunandan, S., Nanda, H. & Krueger, S. (2012). *Computer Physics Communications*, **183**(2), 382–389.
- David, G. & Pérez, J. (2009). *Journal of applied crystallography*, **42**(5), 892–900.
- Debye, P. (1915). *Ann. Phys.* **351**(6), 809–823.
- Deek, J., Chung, P. J., Kayser, J., Bausch, A. R. & Safinya, C. R. (2013). *Nature communications*, **4**.
- Diaz, F. J., Pantos, E., Bordas, J. & Andreu, J. M. (1994). *Journal of molecular biology*, **238**(2), 214–225.
- Dvir, T., Fink, L., Asor, R., Schilt, Y., Steinar, A. & Raviv, U. (2013). *Soft Matter*, **9**(44), 10640–10649.
- Dvir, T., Fink, L., Schilt, Y. & Raviv, U. (2014). *Langmuir*, **30**(49), 14725–14733.
- Eisenberg, I., Harris, D., Levi-Kalishman, Y., Yochelis, S., Shemesh, A., Ben-Nissan, G., Sharon, M., Raviv, U., Adir, N., Keren, N. *et al.* (2017). *Photosynthesis research*, **134**(1), 39–49.
- Evans, G. & Pettifer, R. F. (2001). *Journal of applied crystallography*, **34**(1), 82–86.
- Farrell, K. W. & Wilson, L. (1984). *Biochemistry*, **23**(16), 3741–3748.
- Fedorov, B., Ptitsyn, O. & Voronin, L. (1972). *FEBS letters*, **28**(2), 188–190.
- Fink, L., Feitelson, J., Noff, R., Dvir, T., Tamburu, C. & Raviv, U. (2017a). *Langmuir*, **33**(23), 5636–5641.
- Fink, L., Feitelson, J., Noff, R., Dvir, T., Tamburu, C. & Raviv, U. (2017b). *Langmuir*.
- Förster, F., Webb, B., Krukenberg, K. A., Tsuruta, H., Agard, D. A. & Sali, A. (2008). *Journal of molecular biology*, **382**(4), 1089–1106.
- Förster, S., Apostol, L. & Bras, W. (2010). *Journal of Applied Crystallography*, **43**(3), 639–646.
- Franke, D., Jeffries, C. M. & Svergun, D. I. (2015). *Nature methods*, **12**(5), 419–422.
- Franke, D. & Svergun, D. I. (2009). *Journal of applied crystallography*, **42**(2), 342–346.
- Fraser, R., MacRae, T. & Suzuki, E. (1978). *Journal of Applied Crystallography*, **11**(6), 693–694.
- Ginsburg, A., Ben-Nun, T., Asor, R., Shemesh, A., Ringel, I. & Raviv, U. (2016). *Journal of Chemical Information and Modeling*, **56**(8), 1518–1527.
- Ginsburg, A., Shemesh, A., Millgram, A., Dharan, R., Levi-Kalishman, Y., Ringel, I. & Raviv, U. (2017). *The Journal of Physical Chemistry B*, **121**(36), 8427–8436.
- Goldstein, H., Poole, C. P. & Safko, J. L. (2001). *Classical Mechanics*. Addison-Wesley, 3rd ed.
- Graewert, M. A., Franke, D., Jeffries, C. M., Blanchet, C. E., Ruskule, D., Kuhle, K., Flieger, A., Schäfer, B., Tartsch, B., Meijers, R. *et al.* (2015). *Scientific reports*, **5**, 10734.
- Grant, T. D. (2018). *Nature methods*, **15**(3), 191.
- Grudin, S., Garkavenko, M. & Kazennov, A. (2017). *Acta Crystallographica Section D: Structural Biology*, **73**(5).
- Guennebaud, G., Jacob, B. *et al.*, (2010). Eigen v3. <http://eigen.tuxfamily.org>.
- Gumerov, N. A., Berlin, K., Fushman, D. & Duraiswami, R. (2012). *J. Comput. Chem.* **33**(25), 1981–1996.
- Hamilton, W. (1974). *Birmingham: Kynoch Press. (Present distributor Kluwer Academic Publishers, Dordrecht. Table 2.2B.)*, pp. 273–284.
- Hammersley, A. (2016). *Journal of Applied Crystallography*, **49**(2).
- Heckbert, P. S. (1990). chap. A Seed Fill Algorithm, pp. 275–277. San Diego, CA, USA: Academic Press Professional, Inc.

- Hura, G. L., Menon, A. L., Hammel, M., Rambo, R. P., Poole, F. L., Tsutakawa, S. E., Jenney Jr, F. E., Classen, S., Frankel, K. A., Hopkins, R. C. *et al.* (2009). *Nature methods*, **6**(8), 606–612.
- Hyman, A. A., Chrétien, D., Arnal, I. & Wade, R. H. (1995). *The Journal of cell biology*, **128**(1), 117–125.
- Ilavsky, J. & Jemian, P. R. (2009). *J. Appl. Crystallogr.* **42**(2), 347–353.
- Kler, S., Asor, R., Li, C., Ginsburg, A., Harries, D., Oppenheim, A., Zlotnick, A. & Raviv, U. (2012). *Journal of the American Chemical Society*, **134**(21), 8823–8830.
- Knight, C. J. & Hub, J. S. (2015). *Nucleic acids research*, p. gkv309.
- Koltover, I., Salditt, T., Rädler, J. O. & Safinya, C. R. (1998). *Science*, **281**(5373), 78–81.
- Kornreich, M., Malka-Gibor, E., Zuker, B., Laser-Azogui, A. & Beck, R. (2016). *Physical Review Letters*, **117**(14), 148101.
- Koutsioubas, A. & Perez, J. (2013). *Journal of Applied Crystallography*, **46**(6), 1884–1888.
- Kronrod, A. (1964). *DOKLADY AKADEMII NAUK SSSR*, **154**(2), 283.
- Laurie, D. (1997). *Mathematics of Computation of the American Mathematical Society*, **66**(219), 1133–1145.
- Lotan, O., Fink, L., Shemesh, A., Tamburu, C. & Raviv, U. (2016). *The Journal of Physical Chemistry A*.
- Louzon, D., Ginsburg, A., Schwenger, W., Dvir, T., Dogic, Z. & Raviv, U. (2017). *Biophysical Journal*, **112**(10), 2184–2195.
- Mandelkow, E., Schultheiss, R., Rapp, R., Müller, M. & Mandelkow, E. (1986). *The Journal of cell biology*, **102**(3), 1067–1073.
- Mantina, M., Chamberlin, A. C., Valero, R., Cramer, C. J. & Truhlar, D. G. (2009). *The Journal of Physical Chemistry A*, **113**(19), 5806–5812.
- Marsh, R. & Slagle, K. (1983). *Acta Crystallographica Section A: Foundations of Crystallography*, **39**(1), 173–173.
- Möller, J., Chushkin, Y., Prevost, S. & Narayanan, T. (2016). *Journal of Synchrotron Radiation*, **23**(4).
- Moshe, L., Saper, G., Szekely, O., Linde, Y., Gilon, C., Harries, D. & Raviv, U. (2013). *Soft Matter*, **9**(29), 7117–7126.
- Nadler, M., Steiner, A., Dvir, T., Szekely, O., Szekely, P., Ginsburg, A., Asor, R., Resh, R., Tamburu, C., Peres, M. *et al.* (2011). *Soft Matter*, **7**(4), 1512–1523.
- Ojeda-Lopez, M. A., Needleman, D. J., Song, C., Ginsburg, A., Kohl, P. A., Li, Y., Miller, H. P., Wilson, L., Raviv, U., Choi, M. C. *et al.* (2014). *Nature materials*, **13**(2), 195–203.
- Papoulis, A. (1968). *McGraw-Hill Series in System Science, Malabar: Krieger, 1968*, **1**.
- Park, S., Bardhan, J. P., Roux, B. & Makowski, L. (2009). *J. Chem. Phys.* **130**(13), 134114.
- Pavlov, M. Y. & Fedorov, B. A. (1983). *Biopolymers*, **22**(6), 1507–1522.
- Pedersen, M. C., Arleth, L. & Mortensen, K. (2013). *Journal of Applied Crystallography*, **46**(6), 1894–1898.
- Pérez, J. & Koutsioubas, A. (2015). *Acta Crystallographica Section D: Biological Crystallography*, **71**(1), 86–93.
- Peter Lepage, G. (1978). *Journal of Computational Physics*, **27**(2), 192–203.
- Petoukhov, M. V., Franke, D., Shkumatov, A. V., Tria, G., Kikhney, A. G., Gajda, M., Gorba, C., Mertens, H. D., Konarev, P. V. & Svergun, D. I. (2012). *Journal of applied crystallography*, **45**(2), 342–350.
- Petoukhov, M. V. & Svergun, D. I. (2005). *Biophysical journal*, **89**(2), 1237–1250.
- Pierson, G. B., Burton, P. R. & Himes, R. H. (1978). *The Journal of cell biology*, **76**(1), 223–228.
- Poitevin, F., Orland, H., Doniach, S., Koehl, P. & Delarue, M. (2011). *Nucleic acids research*, **39**(suppl 2), W184–W189.

- Rädler, J. O., Koltover, I., Salditt, T. & Safinya, C. R. (1997). *Science*, **275**(5301), 810–814.
- Ravikumar, K. M., Huang, W. & Yang, S. (2013). *J. Chem. Phys.* **138**(2), 024112.
- Raviv, U., Nguyen, T., Ghafouri, R., Needleman, D. J., Li, Y., Miller, H. P., Wilson, L., Bruinsma, R. F. & Safinya, C. R. (2007). *Biophysical journal*, **92**(1), 278–287.
- Richards, F. M. (1977). *Annual review of biophysics and bioengineering*, **6**(1), 151–176.
- Ridders, C. (1982). *Advances in Engineering Software (1978)*, **4**(2), 75–76.
- Ringel, I. & Horwitz, S. B. (1987). *J. Pharmacol. Exp. Ther.* **242**(2), 692–698.
- Rubin, E., Levy, E., Barak, A. & Ben-Nun, T. (2015). *ACM Transactions on Architecture and Code Optimization (TACO)*, **11**(4), 44.
- Saper, G., Kler, S., Asor, R., Oppenheim, A., Raviv, U. & Harries, D. (2012). *Nucleic acids research*, p. gks1270.
- Sarje, A., Li, X. S. & Hexemer, A. (2014). In *Parallel Processing (ICPP), 2014 43rd International Conference on*, pp. 201–210. IEEE.
- Schilt, Y., Berman, T., Wei, X., Barenholz, Y. & Raviv, U. (2016). *Biochimica et Biophysica Acta (BBA)-General Subjects*, **1860**(1), 108–119.
- Schneidman-Duhovny, D., Hammel, M. & Sali, A. (2010). *Nucleic acids research*, **38**(suppl 2), W540–W544.
- Schneidman-Duhovny, D., Hammel, M., Tainer, J. A. & Sali, A. (2013). *Biophysical journal*, **105**(4), 962–974.
- Schneidman-Duhovny, D., Hammel, M., Tainer, J. A. & Sali, A. (2016). *Nucleic acids research*, p. gkw389.
- Schrödinger, L. (2015). *The PyMOL Molecular Graphics System, Version 1.8*.
- Shaharabani, R., Ram-On, M., Avinery, R., Aharoni, R., Arnon, R., Talmon, Y. & Beck, R. (2016). *Journal of the American Chemical Society*, **138**(37), 12159–12165.
- Shannon, C. E. (1949). *Proceedings of the IRE*, **37**(1), 10–21.
- Slater, J. C. (1964). *J. Chem. Phys.* **41**(10), 3199–3204.
- Sonneveld, P. (1969). *Journal of Engineering Mathematics*, **3**(2), 107–117.
- Spinozzi, F., Ferrero, C., Ortore, M. G., De Maria Antolinos, A. & Mariani, P. (2014). *Journal of applied crystallography*, **47**(3), 1132–1139.
- Steiner, A., Szekely, P., Szekely, O., Dvir, T., Asor, R., Yuval-Naeh, N., Keren, N., Kesselman, E., Danino, D., Resh, R. *et al.* (2012). *Langmuir*, **28**(5), 2604–2613.
- Svergun, D., Barberato, C. & Koch, M. (1995). *Journal of Applied Crystallography*, **28**(6), 768–773.
- Szekely, O., Schilt, Y., Steiner, A. & Raviv, U. (2011*a*). *Langmuir*, **27**(24), 14767–14775.
- Szekely, O., Steiner, A., Szekely, P., Amit, E., Asor, R., Tamburu, C. & Raviv, U. (2011*b*). *Langmuir*, **27**(12), 7419–7438.
- Székely, P., Ginsburg, A., Ben-Nun, T. & Raviv, U. (2010). *Langmuir*, **26**(16), 13110–13129.
- Van Vaerenbergh, P., Léonardon, J., Sztucki, M., Boesecke, P., Gorini, J., Claustre, L., Sever, F., Morse, J. & Narayanan, T. (2016). In *Proceedings of the 12th International Conference on Synchrotron Radiation Instrumentation (SRI2015)*, vol. 1741, p. 030034. AIP Publishing.
- Virtanen, J. J., Makowski, L., Sosnick, T. R. & Freed, K. F. (2011). *Biophysical journal*, **101**(8), 2061–2069.
- Wade, R. H., Chrétien, D. & Job, D. (1990). *Journal of molecular biology*, **212**(4), 775–786.
- Warren, H. S. (2012). *Hacker's Delight*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.
- Watson, M. C. & Curtis, J. E. (2013). *J. Appl. Crystallogr.* **46**(4), 1171–1177.
- Weisstein, E. W., (2015). Sphere point picking from mathworld—a wolfram web resource. <http://mathworld.wolfram.com/SpherePointPicking.html>. Accessed: 2014-7-22.

- Wong, G. C., Tang, J. X., Lin, A., Li, Y., Janmey, P. A. & Safinya, C. R. (2000). *Science*, **288**(5473), 2035–2039.
- Wright, D. W. & Perkins, S. J. (2015). *Journal of applied crystallography*, **48**(3), 953–961.

Synopsis

D+ is a computer program that computes both monochromatic and anomalous X-ray scattering curves from solutions of complex structures, using a hierarchical reciprocal grid algorithm.

Supporting Online Material

***D+*: Software for High-Resolution Hierarchical Modeling of Solution X-Ray Scattering from Complex Structures**

AVI GINSBURG,^{a,b} TAL BEN-NUN,^{a,b,c} ROI ASOR,^{a,b} ASAF SHEMESH,^{a,b}

LEA FINK,^{a,b} ROEE TEKOAH,^{a,b} YEHONATAN LEVARTOVSKY,^{a,b}

DANIEL KHAYKELSON^{a,b} AND URI RAVIV ^{a,b*}

^a*Institute of Chemistry, The Hebrew University of Jerusalem, Edmond J. Safra
Campus, Givat Ram, 9190401, ^bCenter for Nanoscience and Nanotechnology, The
Hebrew University of Jerusalem, Edmond J. Safra Campus, Givat Ram,
9190401, Jerusalem, Israel, and ^cSchool of Computer Science and Engineering, The
Hebrew University of Jerusalem, Edmond J. Safra Campus, Givat Ram, 9190401
Jerusalem, Israel. E-mail: uri.raviv@mail.huji.ac.il*

(Received 0 XXXXXXXX 0000; accepted 0 XXXXXXXX 0000)

1. Uniformly spaced 3D Cartesian grid

The naïve approach to span the relevant reciprocal-space would be to use a cube with a side of $2q_{\max}$ from $-q_{\max}$ to $q_{\max}$. However, because only values within a radius of $q_{\max}$ are needed, the corners need not be allocated and calculated. The ratio between the volumes of the sphere and its enclosing cube is:

$$\frac{\frac{4\pi}{3}q_{\max}^3}{(2q_{\max})^3} = \frac{\pi}{6} \approx \frac{1}{2}$$

Thus, only the points that satisfy

$$|\vec{q}| \leq q_{\max} + 3\Delta q$$

where Δq is the spacing of the grid, should be used. The three extra layers ensure that all points along the shell of radius $q_{\max}$ have calculated points around them for grid interpolation.

2. Finding the rotation angles of an object

To save computations, we bin copies of an object together when their orientation is identical. To find the final rotation angles of an object that has been rotated several times, relative to its initial lab axes directions: $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$, we apply the same rotation on the lab axes unit vector. In other words, we multiply the rotation matrices and obtain the final rotation matrix:

$$\mathbf{XYZ} = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} \quad (\text{S1})$$

To find the α , β , and γ rotation angles (about the x , y , and z -axes, respectively), in the range between $-\pi$ and π , we compare the $\mathbf{XYZ}$ matrix with the rotation matrix:

$$\mathbf{A}(\alpha, \beta, \gamma) = \begin{bmatrix} \cos \beta \cos \gamma & -\cos \beta \sin \gamma & \sin \beta \\ \cos \alpha \sin \gamma + \cos \gamma \sin \alpha \sin \beta & \cos \alpha \cos \gamma - \sin \alpha \sin \beta \sin \gamma & -\cos \beta \sin \alpha \\ \sin \alpha \sin \gamma - \cos \alpha \cos \gamma \sin \beta & \cos \gamma \sin \alpha + \cos \alpha \sin \beta \sin \gamma & \cos \alpha \cos \beta \end{bmatrix}$$

and find the three angles.

To calculate the angles we use the function $\arctan(y, x)$ (the function `ATAN2` in C++), which gets the values of the $\cos$ (for y) and the $\sin$ (for x) of the relevant angle and returns the angle (in radians) in its correct quadrant, based on the sign of the two arguments. If both the $\cos$ and the $\sin$ are positive the angle is between 0 and $\pi/2$. If the $\sin$ is positive and the $\cos$ is negative the angle is between $\pi/2$ and π . If both are negative the angle is between $-\pi$ and $-\pi/2$, and if the $\sin$ is negative and the $\cos$ is positive then the angle is between $-\pi/2$ and 0.

We then proceed as follows (Heckbert, 1994).

$$\alpha_0 = \arctan(z_2, z_3)$$

We define $c_2 \equiv \sqrt{x_1^2 + y_1^2}$.

If $\alpha_0 > 0$:

$$\alpha_0 = \alpha_0 - \pi$$

$$\beta_0 = \arctan(-z_1, -c_2)$$

else

$$\beta_0 = \arctan(-z_1, c_2)$$

We define:

$$s_1 \equiv \sin(\alpha_0)$$

$$ca_1 \equiv \cos(\alpha_0)$$

and then:

$$\gamma_0 = \arctan(s_1 \cdot x_3 - ca_1 \cdot x_2, ca_1 \cdot y_2 - s_1 \cdot y_3)$$

Finally we get that:

$$\alpha = -\alpha_0, \beta = -\beta_0, \gamma = -\gamma_0.$$

3. PDBUnits: Accessory tool for identifying the rotation and translation of repeating subunits in a PDB file

Complex and large structures often contain repeating subunits that differ from one another in orientation and location. When structures can be represented by hierarchical data tree structures, *D+* is using the Reciprocal Grid (RG) algorithm (Ginsburg *et al.*, 2016). In the data tree structures subunits are leaves and the manner by which the subunits are organized is represented by assembly symmetry nodes (Fig. 1). In the RG method the scattering amplitude of the assembly is calculated by computing the

amplitudes of the basic subunits (leaves) on 3D reciprocal-space grids, moving up in the hierarchy, calculating the grids of larger structures containing repeats of the subunits. By repeating this process for all the leaves and nodes of the tree the scattering amplitude of the final complex structure is obtained. In this way, there is no need to recalculate the scattering of each instance of a repeating subunit. Therefore, to reduce the computation time when dealing with large supramolecular structures, it is best to identify repeating subunits. *D+* allows users to import **Manual Symmetries**, which describe the manner by which repeating subunits are rotated and shifted in space. To efficiently compute the scattering amplitude from a Protein Data Bank (PDB) file of a complex structure made of many repeating subunits using the RG algorithm, *D+* requires a list of all the orientations and translations (or symmetries) of all the instances of each repeating subunit. The PDBUnits tool, which will be described in the following, can automatically export the symmetry of each type of subunit. The input is the complete structure (represented by a PDB file), the structure of a subunit (represented by a smaller PDB file) at a specific orientation (that we shall call the “original” orientation), and the tolerance, given by the maximum root-mean-squared-displacement (RMSD) value within which repeating subunits can be considered similar. The program reads the complete structure and finds all the instances of the subunit. For each instance, the rotation and translation with respect to the original subunit is computed. To ensure consistency, the rotation and translations are applied to the original subunit and the RMSD between the original subunit and the examined instance of the subunit is computed. A comparison is made between the examined instance and a duplicated subunit, obtained by applying the translation and rotation that best fitted the examined subunit, to its original orientation. This process can be repeated for every additional similar subunit that the complete structure contains. The results are exported to *D+* input format for immediate use.

3.1. Implementation

The program gets the PDB file of the complete structure and the PDB of the inspected subunit. It reads the files and converts them into arrays of atomic symbols and their real-space location coordinates. The array of the complete structure is called $\mathbf{A}$ and the array of the subunit is called $\mathbf{S}$. $\mathbf{A}_i$ corresponds to the i th atom in array $\mathbf{A}$. The first condition for identifying a group of atoms in $\mathbf{A}$ as an instance of $\mathbf{S}$ is having an identical atomic sequence to $\mathbf{S}$. The program runs through array $\mathbf{A}$ and identifies all the $\mathbf{S}$ atom sequences. Each sequence of atoms in $\mathbf{A}$ with atomic sequence $\mathbf{S}$ is called $\mathbf{P}$ (as this set is a potential instance of $\mathbf{S}$). Being identical to the atomic sequence of $\mathbf{S}$ is a necessary but not a sufficient condition for $\mathbf{P}$ to be identified as an instance of the subunit $\mathbf{S}$.

To further explore the character of $\mathbf{P}$ we need to find its translation and rotation in relation to $\mathbf{S}$. To find the rotation of $\mathbf{P}$ with respect to $\mathbf{S}$, we need to translate both to the origin:

$$\vec{P}_i = B\vec{P}_i - \vec{P}_{\text{cm}}$$

$$\vec{S}_i = B\vec{S}_i - \vec{S}_{\text{cm}}$$

This operation is applied to each atom in $\mathbf{P}$ and $\mathbf{S}$ where $\vec{P}_i$ and $\vec{S}_i$ are the i th atom vectors after translation and, $B\vec{P}_i$ and $B\vec{S}_i$ are the i th atom vectors before translation ($i \in \{1, \dots, N_p\}$, where N_p is the number of atoms in each array) and $\vec{P}_{\text{cm}}$ and $\vec{S}_{\text{cm}}$ are the centers of mass of each structure respectively.

The tool then creates two reference matrices from the first four atoms of $\mathbf{P}$ and $\mathbf{S}$:

$$\mathbf{M}_P = \begin{bmatrix} \vec{P}_1 - \vec{P}_0 \\ \vec{P}_2 - \vec{P}_0 \\ \vec{P}_3 - \vec{P}_0 \end{bmatrix},$$

$$\mathbf{M}_S = \begin{bmatrix} \vec{S}_1 - \vec{S}_0 \\ \vec{S}_2 - \vec{S}_0 \\ \vec{S}_3 - \vec{S}_0 \end{bmatrix}.$$

$\mathbf{M}_P$ is the reference matrix for $\mathbf{P}$ and $\mathbf{M}_S$ is the reference matrix for $\mathbf{S}$. The aim is to find the rotation matrix, $\mathbf{R}$, that can rotate $\mathbf{M}_S$ to $\mathbf{M}_P$:

$$\mathbf{M}_P = \mathbf{R} \cdot \mathbf{M}_S.$$

By multiplying both sides by $\mathbf{M}_S^{-1}$ we get:

$$\mathbf{R} = \mathbf{M}_P \cdot \mathbf{M}_S^{-1}.$$

The program computes $\mathbf{R}$. If $\mathbf{P}$ is an instance of the $\mathbf{S}$, then each atom in $\mathbf{P}$ should be obtained by rotating the corresponding atom in $\mathbf{S}$ by $\mathbf{R}$. If any atom in $\mathbf{P}$ has a different rotation with respect to $\mathbf{S}$, then $\mathbf{P}$ is not an instance of $\mathbf{S}$. In that case, $\mathbf{P}$ and $\mathbf{S}$ have identical atomic sequence but different conformations. The tool rotates a copy of $\mathbf{S}$ by $\mathbf{R}$:

$$\mathbf{RS} = \mathbf{R} \cdot \mathbf{S}.$$

If $\mathbf{RS}$ is overlapping with $\mathbf{P}$, then $\mathbf{R}$ describes the rotation of $\mathbf{S}$ to $\mathbf{P}$. Because PDBs are based on experimental data, and because of numerical artifacts, $\mathbf{RS}$ does not always accurately overlap with $\mathbf{P}$, even if $\mathbf{P}$ is indeed an instance of $\mathbf{S}$. To take this into account, the program calculates the $RMSD_{\text{rotation}}$ between $\mathbf{RS}$ and $\mathbf{P}$:

$$RMSD_{\text{rotation}} = \frac{1}{N_p} \sum_{i=1}^{N_p} \sqrt{(\vec{RS}_i - \vec{P}_i)^2}$$

If the $RMSD_{\text{rotation}}$ exceeds a given tolerance value, ϵ , the tool does not count $\mathbf{P}$ as an instance of $\mathbf{S}$ and continues to search for other instances. If $RMSD_{\text{rotation}} \leq \epsilon$, then the tool continues to check the translation of $\mathbf{P}$ with respect to $\mathbf{S}$.

In the complete structure (**A**), an instance of the subunit (**S**) can be translated. In order to find that translation-vector, **S** is rotated by the matrix **R** that was found above, $\mathbf{RS} = \mathbf{R} \cdot \mathbf{S}$. By rotating **S** to the orientation of **P**, the tool eliminates the effect of the rotation on the coordinates and examines the translation alone:

$$\vec{T} = \vec{P}_{\text{cm}} - \vec{R}\vec{S}_{\text{cm}}$$

$\vec{T}$ is the translation vector, $\vec{P}_{\text{cm}}$ is the center of mass of **P** and $\vec{R}\vec{S}_{\text{cm}}$ is the center of mass of **RS**. If **P** is an instance of **S**, then $\vec{T}$ describes the translation relations of all of the atoms of **P** with respect to **RS**. The tool then translates all the atoms of **RS** by the vector $\vec{T}$:

$$\mathbf{TS} = \mathbf{RS} + \vec{T}$$

If **P** is an instance of **S**, **TS** should overlap with **P** to within an $RMSD_{\text{translation}}$ smaller than ϵ , where:

$$RMSD_{\text{translation}} = \frac{1}{N_p} \sum_{i=1}^{N_p} \sqrt{(\vec{T}\vec{S}_i - \vec{P}_i)^2}$$

If $RMSD_{\text{translation}} \leq \epsilon$, **P** is considered an instance of **S**, otherwise the tool does not count **P** as an instance of **S** and continues to search for other instances. The process that was described above is repeated for every group of atoms that their atomic sequence is identical to **S**. In the end, the tool recognizes all of the instances of subunit **S** in **A**.

3.1.1. Format and units Docking list (DOL) text file is the input format for manual symmetries in *D+*, and is therefore, the output format of the tool. The tool writes every instance of subunit, **S**, to a DOL file that can be later used by *D+*. To match the format of the DOL file, the translation vector which is in PDB units (Å) is divided by 10, to match the units of *D+* (nanometers). The tool extracts the angles from the rotation matrix by Tait-Bryan rotation angle convention used in *D+* (in which the

rotation matrix is ordered as **XYZ**). The Tait-Bryan rotation angles are in radians. *D+* uses angles in degrees in the DOL file. The Tait-Bryan rotation angles are therefore converted into degrees before writing them into the DOL file.

3.2. Test examples

A small CCl_4 subunit in tetrahedron symmetry was created manually as a test case. The subunit was placed in a PDB twice, in different orientations and translations. It was placed along with random atoms and some atomic sequences that are similar but not identical to the subunit. Two identical atomic sequences were also placed (C and 4 Cl) but as separate atoms and not in a structure (to verify that the tool successfully discards them). The tool loaded the PDB of CCl_4 and of the complete structure. The tool successfully recognized the two instances of CCl_4 and discarded any other similar but wrong cases. A DOL file was exported. *D+* loaded the DOL file and the PDB of the subunit, and a scattering curve was computed. A PDB containing the two CCl_4 orientations only was created by removing the random atoms and misleading structures and its scattering curve was computed by *D+*, using a **Grid Size** of 100 (Fig. S1A).

A second example, is the PDB 3J6F (Alushin *et al.*, 2014), containing 9 tubulin dimers from the microtubule (MT) structure. A PDB representation of a single tubulin dimer subunit was extracted from 3J6F. The two PDB files were loaded to the tool. The tool successfully recognized all the 9 tubulin dimers and exported the translations and rotations into a DOL file. The DOL file was loaded to *D+* along with the tubulin dimer subunit PDB and the scattering signal was computed. The scattering curve of 3J6F PDB was also computed by *D+*, using a **Grid Size** of 200 (Fig. S1B).

The last example is of tubulin ring comprising 13 tubulin dimers. The PDB of the full ring structure and the PDB of a tubulin dimer were loaded to our tool. The tool successfully identified the 13 repeating dimer subunits and exported a correct DOL

file. *D+* computed the scattering curves from the PDB of the full ring or from the DOL file and the PDB of the dimer, using a **Grid Size** of 300 (Fig. S1C).

Fig. S1. Solution scattering intensity as a function of the magnitude of the scattering vector, q . The scattering curves were computed by *D+* using the subunit PDB file and the DOL file obtained from our PDBUnits tool, using ϵ of $0.01 \text{ \AA}$ (red curves), or the PDB file of entire structure (black curves). **A.** CCl_4 **B.** Nine tubulin dimers from the MT structure. **C.** Tubulin ring comprising 13 dimers. In all cases, *D+* used a **Convergence** tolerance of 0.001.

Fig. S1 shows that scattering curves that were computed with the DOL that was

created by the tool, completely overlapped with the curves that were computed from the PDB of the entire assembly. In addition, the PDB files that were exported from $D+$ using the DOL files and the PDB of the subunit were identical to the original full PDB. The computation time was about ten times shorter when DOL files were used owing to the advantages of the RG algorithm (Ginsburg *et al.*, 2016). Therefore, whenever there are repeating subunits, especially in large assemblies, it is recommended to analyze the structure with the tool and only then by $D+$.

4. Finding the principal axes of an atomic structure

Given a $3 \times n$ matrix containing the coordinates of n atoms in a PDB file (whose origin is at the center of mass):

$$\mathbf{A} = \begin{bmatrix} x_1 & y_1 & z_1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & z_n \end{bmatrix},$$

the principal axes can be found (as in principle component analysis (Jolliffe, 2002)), in the following manner. Matrix $\mathbf{M}$ is defined by:

$$\mathbf{M} = \mathbf{A}^T \mathbf{A}.$$

$\mathbf{E}$ is the eigenvector $(\vec{E}_i)$ matrix of $\mathbf{M}$, sorted in descending order of the eigenvalues (e_i). If the object is more prolate, i.e. $(e_1 - e_2) < (e_2 - e_3)$ then we align e_3 with the z -axis. The rotation axis is given by $\hat{\mathbf{r}}\mathbf{a} = \frac{\hat{z} \times \vec{E}_3}{\|\vec{E}_3\|}$ and the angle is:

$$\theta = \arccos \left(-\frac{\hat{z} \cdot \vec{E}_3}{\|\vec{E}_3\|} \right).$$

The rotation matrix is:

$$\mathbf{R} = \begin{bmatrix} ra_x^2(1 - \cos \theta) + \cos \theta & ra_x ra_y(1 - \cos \theta) - ra_z \sin \theta & ra_x ra_z(1 - \cos \theta) + ra_y \sin \theta \\ ra_x ra_y(1 - \cos \theta) + ra_z \sin \theta & ra_y^2(1 - \cos \theta) + \cos \theta & ra_y ra_z(1 - \cos \theta) - ra_x \sin \theta \\ ra_x ra_z(1 - \cos \theta) - ra_y \sin \theta & ra_y ra_z(1 - \cos \theta) + ra_x \sin \theta & ra_z^2(1 - \cos \theta) + \cos \theta \end{bmatrix}$$

where ra_x, ra_y , and ra_z are the elements of $\hat{\mathbf{r}}\mathbf{a}$. If the object is more oblate i.e. $(e_1 - e_2) > (e_2 - e_3)$, $\vec{E}_1$ is aligned to the z -axis using $\hat{z} \times \vec{E}_1$. Aligning the second axis is done by setting $\mathbf{A}' = \mathbf{R} \cdot \mathbf{A}$ and repeating the process using $\hat{y} \times \vec{E}_2$.

5. Supernatant subtraction from microtubule sample

Dynamic microtubule (MT) was assembled in the presence of 4 mM GTP and no stabilizing agent was added. The MT sample contained MTs as well as other smaller tubulin aggregates. By centrifuging the sample (20,800 g, at 36 °C for 30 min), the MT were removed to a pellet and the remaining supernatant was measured and used as background for the MT sample. The resulting subtracted curve has identifiable features throughout the entire scattering range (Fig. S2).

Fig. S2. Azimuthally integrated scattered intensity curves as a function of the magnitude of the scattering vector, q , from the dynamic microtubule (MT) sample in Fig. 6, its supernatant, and the subtracted curve. **A.** The blue curve is the scattering from the MT sample, containing MT, tubulin dimer, and small tubulin assemblies. The red curve is the scattering from the supernatant after centrifuging the sample. **B.** The black curve is the resulting subtracted signal (sample minus supernatant). The insets show the high q -range data on an expanded scale.

6. Comparing D+ with CRY SOL

Fig. S3A compares the computed scattering curves from the atomic model of lysozyme in water, based on PDB ID 1LYZ, where hydrogen atoms were implicitly taken into account. The curves were computed by *D+*, *DebyeCalculator*, and CRY SOL. In Fig. S3B, a similar comparison was done after hydrogen atoms were explicitly added to the PDB file (using PyMol software (Schrödinger, 2015)). Fig. S3C, compares the computation results of the two programs when a solvation layer was added. We attribute the differences in the results to the slightly different algorithms used by the two programs. In Fig. S3A and B, the computations were also compared with *DebyeCalculator*, which applies Debye formula, and is in good agreement with *D+*.

Fig. S3. Computed scattered intensity curves as a function of q , from the atomic models of lysozyme (PDB ID 1LYZ). The dashed black curve was computed by CRY SOL (ATSAS 2.8.2) (Svergun *et al.*, 1995), where the maximum order of harmonics was 50 and the order of Fibonacci grid was 18. The solid blue curves were computed by *D+*. In both programs, the solvent density of water was set to $334 \text{ e} \cdot \text{nm}^{-3}$. **A.** The PDB structure in water using implicit hydrogen atoms. **B.** Adding hydrogen atoms to the PDB using PyMOL (Schrödinger, 2015) and computing the scattering in water with explicit hydrogen atoms. **C.** The solvation layer was taken into account, using the default parameters of both programs: solvent Probe Radius of 0.14 nm, solvation layer thickness of 0.3 nm with an electron density contrast of $30 \text{ e} \cdot \text{nm}^{-3}$. Intensity curves computed by *DebyeCalculator* gave similar results to those of *D+* in **A** and **B**. Solvation shells cannot be added in *DebyeCalculator*.

7. Performance Analysis

Figure S4 shows the number of spherical harmonics expansion terms, lm , in CRY SOL (ATSAS 2.8.2) (Svergun *et al.*, 1995) as a function of the number of tubulin dimers, N , in the models. The reported lm is the minimum value need to obtain residuals (Eq. S2) of 0.05 or less at each q point between CRY SOL (ATSAS 2.8.2) and D+ or *DebyeCalculator*, which uses Debye formula (Debye, 1915). For $N \geq 43$ the maximum lm value of CRY SOL (99), was insufficient to meet accuracy criterion.

Figure S5 repeats the performance analysis of Fig. 7, using a much smaller Solvent Voxel Size of 0.05 nm (instead of 0.2 nm, which is already smaller than used by other programs (Grudin *et al.*, 2017; Svergun *et al.*, 1995)), to compute the contribution of the solvation layer. This computation is slower but can be more accurate in some cases.

Fig. S4. The minimum number of spherical harmonics expansion terms, lm , in CRY SOL (ATSAS 2.8.2) (Svergun *et al.*, 1995), which were required to attain accurate results in Fig. 7, as a function of the number of tubulin dimers, N_{Dimers} . The level of accuracy was set to have a residual value (Eq. S2) of 0.05 or less, compared with either *DebyeCalculator* or D+.

Fig. S5. Performance analysis of D+ and CRY SOL (ATSAS 2.8.2), using atomic models of tubulin assemblies. **A.** Computation time, C_t , of the atomic models of tubulin assemblies in solution, as a function of the number of tubulin dimers, N_{Dimers} . Solid black squares correspond to computations done by CRY SOL, where the minimum number (leading to the shortest computation time) of spherical harmonics expansion terms, lm , used for each N_{Dimers} are given in Fig. S4. The solid blue diamonds correspond to the upper limit of the computation time of D+ on a single CPU processor. D+ uses all the available CPU processors in parallel, whereas CRY SOL uses a single CPU processor. To properly compare the two programs, we multiplied the actual CPU time of D+ by the number of CPU processors, providing an upper limit on the computation time of D+, using a single CPU processor. The red solid circles, correspond to the computation time of D+ on a Titan Black GPU. For structures with more than one dimer, the Hybrid method of D+ was used with the following parameters: **Grid size**: 60, **Convergence**: 10^{-4} , **Integration Iterations**: 10^7 , **Integration Method**: Monte Carlo (Meresenne Twister) on CPU and Adaptive (VEGAS) Monte Carlo on GPU. **B.** C_t as a function of N_{Dimers} of the same atomic models when the solvation layer of each model was also taken into account. Open symbols match the solid symbols in **A** with similar shapes. The gray open squares corresponds to the computation times of CRY SOL. The results, however, started to be inaccurate, as the structures contained voids in which the solvation layers are not taken into account by CRY SOL. In D+, the computations of the solvation layers were done using the correction method explained in SOM §10, and the following parameters: **Solvent Probe Radius**: 0.14 nm, **Solvation Thickness**: 0.28 nm, **Outer Solvent ED**: $364 e/nm^3$, the **Fill Holes** option was applied, and the **Solvent Voxel Size** was 0.05 nm. A larger voxel size of 0.2 nm was also computed and the results are shown in Fig. 7. **C.** The normalized computation time per dimer, given by $C_t/N_{\text{Dimers}} \cdot C_t^{\text{Dimer}}$, as a function of N_{Dimers} , where C_t^{Dimer} is the computation time of a single tubulin dimer. Solid symbols correspond to the symbols in **A** and open symbols corresponds to the symbols in **B**. The insets show parts of the graphs on expanded scales.

8. Comparing experimental microtubule solution scattering data to microtubule models with varying protofilament number

Fig. S6 compares the experimental dynamic MT scattering data with different atomic MT models whose protofilament number vary between 12 and 15.

Fig. S6. Solution X-ray scattering data from dynamic microtubule (MT) and MT models with varying number of protofilaments. Azimuthally integrated, background-subtracted, experimental scattering intensity from dynamic MT (black curves), taken from Fig. S2B. The blue solid curves were computed from atomic MT models with 12 (A), 13 (B), 14 (C) or 15 (D) protofilaments, each containing 48 tubulin dimers along the long MT axis. The dimers were arranged in a 3-start left-handed helical lattice, whose radii were 10.2, 11.05, 11.9, and 12.75 nm, corresponding to 12, 13, 14, and 15 protofilaments, respectively. The pitch in all the models was 12.214 nm, in agreement with PDB 3J6F (Alushin *et al.*, 2014). A solvation layer was added to the MT models, using a Probe Radius of 0.14 nm and the solvation overlap correction method, described in §10.2. The layer thickness was 0.28 nm and its electron density was $364 \text{ e} \cdot \text{nm}^{-3}$. The inset shows the low q -range on an expanded scale. Intensity curves were computed in about half a minute on an NVIDIA Titan GPU, using the Hybrid method.

9. Validation tests of D+

D+ has been extensively tested, using the hardware architecture described in §6.1.3, in order to ensure that its computations are correct and consistent with other programs. In §9.1, geometric models were compared with X+ (Ben-Nun *et al.*, 2010), which uses analytical expressions when possible. In §9.2, atomic models were compared with CRY SOL (ATSAS 2.8.2) (Svergun *et al.*, 1995) or *DebyeCalculator*, which uses Debye formula (Debye, 1915). Computations in CRY SOL were done using its maximum accuracy settings, including 99 as the order of spherical harmonics terms and 18 as the order of the Fibonacci grid (when relevant). In §9.3, the geometric models were cross-validated using atomic structures that assumed the corresponding geometric shapes.

In addition, various cross validation tests were designed to check that D+ is consistent within itself by constructing the same structures using different options in D+ and comparing the results. We present several examples that demonstrate how D+ was tested. The full set of tests is larger and provided as an additional supporting folder that is part of the software package. In each test, we show comparable curves and a zoom into a section where small deviations between curves can be observed. At each q value, we also present the normalized residuals:

$$R(q) = \frac{I_{test}(q) - I_{ref}(q)}{I_{test}(q)}. \quad (\text{S2})$$

I_{test} is the tested intensity, computed by D+. I_{ref} is the reference curve, computed by CRY SOL (ATSAS 2.8.2) (Svergun *et al.*, 1995), or X+ (Ben-Nun *et al.*, 2010), or *DebyeCalculator*, or by D+ using an alternative way of computation. The differences between the intensities are normalized to the magnitude of the intensity. Although I_{test} is close to I_{ref} , the analytical results of X+ may lead to values of I_{ref} , which can be very close to zero. We therefore normalized the residuals by I_{test} rather than

I_{ref} . In those cases, we plotted the normalized absolute residuals, $|R|$, on logarithmic scales, which are suitable to present a wide range of residual magnitudes. The wide range was owing to the sharp minima in the scattering curves where the normalized absolute residuals approached unity, whereas the other parts of the curves approached zero. Some of the tests can be found in `./Example Files/` directory, where D+ is installed.

9.1. Geometric models

9.1.1. Cylindrical models Fig. S7 compares between the scattering curve from a model of a single 100 nm long rod with a radius of 1 nm, computed in X+, with a similar model, computed in D+, using a **Grid Size** of 330. The same model was then constructed in D+ from 10 identical 10 nm long rods with the same radius, placed on top of one another, using a space-filling 1D symmetry (see cartoon in Fig. S7). In the second computation, the **Grid Size** of the small rod was only 50 and the Hybrid method was used to compute the Space-Filling symmetry. The results were very similar up to negligible small numerical errors.

Fig. S7. Solution X-ray scattering intensity curve from 100 nm long uniform rod with a radius of 1 nm and electron density of $400 \text{ e}/\text{nm}^3$, computed in X+ (solid black curve) or in D+ using GPU, adaptive (VEGAS) Monte Carlo integration and **Grid Size** of 330 (dashed red curve). The same structure was computed in D+ by computing the form-factor of 10 nm long rod and placing it in a Space-Filling symmetry containing 10 identical units on top of one another (see cartoon). The **Grid Size** of the short rod was 50 and the Space-Filling symmetry was computed using the Hybrid method (dotted blue curve). The inset presents, on an expended scale, a small part of the results where the deviation between the methods is large. The top curve shows the normalized residuals, R , where the reference curve is the intensity computed by X+ (Eq. S2).

Fig. S8 compares between models of uniform concentric cylinders of finite height computed in X+, with those computed in D+. Within D+, the same models were computed in several different ways. All the computations gave very similar results.

Fig. S8. Solution X-ray scattering intensity from 10 nm long uniform rod surrounded by two concentric hollow cylinders. The radius of the rod is 1 nm and its electron density is $400 \text{ e}/\text{nm}^3$. The second shell is 1 nm thick and its electron density is $500 \text{ e}/\text{nm}^3$. The third shell is 1 nm thick and its electron density is $600 \text{ e}/\text{nm}^3$. The solid black curve was calculated with X+. The other broken curves were computed by D+ using different options as follows. The model was constructed using a single cylindrical geometry with three shells (as in X+) and was computed either on the CPU (olive), using adaptive Gauss-Kronrod integration, or on the GPU (navy), using adaptive (VEGAS) Monte Carlo integration. We then created the same structure using a rod and two separate concentric hollow cylinders with one shell each. The three models (leaves in the data tree structure) were then positioned in a concentric symmetry using the Manual Symmetry option of D+. The models were then calculated on CPU (pink) using the adaptive Gauss Kronrod integration method and on the GPU (blue) using the adaptive (VEGAS) Monte Carlo integration option. The same structure was constructed from two 5 nm long concentric cylinders that were placed on top of one another using Manual Symmetry (CPU-violet and GPU - red). The inset shows a small part of the curves on an expanded scale. The cartoon shows a side-view and a top-view of the computed models. The top curves show the normalized residuals, R , where the reference curve is the intensity computed by X+ (Eq. S2).

9.1.2. *Spherical models* Fig. S9 compares between models of uniform concentric spheres computed in X+ with those computed in D+. Within D+, the same models were computed in several different ways. All the computations gave very similar results.

Fig. S9. Solution X-ray scattering intensity from a uniform sphere surrounded by two concentric uniform hollow spherical shells. The radius of the core sphere is 1 nm and its electron density is 400 e/nm^3 . The first spherical shell is 1 nm thick, its inner radius is 1 nm, and its electron density is 500 e/nm^3 . The second spherical shell is 1 nm thick, its inner radius is 2 nm, and its electron density is 600 e/nm^3 . The solid (black) curve was computed in X+. The other broken curves were computed by D+, using different options as follows. The model was constructed using a single spherical geometry with three layers (as in X+) and was computed either on the CPU (green) and GPU (pink), using Monte Carlo integration. We then created the same structure using a sphere and two separate concentric hollow spherical shells. The three models (leaves in the data tree structure) were then positioned in a concentric symmetry using the Manual Symmetry option of D+. The models were then computed using CPU (blue) or GPU (red), using Monte Carlo integration. The inset shows a small part of the curves on an expanded scale. The cartoons show a side view and a cut through the computed structures. The top curves show the normalized absolute residuals, $|R|$, where the reference curve is the intensity computed by X+ (Eq. S2).

9.1.3. Slab models Fig. S10 compares between a model of an infinite multi-layered slabs, computed by X+, with similar symmetric slab models computed by D+, using finite size structures. A $100 \times 100 \text{ nm}^2$ multi-layered slab was computed by the Direct method. Then, a $1,000 \times 1,000 \text{ nm}^2$ multi-layered slabs was computed by constructing each slab from 10,000 slabs of $10 \times 10 \text{ nm}^2$, using the scripted symmetry option of D+. The deviations between the 3 models were small. The larger finite sized slabs were somewhat closer to the finite model of X+. Fig. S11 shows an equivalent comparison, using the asymmetric slab model of D+ instead of the symmetric slab model. Both models adequately fit to the results of X+.

Fig. S10. Solution X-ray scattering intensity from layers of symmetric slabs. Each slab has an inner layer that is 2 nm thick with an electron density of 280 e/nm^3 that is sandwiched between two outer slabs that are each 1 nm thick and their electron density is 400 e/nm^3 . The solvent electron density is 334 e/nm^3 . (see cartoon). The black solid curve was computed, using an infinite slab in X+. The broken red curve was computed in D+, using $1,000 \times 1,000 \text{ nm}^2$ slab that was constructed from 10,000 slabs of $10 \times 10 \text{ nm}^2$ using Scripted Symmetry and **Grid Size** of 100. Orientation average was calculated using adaptive (VEGAS) Monte Carlo integration. The broken blue curve was computed on CPU with the Direct method in D+, using a $100 \times 100 \text{ nm}^2$ slab. Orientation average was calculated using adaptive Gauss Kronrod integration. The curves at the top show the normalized absolute residuals, $|R|$, where the reference curve is the intensity computed by X+ (Eq. S2).

Fig. S11. Solution X-ray scattering intensity from layers of asymmetric slabs. Each slab has an inner layer that is 2 nm thick with an electron density of $280 \text{ e}/\text{nm}^3$, which is sandwiched between two outer slabs, each 1 nm thick. One of the outer slabs has an electron density of $400 \text{ e}/\text{nm}^3$ and the other has an electron density of $450 \text{ e}/\text{nm}^3$ (see cartoon). The black curve was computed, using an infinite slab in X+. The broken blue curve was computed in D+, using $1,000 \times 1,000 \text{ nm}^2$ slab that was constructed from 10,000 slabs of $10 \times 10 \text{ nm}^2$ using Scripted Symmetry and Grid Size of 100. Orientation average was calculated using adaptive (VEGAS) Monte Carlo integration. The broken red curve was computed on CPU with the Direct method in D+, using a $100 \times 100 \text{ nm}^2$ slab. Orientation average was calculated using adaptive Gauss Kronrod integration. The top curve shows the normalized absolute residual, $|R|$, where the reference curve is the intensity computed by X+ (Eq. S2).

Fig. S12 compares between $100 \times 100 \text{ nm}^2$ symmetric slab models within D+, computed in different ways. The Direct method was computed on the CPU and compared with several GPU computations. The GPU computations were done with either a single $100 \times 100 \text{ nm}^2$ slab model or with 100 identical units of $10 \times 10 \text{ nm}^2$ slabs. The

Hybrid method with either Space-Filling symmetry, or Scripted Symmetry, or Manual Symmetry were used to compute the 100 units. All the methods gave very similar results.

Fig. S12. Solution X-ray scattering intensity from layers of symmetric slabs. Each slab has an inner layer that is 2 nm thick with an electron density of $280 \text{ e}/\text{nm}^3$ that is sandwiched between two outer slabs that are each 1 nm thick and their electron density is $400 \text{ e}/\text{nm}^3$. The solvent electron density is $334 \text{ e}/\text{nm}^3$ (see cartoon). All curves were computed in D+, using $100 \times 100 \text{ nm}^2$ slabs with different methods. Direct computation from Fig. S10 (black) is compared to a Direct computation on CPU using the asymmetric slab model (red). Several computations were done using GPU as follows. A single $100 \times 100 \text{ nm}^2$ slab was computed with a **Grid Size** of 460 and orientation average was computed using adaptive (VEGAS) Monte Carlo integration (magenta). Alternatively, the Hybrid method was used to compute the scattering of the same structure from 100 copies of $10 \times 10 \text{ nm}^2$ slabs, for which a **Grid Size** of 100 was used. To create the large slab, Scripted Symmetry (purple), Space-Filling Symmetry (yellow), and Manual Symmetry (blue) were used. The top curve shows the normalized residual, R , where the reference curve is the intensity computed by D+ on the CPU, using the Direct method (Eq. S2).

Fig. S13 compares between a stack of 10 infinite multi-layered slabs, computed by X+, with a model of a stack of 10 finite slabs (or rectangular boxes) models that formed

a similar stack of large uniform slabs, computed by D+. The results are adequately similar, considering the fact that D+ computes finite size slabs and X+ computes infinite size slabs.

Fig. S13. Solution X-ray scattering intensity from Multi-layered structure containing 10 layers of symmetric slabs. Each slab has an inner layer that is 2 nm thick with an electron density of 280 e/nm^3 that is sandwiched between two outer slabs that are each 1 nm thick and their electron density is 400 e/nm^3 . Between adjacent layers there is a 6 nm water gap with an electron density of 334 e/nm^3 (see cartoon). The solid black curve was computed, using a stack of infinite slabs in X+. The broken (blue) curve was computed in D+, using $1,000 \times 1,000 \text{ nm}^2$ slabs. Each slab was constructed from 10,000 slabs of $10 \times 10 \text{ nm}^2$ using Scripted Symmetry and Grid Size of 100. Orientation average was calculated using adaptive (VEGAS) Monte Carlo integration. The top curve shows the normalized absolute residual, $|R|$, where the reference curve is the intensity computed by X+ (Eq. S2).

9.2. Atomic models

We compared the scattering curves from atomic models of D+ with those of CRY SOL (ATSAS 2.8.2) (Svergun *et al.*, 1995), and *DebyeCalculator*. *DebyeCalculator* uses the equations of D+ but instead of numerical orientation average it uses Debye formula (Debye, 1915), which applies analytic orientation averaging. Debye formula is applicable for spherically-symmetric objects like atoms.

MOLProbity (Chen *et al.*, 2010) was used to add hydrogen atoms to PDB files. PyMOL (Schrödinger, LLC, 2015) was used to add charges to residues that were supposed to be charged but the PDB file did not include the charges. The atomic form factor of ions, like O^- , differ from the form-factor of the corresponding neutral atom (O, in this example). Both D+ and *DebyeCalculator* take into account the atomic form factor of the 209 atoms and ions listed in the International Tables for Crystallography, using the five-Gaussian approximation (Hamilton, 1974; Marsh & Slagle, 1983). N^+ , however, is not included in the International Tables for Crystallography and both D+ and *DebyeCalculator* computes the form factor of an N atom instead. If both hydrogen atoms and charges are added to an N^+ group, an excess electron is obtained. An approximation that might be used to better account for an N^+ ion, is to manually remove from the PDB file, one of the hydrogen atoms next to the N^+ ion. This ensures that the correct total number of electrons in the protein is preserved.

CRY SOL does not take the contribution from any ion into account. Hence, to properly compare between CRY SOL and D+ (or *DebyeCalculator*), charges were either not added or removed from the PDB files. Computations in CRY SOL were done using the parameters that provide maximum accuracy. Hence, 99 was the maximum order of harmonics, and 18 was the order of Fibonacci grid.

In all the computations in D+, 0.0001 was the **Convergence** tolerance, the solvent electron density was $334 e/nm^3$, and the type of solvent radius was **Dummy Atoms**. Unless

otherwise is indicated, **Monte Carlo (Mersenne Twister)** was the orientation averaging integration method.

9.2.1. Tubulin Dimer Tubulin heterodimer is made of α -tubulin and β -tubulin, which are homologous and form the so called tubulin dimer. Depending on solution conditions, tubulin dimer may assemble into MT and/or tubulin rings (Diaz *et al.*, 1994). Fig. S14 compares between the scattering curve from the central tubulin dimer in PDB ID 3J6F (which contains nine dimers), by *DebyeCalculator*, CRY SOL, and D+. H atoms were explicitly added to the PDB file (Chen *et al.*, 2010). Fig. S15 shows a similar comparison, however, H atoms were only implicitly taken into account. Figs. S16 and S17 repeats the same comparison, but without removing the charges from the PDB file.

Fig. S14. Solution X-ray scattering intensity from an atomic model of tubulin dimer (PDB ID 3J6F), shown as a ribbon diagram. H atoms were explicitly added to the PDB file (Chen *et al.*, 2010). The solid black curve was computed by *DebyeCalculator*. The red broken curve was computed by CRY SOL and the broken pink and green curves were computed by D+ at low (L, pink broken curve) and high (H, green broken curve) resolution. **Grid Size** of 100 and **Convergence** tolerance of 10^{-4} were used for low resolution computations, and **Grid Size** of 250 and **Convergence** tolerance of 10^{-7} were used for high resolution. The inset shows the same computations on an expanded scale. The top curve shows the normalized residual, R , where the reference curve is the intensity computed by *DebyeCalculator* (Eq. S2).

Fig. S15. Solution X-ray scattering intensity from an atomic model of tubulin dimer (PDB ID 3J6F), shown as a ribbon diagram. H atoms were implicitly taken into account. The solid black curve was computed by *DebyeCalculator*, which applies Debye formula (Debye, 1915). The red broken curve was computed by CRY SOL and the broken pink and green curves were computed by D+ at low (L, pink) or high (H, green) resolution. **Grid Size** of 100 and **Convergence** tolerance of 10^{-4} were used for low resolution computations. For more accurate computation, the **Grid Size** was increased to 250 and the **Convergence** tolerance was decreased to 10^{-7} . The inset shows the same computations on an expanded scale. The top curve shows the normalized residual, R , where the reference curve is the intensity computed by *DebyeCalculator* (Eq. S2).

Fig. S16. Solution X-ray scattering intensity from an atomic model of tubulin dimer (PDB ID 3J6F), shown as a ribbon diagram, where α -tubulin is in green and β -tubulin is in red. H atoms were explicitly added to the PDB file (Chen *et al.*, 2010). Charges (on charged residues) were added using PyMOL (Schrödinger, LLC, 2015). The solid black curve was computed by *DebyeCalculator*, which applies Debye formula (Debye, 1915). The red broken curve was computed by CRY SOL and the broken pink and green curves were computed by D+ at low (L, pink) and high (H, green) resolution. **Grid Size** of 100 and **Convergence** tolerance of 10^{-4} were used for low resolution computations. For high resolution computations, the **Grid Size** was increased to 250 and the **Convergence** tolerance was decreased to 10^{-7} . The inset shows the same computations on an expanded scale. The top curve shows the normalized residual, R , where the reference curve is the intensity computed by *DebyeCalculator* (Eq. S2).

Fig. S17. Solution X-ray scattering intensity from an atomic model of tubulin dimer (PDB ID 3J6F), shown as a ribbon diagram. H atoms were taken implicitly into account. Charges (on charged residues) were added using PyMOL (Schrödinger, LLC, 2015). The solid black curve was computed by *DebyeCalculator*, which applies Debye formula (Debye, 1915). The red broken curve was computed by CRY SOL and the broken pink and green curves were computed by D+ at low (L, pink) and high (H, green) resolution. **Grid Size** of 100 and **Convergence** tolerance of 10^{-4} were used for low resolution computations. For high resolution computations, the **Grid Size** was 250 and the **Convergence** tolerance was 10^{-7} . The inset shows the same computations on an expanded scale. The top curve shows the normalized residual, R , where the reference curve is the intensity computed by *DebyeCalculator* (Eq. S2).

Fig. S18 compares between different computation methods within D+. Tubulin dimers, with explicitly added H atoms and after removing the charges, were computed in different ways in D+. The computations are compared with *DebyeCalculator*, whose

results were comparable with the results of CRY SOL (Fig. S14). The computations of $D+$ were either done on CPU or GPU, using the Direct method (implemented only for CPU) and the reciprocal grid (RG) algorithm. When using the RG algorithm, different orientation average methods were used: Adaptive Gauss-Kronrod (using CPU), adaptive (VEGAS) Monte Carlo (using GPU), or regular Monte Carlo (using CPU or GPU).

Fig. S18. Solution X-ray scattering intensity from an atomic model of tubulin dimer (PDB ID 3J6F), shown as a ribbon diagram. H atoms were explicitly added to the PDB file (Chen *et al.*, 2010). The solid black curve was computed by *DebyeCalculator*, which applies Debye formula (Debye, 1915). The broken green curve was computed by $D+$ using the Direct method (on CPU). The other broken curves were computed by the reciprocal grid (RG) algorithm of $D+$, where orientation average was computed by the following integration algorithms: Adaptive Gauss-Kronrod (magenta - using CPU only, purple - using GPU for computing the grid and CPU for orientation average), adaptive (VEGAS) Monte Carlo (cyan, using GPU), or Monte Carlo (blue, using CPU, red using GPU). The inset shows a small part of the curve on an expanded scale. The top curves show the normalized residual, R , where the reference curve is the intensity computed by *DebyeCalculator* (Eq. S2).

9.2.2. Tubulin ring Under certain experimental conditions (high tubulin concentration and low temperature), tubulin dimers may assemble into rings (Diaz *et al.*, 1994). To get the ring structure the dimer subunit should be docked into an assembly symmetry, which involves both rotations and translations. To compute the scattering curve from a tubulin ring, we used the same central tubulin dimer from PDB ID 3J6F and explicitly added H atoms. We then rotated the dimer by $\gamma = 180^\circ$ (about the z -axis), $\alpha = 90^\circ$ (about the x -axis), and $\beta = 50^\circ$ (about the y -axis). This rotation puts the dimer in a parallel direction to the tangent of the ring. The ring itself was built from 13 dimers that were rotated about the z axis by $\gamma = \frac{360^\circ}{13}n$ (in the plane of the ring), and translated by $x = R_{Ring} \sin\left(\frac{360^\circ}{13}n\right)$ and $y = R_{Ring} \cos\left(\frac{360^\circ}{13}n\right)$ (where $n \in [0, 1, \dots, 12]$). A PDB file of the ring (containing all the 13 dimers) was then created and its scattering curve was computed by CRY SOL and by *DebyeCalculator*. The results were compared with the computations of D+ using the PDB file of the rotated dimer and the Hybrid method to account for the assembly symmetry of the ring. Grid was computed only for the dimer. The computations are compared in Fig. S19, using implicit H atoms, and in Fig. S20, using explicit H atoms.

Fig. S19. Solution X-ray scattering intensity from an atomic model of tubulin ring, shown as a ribbon diagram. H atoms were implicitly taken into account (PDB ID 3J6F). The solid black curve was computed by *DebyeCalculator*, which applies Debye formula (Debye, 1915). The broken red curve was computed by CRY SOL. Both used a single PDB file for the entire ring. The broken blue curve was computed by D+ on a GPU, using a **Grid Size** of 200 for the tubulin dimer, and the Hybrid method for the ring symmetry. Orientation average was computed using Monte Carlo integration with **Convergence** tolerance of 10^{-4} . The inset shows a small part of the curve on an expanded scale. The top curves show the normalized residuals, R , where the reference curve is the intensity computed by *DebyeCalculator* (Eq. S2).

Fig. S20. Solution X-ray scattering intensity from an atomic model of tubulin ring, shown as a ribbon diagram. H atoms were explicitly added to the PDB file of the dimer (Chen *et al.*, 2010) (PDB ID 3J6F). The solid black curve was computed by *DebyeCalculator*, which applies Debye formula (Debye, 1915). The broken red curve was computed by CRY SOL. The broken blue curve was computed by D+ on GPU, using a **Grid Size** of 200 for the tubulin dimer, and the Hybrid method for the ring symmetry. Orientation average was computed using Monte Carlo integration with **Convergence** tolerance of 10^{-4} . The inset shows a small part of the curve on an expanded scale. The top curves show the normalized residual, R , where the reference curve is the intensity computed by *DebyeCalculator* (Eq. S2).

We then compared different ways to compute the scattering from tubulin ring, within D+. The results are shown in Fig. S21 (for explicit H atoms) and in Fig. S22 (for implicit H atoms). Computations were done by using a single PDB file for the entire ring by either D+ or *DebyeCalculator*, or in D+ using the grid of the dimer and the Hybrid method for the ring symmetry, or using the grid for the entire ring symmetry, or using the Direct method.

Fig. S21. Solution X-ray scattering intensity from an atomic model of tubulin ring, shown as a ribbon diagram. H atoms were explicitly added to the PDB file of the dimer (PDB ID 3J6F) (Chen *et al.*, 2010). Unless otherwise indicated, computations were done on GPU. The solid black curve was computed by *DebyeCalculator*, which applies Debye formula (Debye, 1915). The broken red curve was computed by D+ on the CPU, using the Direct method. The broken blue curve was computed by D+ using a single PDB file for the entire ring (which was used for CRY SOL and *DebyeCalculator*). The broken magenta curve was computed using a grid of tubulin dimer (**Grid Size** of 200) in a Manual Symmetry that was computed using the Hybrid method. The broken green curve was computed by the dimer in Manual Symmetry using a **Grid Size** of 200 for both the tubulin and the ring. The broken purple curve was computed using the grid of the dimer (**Grid Size** of 200) in a Scripted Symmetry, which was computed by the Hybrid method. The inset shows a small part of the curves on an expanded scale. The top curves show the normalized residuals, R , where the reference curve is the intensity computed by *DebyeCalculator* (Eq. S2).

Fig. S22. Solution X-ray scattering intensity from an atomic model of tubulin ring, shown as a ribbon diagram. H atoms were implicitly taken into account. Unless otherwise indicated, computations were done on GPU. The solid black curve was computed by *DebyeCalculator*, which applies Debye formula (Debye, 1915). The broken red curve was computed by D+ on the CPU, using the Direct method. The broken blue curve was computed by D+ using a single PDB file for the entire ring. The broken magenta curve was computed from a grid of tubulin dimer (PDB ID 3J6F) in a Manual Symmetry, using the Hybrid method. The broken green curve was computed from the dimer in a Manual Symmetry, using a grid for the ring. The broken purple curve was computed from the grid of the dimer in a Scripted Symmetry, using the Hybrid method. The size of the grid was fixed at 200. The inset shows a small part of the curve on an expanded scale. The top curves show the normalized residual, R , where the reference curve is the intensity computed by *DebyeCalculator* (Eq. S2).

9.2.3. Tubulin is a Space-Filling symmetry To test the option of Space-Filling symmetry with atomic models, we used the same central tubulin dimer from PDB ID 3J6F. No H atoms or charges were added to the PDB file. The dimer was docked into a simple cubic Space-Filling symmetry (Fig. S23A). The scattering curve was computed by D+, using different options for defining the assembly symmetry (Space-Filling, Scripted, or Manual), using Grids for the entire structure or the Hybrid method where a Grid was computed for the dimer, or by loading the precomputed scattering amplitude from the dimer and putting it in the assembly symmetry. Each computation was repeated with different orientation average integration method: Monte Carlo, adaptive (VEGAS) Monte Carlo, or adaptive Gauss Kronrod. A PDB file of the entire cubic structure (containing 18 dimers) was then created and its scattering curve was computed by *DebyeCalculator*. Fig. S23A shows that all the computations of D+ resulted in the same curve, which agreed with the curve obtained by *DebyeCalculator*. Fig. S23B shows that *DebyeCalculator* is in good agreement with CRY SOL. When solvation layer was added to the PDB file of the entire structure (Fig. S23C) or to the dimer (Fig. S23D), all the symmetry generation and integration methods led to similar results.

Fig. S23. Solution X-ray scattering intensity from an atomic model of tubulin in a simple cubic Space-Filling symmetry, shown as a ribbon diagram. The central tubulin dimer from PDB ID 3J6F was rotated by $\gamma = 180^\circ$ (about the z -axis), and $\alpha = 180^\circ$ (about the x -axis). H atoms were implicitly taken into account. The PDB was then docked into a simple cubic Space-Filling symmetry with 3 subunits in the x direction with a repeat distance of 4 nm, 2 subunits in the y direction with a repeat distance of 4 nm, and 3 subunits in the z direction with a repeat distance of 8.13 nm. **A.** Scattering curves were computed in D+ using different options to define the assembly symmetry (Space-Filling, Scripted Symmetry, or Manual Symmetry), using Grids, or the Hybrid method (with a Grid for the dimer), or by loading the precomputed scattering amplitude from the dimer. Each computation was repeated using different orientation average integration methods: Monte Carlo, adaptive (VEGAS) Monte Carlo, or Gauss Kronrod. A PDB file of the entire structure (containing all the 18 dimers) was then created and its scattering curve was computed by *Debye-Calculator*. **B.** The curve that was computed by *Debye-Calculator* in **A** (solid red curve) is compared with the curve computed by CRY SOL for the same PDB (broken curve). **C.** The PDB of the entire structure was computed with a solvation layer whose thickness was 3 nm and electron density $364 \text{ e} \cdot \text{nm}^{-1}$. The computation was done by computing Grids or by loading a precomputed scattering amplitude. Orientation average was done using Monte Carlo, adaptive (VEGAS) Monte Carlo, or Gauss Kronrod integrations. **D.** The computation from **A** were repeated, using solvated tubulin dimer. The solvation layer was 3 nm thick and its electron density was $364 \text{ e} \cdot \text{nm}^{-1}$.

9.3. Atomic vs. geometric models

We created PDB files in which oxygen atoms were randomly close packed into geometric shapes. We used the PDB files to compare their scattering curve with the curves of their corresponding geometric models. All the curves were computed in D+. The PDB files were computed using the `Solvent Only` option in D+, where the solvent method was `Dummy Atoms Voxelized`, which gave the voxels forming the enveloping geometric shape (the `Solvent Only` option did not compute the form-factors of the oxygen atoms).

Fig. S24 compares atomic with geometric spherical models. Fig. S25 compares a solvated atomic spherical model with the equivalent geometric core-shell spherical model, where the shell corresponds to the solvation layer. We then compare between atomic and geometric models of a hollow sphere (Fig. S27) and then the corresponding solvated hollow sphere models (Fig. S27), in which solvation layers are both internal and external. The comparisons validate the solvation algorithm used in D+. We then tested the cylindrical (Fig. S28) and the slab (Fig. S29) geometries. Within the tested q -range (between 0 and 5 nm^{-1}), all the atomic and their corresponding geometric models were very similar (in other words, the residuals were very small).

Fig. S24. Solution X-ray scattering intensity of atomic and geometric spherical models.

The solid black curve was computed from a PDB file, in which oxygen atoms were randomly close packed into a sphere. The computation was done in D+ from the atomic model using Solvent Only, Dummy Atoms Voxelized, solvent electron density of $1 \text{ e}/\text{nm}^3$, and no solvation shell. The broken (red) curve was computed in D+, using the corresponding geometric sphere model with a radius of 1.113 nm and an electron density of $1 \text{ e}/\text{nm}^3$. The solvent electron density was set to 0, corresponding to vacuum. The red curve at the top inset shows the normalized absolute residuals, $|R|$, as a function of the magnitude of the scattering vector, q , where the reference curve is the intensity of the atomic model (Eq. S2). The cartoon shows the closed packed spherical atomic model in Red and the corresponding geometric model in green.

Fig. S25. Solution X-ray scattering intensity of solvated atomic spherical model and a geometric model of a spheres and a thin shell. The solid black curve was computed from a PDB file, in which oxygen atoms were randomly close packed into a sphere. The computation was done in D+ from the atomic model using Solvent Only, Dummy Atoms Voxelized, solvent electron density $1 \text{ e}/\text{nm}^3$, an outer solvent electron density of $-1 \text{ e}/\text{nm}^3$, and solvation thickness of 0.30 nm . These parameters produced a second shell with an electron density contrast of $+1$ to the core. The broken (red) curve was computed in D+, using the corresponding geometric core-shell spherical model, with a core radius of 1.113 nm and an electron density of $1 \text{ e}/\text{nm}^3$, and an outer shell with a thickness of 0.30 nm and an electron density of $2 \text{ e}/\text{nm}^3$. The top curve shows the normalized absolute residual, $|R|$, as a function of q , where the reference curve is the intensity computed from the atomic model (Eq. S2). The cartoon shows a cross section of the geometric model.

Fig. S26. Solution X-ray scattering intensity of an atomic model of a hollow sphere and a geometric model of a hollow sphere. The solid black curve was computed from a PDB file, in which oxygen atoms were randomly close packed into a hollow sphere. The computation was done in D+ from the atomic model using Solvent Only, Dummy Atoms Voxelized, solvent electron density $1 \text{ e}/\text{nm}^3$ and no solvation shell. The broken (red) curve was computed in D+, using the corresponding geometric hollow sphere model with an inner radius of 0.868 nm and an outer radius of 1.130 nm , and an electron density of $1 \text{ e}/\text{nm}^3$. The solvent electron density was set to 0, corresponding to vacuum. The top curve shows the normalized absolute residual, $|R|$, as a function of q , where the reference curve is the intensity computed from the atomic model (Eq. S2). The cartoon shows a cross section of the atomic model in red.

Fig. S27. Solution X-ray scattering intensity from a solvated hollow sphere atomic model and a geometric concentric multi layered spherical shell model. The solid black curve was computed using the atomic model from Fig. S26 with Solvent Only, Dummy Atoms Voxelized, solvent electron density $1 \text{ e}/\text{nm}^3$, solvation thickness 0.3 nm , and an outer solvent electron density of $-1 \text{ e}/\text{nm}^3$. The **Fill Holes** check-box remained unchecked. The broken (red) curve was computed in D+, using the corresponding concentric multi layered spherical shell model. The first spherical shell is 0.301 nm thick, its inner radius is 0.575 nm , and its electron density is $2 \text{ e}/\text{nm}^3$. The second spherical shell is 0.256 nm thick, and its electron density is $1 \text{ e}/\text{nm}^3$. The third spherical shell is 0.295 nm thick, and its electron density is $2 \text{ e}/\text{nm}^3$. The top curve shows the normalized absolute residual, $|R|$, as a function of q , where the reference curve is the intensity computed from the atomic model (Eq. S2). The cartoon shows a cross section of the geometric model.

Fig. S28. Solution X-ray scattering intensity from an atomic rod model and a geometric model of a uniform rod. The solid black curve was computed from a PDB file, in which oxygen atoms were randomly close packed into a rod. The computation was done in D+ from the atomic model with Solvent Only, Dummy Atoms Voxelized, solvent electron density 1 e/nm^3 , and no solvation. The broken (red) curve was computed in D+, using the corresponding rod model, with a rod radius of 1.088 nm , rod height of 2.121 nm , and an electron density of 1 e/nm^3 . The solvent electron density was set to 0, corresponding to vacuum. The top curve shows the normalized absolute residual, $|R|$, as a function of q , where the reference curve is the intensity computed from the atomic model (Eq. S2). The cartoon shows that atomic model in Red and the corresponding geometric model in green.

Fig. S29. Solution X-ray scattering intensity from atomic and geometric models of a uniform rectangular slab. The solid black curve was computed from a PDB file, in which oxygen atoms were randomly close packed into a rectangular slab. The computation was done in D+ from the atomic model with Solvent Only, Dummy Atoms Voxelized, solvent electron density $1 \text{ e}/\text{nm}^3$, and no solvation. The broken (red) curve was computed in D+, using the corresponding uniform slab model with an electron density of $1 \text{ e}/\text{nm}^3$ and dimensions of $3.055 \text{ nm} \times 2.115 \text{ nm} \times 4.108 \text{ nm}$. The solvent electron density was set to 0, corresponding to vacuum. The top curve shows the normalized residual, $|R|$, as a function of q , where the reference curve is the intensity computed from the atomic model (Eq. S2). The cartoon shows the atomic model in red and the geometric model in green.

9.4. Fitting algorithm tests

The fitting algorithm of D+ was tested against calculated (Figs. S30,S31,S32), and experimental scattering curves of lipid membranes (Fig. S33), or protein solutions (Figs. S34 and S35). A scattering signal, with known parameters, served as the target

signal, to which D+ was trying to fit its computed curve. The mutable model parameters were described in the figure captions with the initial values (in parenthesis) and the final fitted values (after the parenthesis). The figures show adequate fits.

Fig. S30. Fitting to a solution X-ray scattering intensity curve that was computed from a model of a uniform sphere surrounded by four concentric uniform hollow spherical shells. The radius of the internal sphere is (0.15)0.1 nm and its electron density is 400 e/nm³. The first spherical shell is (0.15)0.1 nm thick, and its electron density is 500 e/nm³. The second spherical shell is (0.05)0.1 nm thick, and its electron density is 600 e/nm³. The third spherical shell is (0.08)0.1 nm thick, and its electron density is 700 e/nm³. The forth spherical shell is (0.15)0.1 nm thick, and its electron density is 800 e/nm³. The solid black curve was computed in D+, and used as the target signal for the fitting. The broken magenta curve was computed by D+, using the initial parameters (given in the parenthesis, if mutable) and the broken red curve is the result of the fit. Fitting was done using Levenberg-Marquardt algorithm, Ratio Residuals, Trivial Loss, with 20 iterations. Step Size was 10⁻⁵, the Convergence tolerance was 10⁻⁶, and the Der eps 10⁻⁶. Intensity computations were done with a Grid Size of 30 and the Gauss Kronrod integration algorithm.

Fig. S31. Fitting to a solution X-ray scattering intensity curve that was computed from a model of a uniform rod surrounded by two concentric uniform cylindrical shells. The radius of the internal rod is (0.15)0.1 nm and its electron density is $400 \text{ e}/\text{nm}^3$. The first cylindrical shell is (0.2)0.1 nm thick, and its electron density is $600 \text{ e}/\text{nm}^3$. The second cylindrical shell is (0.15)0.1 nm thick, and its electron density is $800 \text{ e}/\text{nm}^3$. The solid black curve was computed in D+, and used as the target signal for the fitting. The broken magenta curve was computed by D+, using the initial values in parenthesis and the red broken curve is the result of the fit. Fitting was done using Levenberg-Marquardt algorithm, Ratio Residuals, Trivial Loss, with 20 iterations. Step Size was 10^{-5} , the Convergence tolerance was 10^{-6} , and Der eps was 10^{-6} . Intensity computations were done with a Grid Size of 30 and the Gauss Kronrod integration algorithm.

Fig. S32.

Fitting to a solution X-ray scattering intensity curve that was computed from a model of asymmetric slab layers. The slabs were $1 \times 1 \text{ nm}^2$ in x-y plane and in the z direction the inner slab layer was 1 nm thick and electron density of $(250)280 \text{ e/nm}^3$. The layer was sandwiched between two outer 0.5 nm thick slabs, one with an electron density of $(300)400 \text{ e/nm}^3$ and the other with an electron density of $(600)450 \text{ e/nm}^3$. The solid black curve was computed in D+, and used as the target signal for the fitting. The broken magenta curve was computed by D+, using the initial values in parenthesis and the broken red curve is the result of the fit. Fitting was done using Levenberg-Marquardt method, Ratio Residuals, Trivial Loss, with 50 iterations. The Step Size was 10^{-5} , the Convergence tolerance was 10^{-6} , and Der eps 10^{-6} . Intensity computations were done with a Grid Size of 50 and the Gauss Kronrod integration algorithm.

Fig. S33. Fitting of a solution X-ray scattering intensity curve that was computed from a model of symmetric slab layers to experimental scattering data from a dispersion of 30 mg/ml 1,2-dipalmitoyl-sn-glycero-3-phosphocholine (DPPC) model membranes. In the model, the size of the slabs was 100 nm $\times$ 100 nm in the x-y plane and in the z direction a (0.6)0.66 nm thick inner layer with an electron density of 139 e/nm³ was sandwiched between two outer slabs. Each outer slab had two layers, the first, was (1.3)1.44 nm thick and had an electron density of 335 e/nm³, and the second was (0.9)0.66 nm thick and had an electron density of 540 e/nm³. The solid black curve was measured in our in-house rotating anode setup (Nadler *et al.*, 2011), and used as the signal for the fitting. The broken magenta curve was computed by D+, using the initial values in parenthesis and the broken red curve is the result of the fit. Fitting was done using Levenberg-Marquardt algorithm, Ratio Residuals, Trivial Loss, and 5 iterations. Step Size was 0.1, the Convergence tolerance was 10⁻⁶, and Der eps 10⁻⁶. Intensity computations were done with a Grid Size of 200 and the adaptive (VEGAS) Monte Carlo integration algorithm.

Fig. S34. Solution X-ray scattering data and models of Ubiquitin. Data taken from SASBDB are shown in black. The computed scattering curves are based on PDB ID 1UBQ. Hydrogen atoms were either implicitly taken into account or explicitly added to the PDB file by PyMol(Schrödinger, 2015). The broken scattering curves were computed by CRY SOL (ATSAS 2.8.2) (Svergun *et al.*, 1995), where the maximum order of harmonics was 50 and the order of Fibonacci grid was 18. The solid curves were computed by D+. In both D+ and CRY SOL the bulk water electron density was set to $334 \text{ e} \cdot \text{nm}^{-3}$ and the contribution of the solvation layer was taken into account. In D+, the **Grid Size** was 80, orientation average was computed by Monte Carlo integration, using Mersenne Twister algorithm. The solvent **Voxel Size** was 0.05 nm, the solvent **Probe Radius** was 0.14 nm and **Dummy Atoms** were used to account for the solvent excluded volume (see §5.5 in the main paper and Eq. 27). The fitting algorithm used the **Normal Residuals** cost function and the **Trivial Loss** function, with 20 iterations, **Step Size** of 0.01, **Convergence** tolerance of 0.01, and **Der eps** 0.1 (see §9.1 in the main paper). In the implicit model, the solvation thickness was (0.3)0.294 nm and its electron density was (364)336.6 $\text{e} \cdot \text{nm}^{-3}$. The excluded volume parameter, c_1 , in Eq. 24 was (1)1.014, and $R^2 = 0.941$. In the explicit model, c_1 was (1)1.012, the solvation thickness was (0.3)0.291 nm, its electron density was (364)335 $\text{e} \cdot \text{nm}^{-3}$, and $R^2 = 0.943$. In CRY SOL, in both models, the solvation thickness was 0.3 nm and the electron density of the solvation shell was 347 $\text{e} \cdot \text{nm}^{-3}$. In the implicit model $\chi^2 = 8.1$ and in the explicit model $\chi^2 = 8.0$.

Fig. S35. Solution X-ray scattering data and models of RNase. Data taken from SASBDB are shown in black. The computed scattering curve are based on PDB ID 1C0B. Hydrogen atoms were either implicitly taken into account or explicitly added to the PDB file by PyMol(Schrödinger, 2015). The broken scattering curves were computed by CRY SOL (ATSAS 2.8.2) (Svergun *et al.*, 1995), where the maximum order of harmonics was 50 and the order of Fibonacci grid was 18. The solid curves were computed by D+. In both D+ and CRY SOL the bulk water electron density was set to $334 \text{ e} \cdot \text{nm}^{-3}$ and the contribution of the solvation layer was taken into account. In D+, the **Grid Size** was 80, orientation average was computed by Monte Carlo integration, using Mersenne Twister algorithm. The solvent **Voxel Size** was 0.05 nm, the solvent **Probe Radius** was 0.14 nm and **Dummy Atoms** were used to account for the solvent excluded volume (see §5.5 in the main paper and Eq. 27). The fitting algorithm used the **Normal Residuals** cost function and the **Trivial Loss** function, with 20 iterations, **Step Size** of 0.01, **Convergence** tolerance of 0.01, and **Der eps** 0.1 (see §9.1) in the main paper). In the implicit model, the solvation thickness was (0.3)0.292 nm and its electron density was (364)363.8 $\text{e} \cdot \text{nm}^{-3}$. The excluded volume parameter, c_1 , in Eq. 24 was (1)1.003, and $R^2 = 0.998$. In the explicit model, c_1 was (1)1.012, the solvation thickness was (0.3)0.305 nm, its electron density was (364)352 $\text{e} \cdot \text{nm}^{-3}$, and $R^2 = 0.997$. In CRY SOL, in both models, the solvation thickness was 0.3 nm. In the implicit model the electron density of the solvation shell was 374 $\text{e} \cdot \text{nm}^{-3}$, and $\chi^2 = 2.67$. In the explicit model and the electron density of the solvation shell was 371 $\text{e} \cdot \text{nm}^{-3}$, and $\chi^2 = 2.65$.

10. Solvation layers of complexes

Solvation layers of large complexes can be computed in D+, in a scalable manner, based on the solvation layers of each of the subunits. Corrections, however, have to be included if the solvation layers of the subunits overlap when docked into the complex structure (Fig. S36). The errors can be corrected by subtracting the overlapping solvation parts.

We can demonstrate the approach with a simple case where three identical subunits form a chain, with overlaps in their solvation layer (Fig. S36). The solvation shell overlap of the central subunit is different than that of the subunits at the edges.

Fig. S36. A short chain containing three adjacent subunits. Each subunit is surrounded by a solvation layer, assuming it was isolated. The numbers correspond to different solvation amplitude scales. If the scale of the solvation layer is set to 1, then area 1 corresponds to a scale of 1, area 2 corresponds to a scale of 2, and area 3 corresponds to a scale of 1. Area 3 also contains the contribution from the subunit amplitude.

In D+, we can compute the amplitude of a monomer subunit, surrounded by a solvation layer ($f_{\text{Solvated Monomer}}$). The Solvated subunit can then be placed in a symmetry containing two repeats next to one another (Fig. S37). The scattering amplitude is then $f_{\text{Solvated Monomer}} \left(e^{i\vec{q} \cdot \vec{r}_1} + e^{i\vec{q} \cdot \vec{r}_2} \right)$. We can also compute the solvation layer amplitude, which is surrounding a single structure, containing the same two subunits next to one another, $f_{\text{Solvated Dimer}}$ (Fig. S38). In the latter case there are no overlapping

solvation layers.

Fig. S37. Two adjacent subunits, where each subunit is surrounded by its own solvation layer (Eqs. S3 and S4). The numbers correspond to different types of overlapping regions, as in Fig. S36.

Fig. S38. Solvated dimer. The solvation layer surrounds a single structure made of two identical adjacent subunits. The scattering amplitude is $f_{\text{Solvated Dimer}}$.

To construct the interface between two monomers, three amplitudes should be calculated: one for each solvated monomer $f_{\text{Solvated Monomer}}$, at its own position (Fig. S37 and Eqs. S3 and S4), and one for the solvated dimer, $f_{\text{Solvated Dimer}}$ (Fig. S38). The two monomers have the same magnitude of scattering amplitude but different phases.

If the position of the first monomer is at the origin, $\vec{r}_1 = 0$, and the second monomer is at $\vec{r}_2 = \vec{R}$, we get:

$$f_{\text{Solvated Monomer 1}} = f_{\text{Solvated Monomer}} \cdot e^{i\vec{q} \cdot \vec{r}_1} = f_{\text{Solvated Monomer}} \quad (\text{S3})$$

$$f_{\text{Solvated Monomer 2}} = f_{\text{Solvated Monomer}} \cdot e^{i\vec{q} \cdot \vec{r}_2} = f_{\text{Solvated Monomer}} \cdot e^{i\vec{q} \cdot \vec{R}} \quad (\text{S4})$$

The solvation correction amplitude is then given by:

$$F_{\text{Solvation correction}} = f_{\text{Solvated Dimer}} - (f_{\text{Solvated Monomer 1}} + f_{\text{Solvated Monomer 2}}) \quad (\text{S5})$$

Fig. S39 shows the result of Eq. S5, when the scale of each amplitude is set to 1.

Fig. S39. The correction amplitude obtained from Eq. S5. If the scale of the solvation layer is set to 1, then the scale of areas 2 and 3 is -1.

The solvated trimer is computed from three solvated monomers at $\vec{r}_1 = 0$, $\vec{r}_2 = \vec{R}$, and $\vec{r}_3 = 2\vec{R}$, and two Solvation corrections:

$$f_{\text{Solvated Trimer}} = f_{\text{Solvated Monomer}} \left(e^{i\vec{q} \cdot \vec{r}_1} + e^{i\vec{q} \cdot \vec{r}_2} + e^{i\vec{q} \cdot \vec{r}_3} \right) + F_{\text{Solvation correction}} \left(e^{i\vec{q} \cdot \vec{r}_1} + e^{i\vec{q} \cdot \vec{r}_2} \right) \quad (\text{S6})$$

$$= f_{\text{Solvated Monomer}} \left(1 + e^{i\vec{q} \cdot \vec{R}} + e^{i2\vec{q} \cdot \vec{R}} \right) + F_{\text{Solvation correction}} \left(1 + e^{i\vec{q} \cdot \vec{R}} \right) \quad (\text{S7})$$

Fig. S40 shows the resulting solvated trimer computed in Eq. S7.

Fig. S40. The solvated trimer computed by Eq. S7.

To apply the correction in a symmetry, the repeating subunit should already include the corrections at each one of the interfaces with its neighboring subunits. This principle is demonstrated in Fig. S41 and in Eq. S8:

$$f_{\text{Corrected Subunit}} = f_{\text{Solvated Monomer}} + \frac{1}{2} \times F_{\text{Centered Solvation correction}} \left(e^{\frac{-i\vec{q} \cdot \vec{R}}{2}} + e^{\frac{i\vec{q} \cdot \vec{R}}{2}} \right) \quad (\text{S8})$$

where $F_{\text{Centered Solvation correction}}$, is $F_{\text{Solvation correction}}$, which was placed at the origin:

$$F_{\text{Centered Solvation correction}} = F_{\text{Solvation correction}} \cdot e^{\frac{-i\vec{q} \cdot \vec{R}}{2}}$$

Fig. S41. Modified subunit to be used in a symmetry. The solvation correction amplitude is added to the repeating subunit with a factor of 0.5, at each end of the monomer, using Eq. S8). If the scale of the solvation layer is set to 1, then the amplitude scale of area 1 and area 3 is 0.5, whereas the scale of area 2 is -0.5.

Note that within a given assembly symmetry, the interfaces may vary in different subunit locations. In this case, the symmetry may be decomposed into smaller symmetries where each has the original subunit with a different solvation correction.

Going back to our example, the modified subunit is then inserted into its assembly symmetry. In the example in Fig. S42 we get:

$$f_{\text{Corrected Subunit in Symmetry}} = f_{\text{Corrected Subunit}} \left(1 + e^{i\vec{q}\cdot\vec{R}} + e^{i2\vec{q}\cdot\vec{R}} \right) \quad (\text{S9})$$

Fig. S42. Modified subunit in a symmetry. The cross product corresponds to convolution of the subunit with a sum of three delta functions, whose centers are where the dots are located. Inserting the corrected subunit from Fig. S41 to the trimer symmetry results in a trimer, with corrections at each one of the chain edges.

The edges, whose interfaces are uncoupled to neighboring subunits at one of the two ends, should be further corrected:

$$f_{\text{Solvated Dimer}} = f_{\text{Corrected Subunit in Symmetry}} - 0.5 \cdot F_{\text{Centered Solvation correction}} \left(e^{\frac{-i\vec{q}\cdot\vec{R}}{2}} + e^{\frac{i5\vec{q}\cdot\vec{R}}{2}} \right) \quad (\text{S10})$$

Fig. S43 demonstrates this final correction.

Fig. S43. Edge corrections. The structure in Fig. S42) is corrected to account for subunits located at the edge of the symmetry.

10.1. Solvated tubulin ring

We applied the approach described in the earlier section, to compute the scattering amplitude of the solvation layer of a tubulin ring, using the amplitude of a solvated tubulin dimer in horizontal orientation and the relevant correction terms. According to Eq. S5 a solvation correction amplitude was computed and added to the amplitude of the solvated dimer (Fig. S44):

$$f_{\text{Corrected Solvated Horizontal Dimer}} = f_{\text{Solvated Horizontal Dimer}} + F_{\text{solvation Horiznotal correction}} \quad (\text{S11})$$

Fig. S44. Solvation correction for a tubulin dimer in horizontal orientation (as in a tubulin ring).

To obtain the scattering amplitude of the solvated tubulin ring, the corrected solvated tubulin dimer amplitude was inserted into the ring symmetry (Fig. S45):

$$f_{\text{Corrected Solvation Ring}} = \sum_{i=0}^{12} f_{\text{Corrected Solvated Horizontal Dimer}} \left(\mathbf{A}_{\mathbf{n}}^{-1} \vec{q} \right) e^{i\vec{q} \cdot \vec{R}_n} \quad (\text{S12})$$

Fig. S45. Solvated tubulin dimer after solvation correction placed in the symmetry of tubulin ring.

The ring rotation matrices, A_n , are defined as:

$$\mathbf{A}_{\mathbf{n}} = \begin{pmatrix} \cos\left(\frac{360}{13}n\right) & -\sin\left(\frac{360}{13}n\right) & 0 \\ \sin\left(\frac{360}{13}n\right) & \cos\left(\frac{360}{13}n\right) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (\text{S13})$$

$$\mathbf{A}_{\mathbf{n}}^{-1} = \begin{pmatrix} \frac{\cos\left(\frac{360}{13}n\right)}{\cos^2\left(\frac{360}{13}n\right) + \sin^2\left(\frac{360}{13}n\right)} & \frac{\sin\left(\frac{360}{13}n\right)}{\cos^2\left(\frac{360}{13}n\right) + \sin^2\left(\frac{360}{13}n\right)} & 0 \\ -\frac{\sin\left(\frac{360}{13}n\right)}{\cos^2\left(\frac{360}{13}n\right) + \sin^2\left(\frac{360}{13}n\right)} & \frac{\cos\left(\frac{360}{13}n\right)}{\cos^2\left(\frac{360}{13}n\right) + \sin^2\left(\frac{360}{13}n\right)} & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (\text{S14})$$

The location vectors, R_n , are defined as:

$$\vec{R}_n = \begin{pmatrix} R \sin\left(\frac{360}{13}n\right) \\ R \cos\left(\frac{360}{13}n\right) \\ 0 \end{pmatrix} \quad (\text{S15})$$

where $n \in [0, 1, \dots, 12]$. As the ring structure is circular, no edge correction are required.

To test our approach, we compared the results of Eq. S12 with the intensity of solvated tubulin ring that was computed from a single PDB file, containing the atoms of the entire ring.

Fig. S46 shows that both methods give similar scattering intensity curves for solvated tubulin ring.

Fig. S46. Solution X-ray scattering intensity from an atomic model of solvated tubulin ring, shown as a ribbon diagram. H atoms were explicitly added to the PDB file (Chen *et al.*, 2010). The solid black curve was computed by D+, using the PDB file of the entire ring. The broken red curve was computed by D+ from the PDB file of the dimer (PDB ID 3J6F) with its solvation corrections and docking the result into the symmetry of the ring (see text). The solvation layers were 0.3 nm thick and their electron density contrasts (with respect to bulk water) were $30 \text{ e}/\text{nm}^3$. Both values are often used for computing the scattering amplitude contribution from solvation layers of proteins (Svergun *et al.*, 1995). The inset shows a small part of the curve on an expanded scale. The top curves show the normalized residual, R , where the reference curve is the intensity computed from the PDB file of the entire ring (Eq. S2).

Fig. S47 compares the contribution of the solvation layer using the two methods and

further confirms that both computations yield very similar results.

Fig. S47. The contribution of the solvation layers to the scattering curves in Fig. S46. Black curve is based on the PDB file of the entire ring. Broken red curve is based on the modified tubulin dimer (PDB ID 3J6F). The solvation layers were 0.3 nm thick and their electron density contrasts (with respect to bulk water) were 30 e/nm³. The inset shows a small part of the curve on an expanded scale. The top curves show the normalized residual, R , where the reference curve is the intensity computed by PDB file of the entire ring (Eq. S2).

10.2. Solvated microtubule

The approach that was used for computing the solvation layer of tubulin rings was adopted and adjusted for computing the scattering from solvated MT (Fig. 6). A few solvation overlap corrections were created and docked at the relevant locations with

the MT lattice. The MT lattice symmetry is described using the following rotation matrices and translation vectors:

$$\mathbf{A}_{\mathbf{n}}^{\text{MT}} = \begin{pmatrix} \cos\left(\frac{360}{N_{\text{PF Number}}}n\right) & -\sin\left(\frac{360}{N_{\text{PF Number}}}n\right) & 0 \\ \sin\left(\frac{360}{N_{\text{PF Number}}}n\right) & \cos\left(\frac{360}{N_{\text{PF Number}}}n\right) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (\text{S16})$$

$$\vec{R}_{j,n}^{\text{MT}} = \begin{pmatrix} R \sin\left(\frac{360}{N_{\text{PF Number}}}n\right) \\ R \cos\left(\frac{360}{N_{\text{PF Number}}}n\right) \\ H_{j-1} + P_n \end{pmatrix} \quad (\text{S17})$$

where $n \in [1, \dots, N_{\text{PF Number}}]$ and $H_j = K \cdot P \cdot j$ for $j \in [1, \dots, N_{\text{Long}}]$. N_{Long} is the number of longitudinal repeats of the subunit in the total helix. K is one over the number of helix starts, which is 3 for the monomers (assuming there is no difference between the α and β monomers) and $1\frac{1}{2}$ for the dimers. $P_n = \frac{P}{N_{\text{PF Number}}} \cdot n$, where $N_{\text{PF Number}}$ is the number of protofilaments in the MT (or the number of dimers per pitch), P is the length of the pitch, defined as the axial rise along the $\hat{z}$ direction per $N_{\text{PF Number}}$ dimers. $P = 0$ corresponds to a ring.

Unlike tubulin ring, MT is not a fully closed structure and it has a seam. Therefore, different corrections were added to dimers that were located at the center of the MT lattice, or at one of its edges, or along the MT seam (see marked dimers in Fig. S48).

The center of mass of the tubulin dimer was first shifted to the origin by computing

$$\rho_0(\vec{r}) = \rho(\vec{r} - \vec{R}_{\text{c.m.}}),$$

where $\rho(\vec{r})$ represents the electron density of the dimer (given by PDB ID 3J6F), and $\vec{R}_{\text{c.m.}}$ is its center of mass. Solvation correction for the dimers that are located side by side within the central part of the MT lattice, was constructed by placing the dimer in the MT symmetry for subunits whose indexes are $j = 1$ and $n \in [1, 2]$, giving

$$\rho_{1,1}^{\text{Dimer Side}}(\vec{r}) = \rho_0 \left[\left(\mathbf{A}_1^{\text{MT}} \right)^{-1} \left(\vec{r} - \vec{R}_{1,1}^{\text{MT}} \right) \right] \text{ and } \rho_{1,2}^{\text{Dimer Side}}(\vec{r}) = \rho_0 \left[\left(\mathbf{A}_2^{\text{MT}} \right)^{-1} \left(\vec{r} - \vec{R}_{1,2}^{\text{MT}} \right) \right],$$

respectively. The electron density of the pair of dimers, $\rho_{1,1 \& 1,2}^{\text{Dimer Pair Side}}(\vec{r}) = \rho_{1,1}^{\text{Dimer Side}}(\vec{r}) +$

$\rho_{1,2}^{\text{Dimer Side}}(\vec{r})$, was then aligned back to $\rho_0(\vec{r})$ (the initial dimer):

$$\begin{aligned} \rho_{1,1 \& 1,2}^{\text{Aligned Dimer Pair Side}}(\vec{r}) &= \rho_{1,1 \& 1,2}^{\text{Dimer Pair Side}} \left[\mathbf{A}_1^{\text{MT}} \vec{r} + \vec{R}_{1,1}^{\text{MT}} \right] = \\ &\rho_{1,1}^{\text{Dimer Side}} \left[\mathbf{A}_1^{\text{MT}} \vec{r} + \vec{R}_{1,1}^{\text{MT}} \right] + \rho_{1,2}^{\text{Dimer Side}} \left[\mathbf{A}_1^{\text{MT}} \vec{r} + \vec{R}_{1,1}^{\text{MT}} \right] \\ &\equiv \rho_{1,1}^{\text{Aligned Dimer Side}}(\vec{r}) + \rho_{1,2}^{\text{Aligned Dimer Side}}(\vec{r}) \end{aligned}$$

so the electron density (given as a PDB file) of the aligned $\rho_{1,1}^{\text{Dimer Side}}(\vec{r})$, which we shall refer to as $\rho_{1,1}^{\text{Aligned Dimer Side}}(\vec{r})$, overlapped with $\rho_0(\vec{r})$. Therefore the side (or lateral) solvation correction is:

$$\begin{aligned} f_{\text{Side Solvation Correction}} &= f_{1,1 \& 1,2}^{\text{Solvated Aligned Dimer Pair Side}} - \\ &\left(f_{1,1}^{\text{Solvated Aligned Dimer Side}} + f_{1,2}^{\text{Solvated Aligned Dimer Side}} \right) \end{aligned}$$

where f is the scattering amplitude of the corresponding solvated structures defined above.

To create the correction for dimers that are located on top of one another within the central part of the MT lattice, the same process was repeated for two dimers with indexes $j \in [1, 2]$ and $n = 1$, giving $\rho_{1,1}^{\text{Dimer Top}}(\vec{r}) = \rho_0 \left[\left(\mathbf{A}_1^{\text{MT}} \right)^{-1} \left(\vec{r} - \vec{R}_{1,1}^{\text{MT}} \right) \right]$ and $\rho_{2,1}^{\text{Dimer Top}}(\vec{r}) = \rho_0 \left[\left(\mathbf{A}_1^{\text{MT}} \right)^{-1} \left(\vec{r} - \vec{R}_{2,1}^{\text{MT}} \right) \right]$, respectively. The electron density (given as a PDB file) of the pair of dimers, $\rho_{1,1 \& 2,1}^{\text{Dimer Pair Top}}(\vec{r}) = \rho_{1,1}^{\text{Dimer Top}}(\vec{r}) + \rho_{2,1}^{\text{Dimer Top}}(\vec{r})$, was then aligned back to $\rho_0(\vec{r})$, the initial dimer: $\rho_{1,1 \& 2,1}^{\text{Aligned Dimer Pair Top}}(\vec{r}) = \rho_{1,1 \& 2,1}^{\text{Dimer Pair Top}} \left[\mathbf{A}_1^{\text{MT}} \vec{r} + \vec{R}_{1,1}^{\text{MT}} \right]$ so the electron density (given as a PDB file) of the aligned $\rho_{1,1}^{\text{Dimer Top}}(\vec{r})$, which we shall refer to as $\rho_{1,1}^{\text{Aligned Dimer Top}}(\vec{r})$, overlapped with $\rho_0(\vec{r})$. Similarly, $\rho_{2,1}^{\text{Aligned Dimer Top}}(\vec{r}) = \rho_{2,1}^{\text{Dimer Top}} \left[\mathbf{A}_1^{\text{MT}} \vec{r} + \vec{R}_{1,1}^{\text{MT}} \right]$. Therefore the top (or vertical) solvation correction is:

$$\begin{aligned} f_{\text{Top Solvation Correction}} &= f_{1,1 \& 2,1}^{\text{Solvated Aligned Dimer Pair Top}} - \\ &\left(f_{1,1}^{\text{Solvated Aligned Dimer Top}} + f_{2,1}^{\text{Solvated Aligned Dimer Top}} \right) \end{aligned}$$

where f is the scattering amplitude of the relevant solvated structures.

Two additional corrections were required for the MT seam, in which α - and β -tubulin are laterally located next to one another (instead of having α - next to α -tubulin and β - next to β -tubulin, in the central MT lattice). The above correction process was repeated for two monomers of the seam. Three dimers with indexes ($j = 1, n = N_{\text{PF Number}}$), ($j = 2, n = 1$) and ($j = 3, n = 1$), giving

$$\rho_{1, N_{\text{PF Number}}}^{\text{Dimer Seam}}(\vec{r}) = \rho_0 \left[\left(\mathbf{A}_{N_{\text{PF Number}}}^{\text{MT}} \right)^{-1} \left(\vec{r} - \vec{R}_{1, N_{\text{PF Number}}}^{\text{MT}} \right) \right],$$

$$\rho_{2,1}^{\text{Dimer Seam}}(\vec{r}) = \rho_0 \left[\left(\mathbf{A}_1^{\text{MT}} \right)^{-1} \left(\vec{r} - \vec{R}_{2,1}^{\text{MT}} \right) \right]$$

and $\rho_{3,1}^{\text{Dimer Seam}}(\vec{r}) = \rho_0 \left[\left(\mathbf{A}_1^{\text{MT}} \right)^{-1} \left(\vec{r} - \vec{R}_{3,1}^{\text{MT}} \right) \right]$, respectively. From dimers $\rho_{2,1}^{\text{Dimer Seam}}(\vec{r})$ and $\rho_{3,1}^{\text{Dimer Seam}}(\vec{r})$, two monomers were joined to form a new dimer (in reverse order) - β -tubulin of dimer ($j = 2, n = 1$), $\rho_{2,1,\beta}^{\text{Dimer Seam}}(\vec{r})$ and α -tubulin of dimer ($j = 3, n = 1$), $\rho_{3,1,\alpha}^{\text{Dimer Seam}}(\vec{r})$.

The newly formed dimer, $\rho_{2,1,\beta}^{\text{Dimer Seam}}(\vec{r})$ & $3,1,\alpha$ ($\vec{r}$) was combined with ($j = 1, n = N_{\text{PF Number}}$) to form the PDB file that describes the electron density of the pair of dimers,

$$\rho_{1, N_{\text{PF Number}} \& 2,1,\beta \& 3,1,\alpha}^{\text{Dimer Pair Seam}}(\vec{r}) = \rho_{1, N_{\text{PF Number}}}^{\text{Dimer Seam}}(\vec{r}) + \rho_{2,1,\beta \& 3,1,\alpha}^{\text{Dimer Seam}}(\vec{r}).$$

The pair was then aligned back to $\rho_0(\vec{r})$ (the initial dimer):

$$\rho_{1, N_{\text{PF Number}} \& 2,1,\beta \& 3,1,\alpha}^{\text{Aligned Dimer Pair Seam}}(\vec{r}) = \rho_{1, N_{\text{PF Number}} \& 2,1,\beta \& 3,1,\alpha}^{\text{Dimer Pair Seam}} \left[\mathbf{A}_{N_{\text{PF Number}}}^{\text{MT}} \vec{r} + \vec{R}_{1, N_{\text{PF Number}}}^{\text{MT}} \right]$$

so the electron density (given as a PDB file) of the aligned $\rho_{1, N_{\text{PF Number}}}^{\text{Dimer Seam}}(\vec{r})$, which we shall refer to as $\rho_{1, N_{\text{PF Number}}}^{\text{Aligned Dimer Seam}}(\vec{r})$, overlapped with $\rho_0(\vec{r})$. Similarly,

$$\rho_{2,1,\beta \& 3,1,\alpha}^{\text{Aligned Dimer Seam}}(\vec{r}) = \rho_{2,1,\beta \& 3,1,\alpha}^{\text{Dimer Seam}} \left[\mathbf{A}_{N_{\text{PF Number}}}^{\text{MT}} \vec{r} + \vec{R}_{1, N_{\text{PF Number}}}^{\text{MT}} \right].$$

From the electron density (given as a PDB file) of $\rho_{1, N_{\text{PF Number}} \& 2,1,\beta \& 3,1,\alpha}^{\text{Aligned Dimer Pair Seam}}(\vec{r})$, two electron densities (given by two PDB files) were extracted - the bottom α and β

monomers, $\rho_{1, N_{\text{PF Number}}, \alpha \& 2,1,\beta}^{\text{Aligned Bottom Monomer Pair Seam}}(\vec{r})$ and the top α and β monomers, $\rho_{1, N_{\text{PF Number}}, \beta \& 3,1,\alpha}^{\text{Aligned Top Monomer Pair Seam}}(\vec{r})$.

Each one of the new PDB files was further separated into their corresponding α - and the β - tubulin monomers, resulting in four PDB files describing the following electron densities: $\rho_{1,N_{\text{PF Number}},\alpha}^{\text{Aligned Bottom Monomer Seam}}(\vec{r})$, $\rho_{2,1,\beta}^{\text{Aligned Bottom Monomer Seam}}(\vec{r})$, $\rho_{1,N_{\text{PF Number}},\beta}^{\text{Aligned Top Monomer Seam}}(\vec{r})$, and $\rho_{3,1,\alpha}^{\text{Aligned Top Monomer Seam}}(\vec{r})$. Therefore the monomer seam solvation corrections are:

$$f_{\text{Solvation Correction Bottom Seam Monomer}} = f_{1,N_{\text{PF Number}},\alpha\&2,1\beta}^{\text{Solvated Aligned Bottom Monomer Pair Seam}} - \left(f_{1,N_{\text{PF Number}},\alpha}^{\text{Solvated Aligned Bottom Monomer Seam}} + f_{2,1,\beta}^{\text{Solvated Aligned Bottom Monomer Seam}} \right)$$

$$f_{\text{Solvation Correction Top Seam Monomer}} = f_{1,N_{\text{PF Number}},\beta\&3,1\alpha}^{\text{Solvated Aligned Top Monomer Pair Seam}} - \left(f_{1,N_{\text{PF Number}},\beta}^{\text{Solvated Aligned Top Monomer Seam}} + f_{3,1\alpha}^{\text{Solvated Aligned Top Monomer Seam}} \right)$$

where f is the scattering amplitude of the relevant solvated structures.

The entire solvated MT was computed by docking the amplitude of the solvated tubulin dimer, $f_{\text{Solvated Tubulin Dimer}}$, into the MT symmetry, with $n \in [1, \dots, N_{\text{PF Number}}]$ and $j \in [1, \dots, 48]$. Solvation correction amplitudes were then docked into the MT symmetry as follows. Side solvation correction amplitude, $f_{\text{Side Solvation Correction}}$, was docked into the MT symmetry with $n \in [1, \dots, N_{\text{PF Number}} - 1]$ and $j \in [1, \dots, 48]$. Top Solvation correction amplitude, $f_{\text{Top Solvation Correction}}$, was docked into the MT symmetry with $n \in [1, \dots, N_{\text{PF Number}}]$ and $j \in [1, \dots, 47]$. Bottom seam solvation correction amplitude, $f_{\text{Solvation Correction Bottom Seam Monomer}}$, was docked into the MT symmetry with $n = N_{\text{PF Number}}$ and $j \in [1, \dots, 47]$. Top seam solvation correction amplitude, $f_{\text{Solvation Correction Top Seam Monomer}}$, was docked into the MT symmetry with $n = N_{\text{PF Number}}$ and $j \in [1, \dots, 46]$.

The scattering amplitude of the solvated MT is then:

$$\begin{aligned}
 f_{\text{Solvated MT}} = & \sum_{n=1}^{N_{\text{PF Number}}} \sum_{j=1}^{48} f_{\text{Solvated Tubulin Dimer}} \left[\left(\mathbf{A}_{\mathbf{n}}^{\text{MT}} \right)^{-1} \vec{q} \right] e^{i\vec{q} \cdot \vec{R}_{j,n}^{\text{MT}}} + \\
 & \sum_{n=1}^{N_{\text{PF Number}}-1} \sum_{j=1}^{48} f_{\text{Side Solvation Correction}} \left[\left(\mathbf{A}_{\mathbf{n}}^{\text{MT}} \right)^{-1} \vec{q} \right] e^{i\vec{q} \cdot \vec{R}_{j,n}^{\text{MT}}} + \\
 & \sum_{n=1}^{N_{\text{PF Number}}} \sum_{j=1}^{47} f_{\text{Top Solvation Correction}} \left[\left(\mathbf{A}_{\mathbf{n}}^{\text{MT}} \right)^{-1} \vec{q} \right] e^{i\vec{q} \cdot \vec{R}_{j,n}^{\text{MT}}} + \\
 & \sum_{j=1}^{47} f_{\text{Solvation Correction Bottom Seam Monomer}} \left[\left(\mathbf{A}_{\mathbf{N}_{\text{PF Number}}}^{\text{MT}} \right)^{-1} \vec{q} \right] e^{i\vec{q} \cdot \vec{R}_{N_{\text{PF Number}},j}^{\text{MT}}} + \\
 & \sum_{j=1}^{46} f_{\text{Solvation Correction Top Seam Monomer}} \left[\left(\mathbf{A}_{\mathbf{N}_{\text{PF Number}}}^{\text{MT}} \right)^{-1} \vec{q} \right] e^{i\vec{q} \cdot \vec{R}_{N_{\text{PF Number}},j}^{\text{MT}}}
 \end{aligned}$$

Fig. S48. Ribbon diagram of a short MT, where α -tubulin is in green and β -tubulin is in red. The numbers in the diagram refer to dimers with a different type of solvation correction. Four solvation corrections types were applied: side (as in Fig. S39), top (as in Fig. S44), bottom and top monomers of the MT seam. 4 dimers are marked: (1) The first dimer in the minus end of the MT lattice, which is corrected for top and side solvation overlap, as applied for most other dimers in the MT lattice. (2) A dimer, which is part of the seam, which is corrected for the top solvation overlap and in addition for the side of the top and bottom monomers solvation overlap. (3) Three final monomers with $n = N_{\text{PF Number}}$ in the plus end of the MT lattice, which do not need side solvation correction. The two top monomers also do not require top solvation overlap correction. (4) The top dimer with $n = 1$, to which only side solvation correction is applied.

References

- Alushin, G. M., Lander, G. C., Kellogg, E. H., Zhang, R., Baker, D. & Nogales, E. (2014). *Cell*, **157**(5), 1117–1129.
- Ben-Nun, T., Ginsburg, A., Székely, P. & Raviv, U. (2010). *Journal of Applied Crystallography*, **43**(6), 1522–1531.
- Chen, V. B., Arendall, III, W. B., Headd, J. J., Keedy, D. A., Immormino, R. M., Kapral, G. J., Murray, L. W., Richardson, J. S. & Richardson, D. C. (2010). *Acta Crystallographica Section D*, **66**(1), 12–21.
- Debye, P. (1915). *Ann. Phys.* **351**(6), 809–823.

- Diaz, F. J., Pantos, E., Bordas, J. & Andreu, J. M. (1994). *Journal of molecular biology*, **238**(2), 214–225.
- Ginsburg, A., Ben-Nun, T., Asor, R., Shemesh, A., Ringel, I. & Raviv, U. (2016). *Journal of Chemical Information and Modeling*, **56**(8), 1518–1527.
- Grudin, S., Garkavenko, M. & Kazennov, A. (2017). *Acta Crystallographica Section D: Structural Biology*, **73**(5).
- Hamilton, W. (1974). *Birmingham: Kynoch Press. (Present distributor Kluwer Academic Publishers, Dordrecht. Table 2.2B.)*, pp. 273–284.
- Heckbert, P. (1994). *Graphics Gems IV (IBM Version)*. Elsevier.
- Jolliffe, I. (2002). *Principal component analysis*. Wiley Online Library.
- Marsh, R. & Slagle, K. (1983). *Acta Crystallographica Section A: Foundations of Crystallography*, **39**(1), 173–173.
- Nadler, M., Steiner, A., Dvir, T., Szekely, O., Szekely, P., Ginsburg, A., Asor, R., Resh, R., Tamburu, C., Peres, M. *et al.* (2011). *Soft Matter*, **7**(4), 1512–1523.
- Schrödinger, L. (2015). *The PyMOL Molecular Graphics System, Version 1.8*.
- Schrödinger, LLC (2015). The PyMOL molecular graphics system, version 1.8.
- Svergun, D., Barberato, C. & Koch, M. (1995). *Journal of Applied Crystallography*, **28**(6), 768–773.