Supplementary Information: Fast Approximations of the Rotational Diffusion Tensor and their Application to Structural Assembly of Molecular Complexes

Konstantin Berlin, Dianne P. O'Leary, David Fushman

January 4, 2011

S1 Covariance Matrix of an Ellipsoid

In this section we derive a formula for the covariance matrix of an ellipsoid and show the relationship between the ellipsoid and its principal semi-axes.

An ellipsoid $\mathcal{E}$ in $\mathbb{R}^3$ is defined as

$$\mathcal{E}(\mathbf{A}, \mathbf{c}) = \left\{ \mathbf{x} \mid (\mathbf{x} - \mathbf{c})^T \mathbf{A} (\mathbf{x} - \mathbf{c}) = 1 \right\},$$

where $\mathbf{A}$ is an 3×3 symmetric positive definite matrix that defines the shape of the ellipsoid, and $\mathbf{c} \in \mathbb{R}^3$ is its center.

The ellipsoid's principal semi-axes can be derived by an eigendecomposition of $\mathbf{A}$, such that

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T = \begin{bmatrix}\mathbf{V}_1\mathbf{V}_2\mathbf{V}_3\end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix}\mathbf{V}_1\mathbf{V}_2\mathbf{V}_3\end{bmatrix}^T, \tag{S1.1}$$

where lengths of the principal semi-axes are

$$\ell_1 = 1/\sqrt{\lambda_1}, \ell_2 = 1/\sqrt{\lambda_2}, \ell_3 = \sqrt{\lambda_3}, \tag{S1.2}$$

and $\mathbf{V}_1$, $\mathbf{V}_2$, $\mathbf{V}_3$ are their associated directions.

Since the covariance matrix is independent of the position of the ellipsoid we let $c = \mathbf{0}$. Changing into the coordinate space of $\mathbf{A}$, the ellipsoid can be rewritten using the Cartesian axes x, y, z:

$$\lambda_1 x^2 + \lambda_2 y^2 + \lambda_3 z^2 = 1. \tag{S1.3}$$

Due to the symmetry of the ellipsoid along the coordinate space axes, $\mathrm{Cov}(\mathcal{E}_i, \mathcal{E}_j) = 0$ when $i \neq j$. Therefore the covariance matrix of $\mathcal{E}$ is

$$\begin{bmatrix} \mathrm{Var}(\mathcal{E}_x) & 0 & 0 \\ 0 & \mathrm{Var}(\mathcal{E}_y) & 0 \\ 0 & 0 & \mathrm{Var}(\mathcal{E}_z) \end{bmatrix}, \tag{S1.4}$$

where $\mathrm{Var}(\mathcal{E}_x)$ is the variance along the x axis, $\mathrm{Var}(\mathcal{E}_y)$ is the variance along the y axis, and $\mathrm{Var}(\mathcal{E}_z)$ is the variance along the z axis.

The variance along the z axis is

$$\mathrm{Var}(\mathcal{E}_z) = \frac{\int_{\mathcal{E}} z^2 \, dx \, dy \, dz}{\int_{\mathcal{E}} \, dx \, dy \, dz}. \tag{S1.5}$$

Performing the change of variables into

$$\begin{aligned} x &= \frac{1}{\sqrt{\lambda_1}} \sin\phi \cos\theta, \\ y &= \frac{1}{\sqrt{\lambda_2}} \sin\phi \sin\theta, \\ z &= \frac{1}{\sqrt{\lambda_3}} \cos\phi, \end{aligned} \tag{S1.6}$$

where θ is the azimuthal angle, ϕ is the polar angle, and the Jacobian determinant is

$$|\mathbf{J}| = (\lambda_1\lambda_2\lambda_3)^{-1/2} \sin\phi, \tag{S1.7}$$

we get

$$\begin{aligned} \mathrm{Var}(\mathcal{E}_z) &= \frac{\int_0^{2\pi}\int_0^{\pi} (\lambda_3^{-1/2}\cos\phi)^2 \, |\mathbf{J}| \, d\phi \, d\theta}{\int_0^{2\pi}\int_0^{\pi} |J| \, d\phi \, d\theta} \\ &= \frac{4/3\lambda_3^{-1}\pi(\lambda_1\lambda_2\lambda_3)^{-1/2}}{4\pi(\lambda_1\lambda_2\lambda_3)^{-1/2}} \\ &= \frac{1}{3\lambda_3}. \end{aligned} \tag{S1.8}$$

Similarly, $\mathrm{Var}(\mathcal{E}_x) = 1/(3\lambda_1)$, and $Var(\mathcal{E}_y) = 1/(3\lambda_2)$.

Changing back to the original coordinate system, the covariance matrix of $\mathcal{E}$ is:

$$\mathbf{C}_{\mathcal{E}} = \mathbf{V} \begin{bmatrix} 1/(3\lambda_1) & 0 & 0 \\ 0 & 1/(3\lambda_2) & 0 \\ 0 & 0 & 1/(3\lambda_3) \end{bmatrix} \mathbf{V}^T. \tag{S1.9}$$

S2 Perrin's Equations

For completeness, here we reproduce Perrin's equations for computing the diffusion tensor from an equivalent ellipsoid, as shown in Perrin [1, 2].

Given the lengths of the equivalent ellipsoid's principal semi-axes, ℓ_1, ℓ_2, and ℓ_3, and the ellipsoid's orientation matrix $\mathbf{V}$, the predicted diffusion tensor of the ellipsoid is:

$$\mathbf{D}_{pred} = \mathbf{V} \begin{bmatrix} D_1 & 0 & 0 \\ 0 & D_2 & 0 \\ 0 & 0 & D_3 \end{bmatrix} \mathbf{V}^T, \tag{S2.1}$$

where

$$\begin{aligned} D_1(\ell_1, \ell_2, \ell_3) &= \frac{k_b T}{I_1}, \\ D_2(\ell_1, \ell_2, \ell_3) &= \frac{k_b T}{I_2}, \\ D_3(\ell_1, \ell_2, \ell_3) &= \frac{k_b T}{I_3}, \end{aligned} \tag{S2.2}$$

T is the temperature (° Kelvin), k_b is the Boltzmann constant, the principal components of the inertia tensor are

$$\begin{aligned} I_1 &= \frac{16\pi\eta(\ell_2^2+\ell_3^2)}{\ell_2^2 Q_2 + \ell_3^2 Q_3}, \\ I_2 &= \frac{16\pi\eta(\ell_1^2+\ell_3^2)}{\ell_1^2 Q_1 + \ell_3^2 Q_3}, \\ I_3 &= \frac{16\pi\eta(\ell_1^2+\ell_2^2)}{\ell_1^2 Q_1 + \ell_2^2 Q_2}, \end{aligned} \tag{S2.3}$$

η is the solvent viscosity, and

$$\begin{aligned} Q_1 &= \int_0^\infty \frac{ds}{\sqrt{(\ell_1^2+s)^3(\ell_2^2+s)(\ell_3^2+s)}}, \\ Q_2 &= \int_0^\infty \frac{ds}{\sqrt{(\ell_2^2+s)^3(\ell_1^2+s)(\ell_3^2+s)}}, \\ Q_3 &= \int_0^\infty \frac{ds}{\sqrt{(\ell_3^2+s)^3(\ell_1^2+s)(\ell_2^2+s)}}. \end{aligned} \tag{S2.4}$$

Thus, given a molecule the steps to predicting its diffusion tensor are: Compute the molecule's PCAE; compute the eigendecomposition of the PCAE; find the lengths of PCAE's axes using equation (S1.2); and finally, predict the diffusion tensor using Perrin's equations.

S3 Inverting the Perrin's Equations

In order to accurately invert Perrin's equations we need an estimate of how many ellipsoids we expect to map into the same diffusion tensor. To do this, we mapped the lengths of ellipsoid's principal semi-axes $[\ell_1, \ell_2, \ell_3]$, where $0 < \ell_1 \leq \ell_2 \leq \ell_3$, sampled at 2Å intervals, into the diffusion tensor space using Perrin's equations. To better spread out the points we adjusted each eigenvalue of the diffusion tensor using the function ξ, where

$$\xi(D_i) = \log(\log(\log(\log(\log(D_i+1)+1)+1)+1)+1), \tag{S3.1}$$

for $i = x, y, z$.

We split the cube of the diffusion tensor space (from 0 to $\max(T)$) into $20 \times 20 \times 20$ cubes, and for each cube observed which triples of $[\ell_1, \ell_2, \ell_3]$ are mapped into that cube. To quantify the number of distinct regions that map into each cube we performed hierarchical clustering on the triples based on their Euclidean distances and recorded the number of clusters. The distance between two clusters was measured as the Euclidean distance between the two closest points of the clusters and the clustering cutoff at 3.7Å, a value that is smaller than 4Å, the shortest possible distance between two non-adjacent sample points. The maximum number of clusters in any cube was two, and the majority of the occupied cubes contain only one cluster. Therefore, we expect at most two distinct triplets of $[\ell_1, \ell_2, \ell_3]$ to have the same diffusion tensor. Since there are only two solutions, we can try to find both of the solutions by simply trying eight different starting points, $[a, a, a]$, $[a, a, b]$, $[a, b, a]$, $[a, b, b]$, $[b, a, a]$, $[b, a, b]$, $[b, b, a]$, and $[b, b, b]$ in the nonlinear least squares algorithm, where $[a, b]$ is the expected range of the axes' lengths. In practice, we are able to eliminate all but one of the solutions by simply picking the solution with the lowest residual.

S4 Quadratic Approximation of a Molecule's Covariance Matrix

In this section we derive the quadratic approximation $\mathbf{Q}$ of the function $\mathbf{G}$ around a point $\mathbf{x}$. The approximation will allow us to quickly approximate the descent step for our minimization of χ^2_G.

Let $\mathbf{a}^1, \ldots, \mathbf{a}^{n_a}$ be the surface points for $M(\mathbf{x})$ that come from domain A and let $\mathbf{b}^1, \ldots, \mathbf{b}^{n_b}$ be the surface points for $M(\mathbf{x})$ that come from domain B. We do not expect the set of surface points to change significantly as the position of B is perturbed by $\mathbf{p}$. The majority of the change in the covariance matrix will come from the fact that $\mathbf{b}^i$ points are shifted and not from the actual change in the surface points. The larger $\|\mathbf{p}\|$ is, the more we expect the set of the surface points to change, but at the same time the translation of points that remain on the surface also contributes a greater weight. Thus, we expect that we can estimate the covariance matrix well at $\mathbf{x}+\mathbf{p}$ by simply adjusting the points $\mathbf{b}$ by $\mathbf{p}$ and recomputing the covariance matrix.

We now write out the equation for approximating $G_{ij}(\mathbf{x}+\mathbf{p})$ by simply computing the covariance matrix of set $\mathbf{a}$ and the adjusted set $\mathbf{b}$:

$$
\begin{aligned}
G_{ij}(\mathbf{x}+\mathbf{p}) \approx & \frac{\sum_{v=1}^{n_a} a_i^v a_j^v + \sum_{v=1}^{n_b} (b_i^v + p_i)(b_j^v + p_j)}{n_a + n_b} \\
& - \frac{\left[\sum_{v=1}^{n_a} a_i^v + \sum_{v=1}^{n_b} (b_i^v + p_i)\right] \left[\sum_{v=1}^{n_a} a_j^v + \sum_{v=1}^{n_b} (b_j^v + p_j)\right]}{(n_a + n_b)^2} \\
= & \frac{\sum_{v=1}^{n_a} a_i^v a_j^v + \sum_{v=1}^{n_b} b_i^v b_j^v}{n_a + n_b} \\
& - \frac{\left[\sum_{v=1}^{n_a} a_i^v + \sum_{v=1}^{n_b} b_i^v\right] \left[\sum_{v=1}^{n_a} a_j^v + \sum_{v=1}^{n_b} b_j^v\right]}{(n_a + n_b)^2} \\
& + \frac{\sum_{v=1}^{n_b} (b_i^v p_j + b_j^v p_i + p_i p_j)(n_a + n_b)}{(n_a + n_b)^2} \\
& - \frac{\left(\sum_{v=1}^{n_a} a_i^v + \sum_{v=1}^{n_b} b_i^v\right) n_b p_j + \left(\sum_{v=1}^{n_a} a_j^v + \sum_{v=1}^{n_b} b_j^v\right) n_b p_i + n_b^2 p_i p_j}{(n_a + n_b)^2} \\
= & G_{ij}(\mathbf{x}) + \frac{(n_a + n_b) \sum_{v=1}^{n_b} b_i^v p_j + (n_a + n_b) \sum_{v=1}^{n_b} b_j^v p_i + (n_a + n_b) n_b p_i p_j}{(n_a + n_b)^2} \\
& - \frac{n_b p_j \sum_{v=1}^{n_a} a_i^v + n_b p_j \sum_{v=1}^{n_b} b_i^v + n_b p_i \sum_{v=1}^{n_a} a_j^v + n_b p_i \sum_{v=1}^{n_b} b_j^v + n_b^2 p_i p_j}{(n_a + n_b)^2} \\
= & G_{ij}(\mathbf{x}) + Q_{ij}(\mathbf{p}),
\end{aligned}
\tag{S4.1}
$$

where

$$
Q_{ij}(\mathbf{p}) = \kappa p_i p_j + K_{ij} p_i + K_{ji} p_j, \tag{S4.2}
$$

and

$$
\kappa = \frac{n_a n_b}{(n_a + n_b)^2}, \tag{S4.3}
$$

$$
K_{ij} = \frac{(n_a + n_b) \sum_{v=1}^{n_b} b_j^v - n_b (\sum_{k=1}^{n_a} a_j^v + \sum_{v=1}^{n_b} b_j^v)}{(n_a + n_b)^2}, \tag{S4.4}
$$

for $i, j = 1, 2, 3$.

Observe that if the two sets of points do not change during the translation $\mathbf{p}$ (i.e. the two domains never collide, either before or after) our approximation yields an exact value, and that the analytical formula for the Jacobian of $\mathbf{Q}$ is trivially computed.

S5 Docking Algorithm ELMDOCK

Here we present a detailed description of our Newton-like minimization algorithm for computation of the optimal domain positioning based on the rotational diffusion tensor.

Algorithm S5.1 Docking Algorithm ELMDOCK

Input: Three-dimensional structure of domain A and B, ρ^{exp} – experimental relaxation-rates ratios, $\mathbf{G}(\mathbf{x})$ – a function that computes the covariance matrix of $M(\mathbf{x})$.

Output: $\mathbf{x}^*$ – the translation of B that yields the best docking solution as measured by our energy function.

1: Orient the A and B domains using ρ^{exp} {See section "Orienting Domains using Relaxation Data" in the main text}
2: Compute $\mathbf{D}_{exp}$ using ρ^{exp} from both of the domains using ROTDIF
3: Compute the covariance matrix $\mathbf{C}^*$ from $\mathbf{D}_{exp}$ {See section "Step 1: From Diffusion Tensor to Covariance Matrix" in the main text}
4: $\mathbf{x}^* \leftarrow \infty$
5: **for** every initial guess $\mathbf{x}_0$ {See section "Step 2: From Equivalent Ellipsoid to Domain Position" in the main text} **do**
6: $\quad k \leftarrow 0$
7: $\quad \mathbf{x}_k \leftarrow \mathbf{x}_0$
8: $\quad$ **while** stopping condition not reached {See Section S5.2} **do**
9: $\quad\quad$ Compute a descent direction $\mathbf{p} \in \mathbb{R}^3$ for $\chi^2_G(\mathbf{x}_k)$ {See Section S5.1}
10: $\quad\quad$ Set $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \mathbf{p}$
11: $\quad\quad$ Set $k \leftarrow k + 1$
12: $\quad$ **end while**
13: $\quad$ **if** $\chi^2_G(\mathbf{x}_k) < \chi^2_G(\mathbf{x}^*)$ **then**
14: $\quad\quad \mathbf{x}^* \leftarrow \mathbf{x}_k$
15: $\quad$ **end if**
16: **end for**
17: **return** $\mathbf{x}^*$

The following subsections expend upon steps described in the algorithm.

S5.1 Approximating the Descent Step

Recall that the most important step in a convex minimization is finding a descent step.

At each step k, we would like to find the value for $\mathbf{p}$ such that $\mathbf{x}_k + \mathbf{p}$ minimizes χ^2_G:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \arg\min_{\mathbf{p}} \chi^2_G(\mathbf{x}_k + \mathbf{p}). \tag{S5.1}$$

However, finding the true minimizer of χ^2_G directly is computationally expensive. We therefore

approximate $\chi^2_G(\mathbf{x}_k + \mathbf{p})$ by using our quadratic function approximation derived in Section S4:

$$\chi^2_G(\mathbf{x}_k + \mathbf{p}) \approx \tilde{\chi}^2_G(\mathbf{p}) = \sum_{i=1}^{3}\sum_{j=1}^{3}(G_{ij}(\mathbf{x}_k) + Q_{ij}(\mathbf{p}) - C^*_{ij})^2. \tag{S5.2}$$

The Jacobian of $\mathbf{Q}$ can be trivially computed, and we can very quickly solve for the value of $\mathbf{p}$ that minimizes $\tilde{\chi}_G$.

Therefore, the equation for our next step in each iteration becomes

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \arg\min_p \tilde{\chi}^2_G(\mathbf{p}). \tag{S5.3}$$

We now iteratively converge to the true minimizer of χ^2_G.

To evaluate equation (S5.3) we need to evaluate $\mathbf{G}(\mathbf{x}_k)$. We speedup the minimization by using $\mathbf{G}^{fast}(\mathbf{x}_k)$ (see equation (17) in the main text) instead of $\mathbf{G}(\mathbf{x}_k)$ for all our iterations.

S5.2 Stopping Conditions

There are three conditions which terminate our algorithm. The first case is when we are close enough to the solution:

$$\|\mathbf{G}^{fast}(\mathbf{x}_k) - \mathbf{C}^*\|^2_F < \epsilon_1. \tag{S5.4}$$

The second case is when we are not making enough progress:

$$\|\mathbf{G}^{fast}(\mathbf{x}_k) - \mathbf{G}^{fast}(\mathbf{x}_{k-1})\|^2_F < \epsilon_2. \tag{S5.5}$$

And the last case is when the step size is small enough:

$$\|\mathbf{x}_k - \mathbf{x}_{k-1}\|^2_F < \epsilon_3. \tag{S5.6}$$

S6 Largest Outlier 1I4D

Figure S6.1 shows an illustration of the approximate solvent-accessible surface for the main outlier of our docking method, complex 1I4D, as approximated by $\mathbf{G}^{fast}$ method. Note the relatively large surface area of the big domain relative to the small domain. The overall molecular weight of the complex is 73 kDa.

References

[1] Perrin, F. (1934) Mouvement Brownien d'un ellipsoide (I). Dispersion dielectrique pour des molecules ellipsoidales. *Le Journal de Physique 5*, 497–511.

[2] Perrin, F. (1936) Mouvement Brownien d'un ellipsoide (II). Rotation libre et depolarisation des fluorescences. Translation et diffusion de molecules ellipsoidales. *Le Journal de Physique 7*, 1–11.

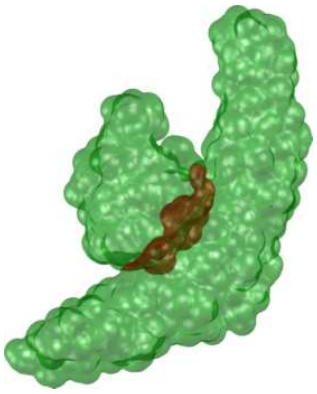

Figure S6.1: Illustration of the solvent-accessible surface of complex 42, 1I4D, as approximated by the $\mathbf{G}^{fast}$ method. The interface surface (colored red) does not contribute to the computation of the covariance matrix.