Efficient generation of random rotation matrices in four dimensions

Jakob Tómas Bullerjahn,[1,*,†] Balázs Fábián,[1,*,‡] and Gerhard Hummer[1,2]
[1]*Department of Theoretical Biophysics, Max Planck Institute of Biophysics, 60438 Frankfurt am Main, Germany*
[2]*Institute of Biophysics, Goethe University Frankfurt, 60438 Frankfurt am Main, Germany*

(Received 23 May 2023; accepted 24 July 2023; published 19 September 2023)

Four-dimensional (4D) rotations have applications in the fields of robotics, computer vision, and rigid-body mechanics. In the latter, they can be used to transform between equimomental systems of point masses. Here we provide an efficient algorithm to generate random 4D rotation matrices covering an arbitrary, predefined range of rotation angles. These matrices can be combined with Monte Carlo methods for the efficient sampling of the SO(4) group of 4D rotations. The matrices are unbiased and constructed such that repeated rotations result in uniform sampling over SO(4). The algorithm can be used to optimize the mass partitioning in coarse-grained simulation models of molecules involving coupled constraints for stable time integration.

DOI: 10.1103/PhysRevE.108.035307

I. INTRODUCTION

Four-dimensional (4D) rotations appear in a wide range of problems, from theoretical physics to robotics. They define the symmetries of two-body motions with inverse-square forces, such as the Kepler problem and the hydrogen atom [1]. Rotations in 4D have been used to model electromagnetic wave propagation for coherent optical communication [2], in algorithms for signal processing [3], and to construct equimomental systems of point masses for rigid bodies [4], i.e., systems of mass points with preserved total mass, center of mass, and inertia tensor. They can also be used to formulate various problems in robotics and computer vision, such as hand-eye calibration problems [5] and point-set registration [6]. The group SO(4) of 4D rotations can be represented in terms of 4×4 orthogonal matrices with determinant 1. A rotation in 4D can be regarded as two distinct rotations in a pair of orthogonal planes [7] or, equivalently, two orthogonal three-dimensional (3D) rotations, because a rotation angle and a normal vector to a plane form an element of SO(3). This is reflected by the fact that SO(4) is a double cover of $\mathrm{SO}(3) \times \mathrm{SO}(3)$ [7]. Most 4D rotations are so-called double rotations, where the two rotation angles, α and β, have distinct values, but isoclinic ($\alpha = \beta$) and simple rotations (either $\alpha = 0$ or $\beta = 0$) can also occur.

There are multiple ways to generate random 4D rotation matrices. For instance, the Cayley transform

$$\mathbf{R}_4 = (\mathbf{I}_4 - \mathbf{S}_4)(\mathbf{I}_4 + \mathbf{S}_4)^{-1}$$

turns any skew-symmetric 4×4 matrix $\mathbf{S}_4 = -\mathbf{S}_4^\top$ into an element of SO(4), where $\top$ indicates the transpose and $\mathbf{I}_4$ is the identity matrix. However, in order to generate uniformly distributed rotation matrices, the matrix elements of $\mathbf{S}_4$ have to be distributed according to a generalization of the multivariate t distribution [8]. Alternatively, one can make use of the QR decomposition [9] or Householder transformations [10] to generate uniformly distributed elements of $\mathrm{SO}(n)$ for $n \geqslant 2$.

While the above-mentioned methods can be used to efficiently generate uniformly distributed SO(4) elements, their main drawback is the fact that they do not give practitioners full control over the matrix-generation process. In particular, none of these methods can be used to exclusively produce small-angle rotations in random orthogonal two-dimensional subspaces. Instead, numerically expensive decomposition schemes have to be used to explicitly check the size of the rotation angles and then either accept or reject the proposed matrix. Small-angle matrices are a prerequisite to perform random walks in 4D rotation space with small steps for efficient Monte Carlo sampling, which are, e.g., needed to search for optimal equimomental systems of certain molecular structures to stabilize the numerics in coarse-grained molecular dynamics simulations [11]. Possible starting points could be quaternions, i.e., a 4-tuple describing 3D rotations in terms of an angle and an axis of rotation, because the product of two distinct quaternions results in a 4D rotation matrix [7,12]. Similarly, it might be tempting to consider Cayley factorization of 4D rotations into a pair of left- and right-isoclinic 4D rotation matrices [13,14]. However, for the resulting rotations to be small, said pairs of quaternions or rotation matrices cannot be chosen arbitrarily, but have to satisfy additional constraints.

Here, we introduce a geometrically appealing way of constructing rotation matrices for simple, double, and isoclinic rotations in 4D with arbitrary rotation angles. The method exploits the fact that any 4D rotation matrix can be written as the matrix exponential of a skew-symmetric matrix, which can then be decomposed into a sum of two orthogonal skew-symmetric matrices, $\mathbf{A}$ and $\mathbf{B}$, weighted by the rotation angles α and β [15].

*These authors contributed equally to this work.
†jakob.bullerjahn@biophys.mpg.de
‡balazs.fabian@biophys.mpg.de

The paper is structured as follows. In Sec. II, the above-mentioned decomposition scheme is briefly reviewed in 3D and 4D, after which we proceed to solve the inverse problem of generating the matrices $\mathbf{A}$ and $\mathbf{B}$. We also explore the nontrivial distributions of 4D rotation angles and show that α and β can essentially be sampled from arbitrary distributions in the small-angle limit. Section III presents an efficient algorithm (in the sense that it requires only a minimal number of pseudorandom numbers) to construct random, small-angle, 4×4 rotation matrices suitable to be used in Markov-chain Monte Carlo sampling schemes. The algorithm is verified in Sec. IV via numerical tests and Sec. V discusses the use of 4D rotation matrices to transform between equimomental systems. A summary of our results can be found in Sec. VI, followed by Appendixes that contain various technical details and derivations.

II. THEORY

In principle, it is possible to construct a random small-angle 4D rotation matrix $\mathbf{R}_4$ by combining a double rotation matrix

$$\mathbf{R}_4'(\alpha,\beta) = \begin{pmatrix} \cos(\alpha) & \sin(\alpha) & 0 & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & \cos(\beta) & \sin(\beta) \\ 0 & 0 & -\sin(\beta) & \cos(\beta) \end{pmatrix},$$

with an arbitrary element $\mathbf{Q}_4$ of the orthonormal group O(4) as follows:

$$\mathbf{R}_4(\alpha,\beta \ll 1) = \mathbf{Q}_4\mathbf{R}_4'(\alpha,\beta)\mathbf{Q}_4^\top. \tag{1}$$

While the planes of rotation are well defined for $\mathbf{R}_4'$ (the matrix describes rotations in the xy and zw planes), the matrix product in Eq. (1) ensures that the planes of rotation for $\mathbf{R}_4$ are random.

The downside of Eq. (1) is its computational inefficiency. In a costly process, $\mathbf{Q}_4$ could be generated by Gram-Schmidt orthogonalization of four 4D vectors. More efficiently, software packages like SCIPY [16] make use of an algorithm, which requires nine pseudorandom numbers [10] to generate 4D rotation matrices, and Eq. (1) requires two additional degrees of freedom for the rotation angles. However, the dimension of SO(4) is 6 [7], so there is definitely room for further improvement, noting that producing quality random numbers is computationally costly.

In the remainder of this section, we discuss and develop the theoretical premise for an efficient algorithm to generate small-angle 4D rotations. In Sec. IV A, we show that our algorithm outcompetes Eq. (1) in runtime performance, which makes it more viable for applications involving Monte Carlo methods.

A. Matrix decomposition for skew symmetric matrices

A 3×3 matrix $\mathbf{R}_3$, representing 3D rotations about the unit vector $\vec{u} = (-s_{23}, s_{13}, -s_{12})^\top$, $|\vec{u}| = 1$, can be expressed in terms of the exponential of a skew-symmetric matrix

$$\mathbf{S}_3 = \begin{pmatrix} 0 & s_{12} & s_{13} \\ -s_{12} & 0 & s_{23} \\ -s_{13} & -s_{23} & 0 \end{pmatrix}$$

via the Rodrigues rotation formula

$$\mathbf{R}_3(\gamma) = e^{\gamma \mathbf{S}_3} = \mathbf{I}_3 + \sin(\gamma)\mathbf{S}_3 + [1-\cos(\gamma)]\mathbf{S}_3^2.$$

Here, γ denotes the angle of rotation and $\mathbf{I}_3$ is the identity matrix. Due to the Cayley-Hamilton theorem and the constraint on the norm of the vector $\vec{u}$, the matrix $\mathbf{S}_3$ satisfies the property $\mathbf{S}_3^3 = -\mathbf{S}_3$.

The eigenvalues of skew-symmetric matrices can either be complex conjugate pairs or zero; hence their rank is always even. Therefore, in 3D, the infinitesimal generator $\mathbf{S}_3$ of a rotation has rank 2 and is singular, i.e., $\det(\mathbf{S}_3) = 0$. Let $\lambda_{1,2} = \pm i\theta$ and $\lambda_3 = 0$ be the eigenvalues of $\mathbf{S}_3$ with corresponding eigenvectors $\vec{s}_1, \vec{s}_2 \in \mathbb{C}^3$ and $\vec{s}_3 \in \mathbb{R}^3$, where $\vec{s}_1$ and $\vec{s}_2$ form a complex conjugate vector pair. The matrix $\mathbf{R}_3$ then possesses the same eigenvectors as $\mathbf{S}_3$ with eigenvalues e^{λ_n} for $n = 1, 2, 3$. The rank deficiency of $\mathbf{S}_3$ expresses the fact that rotations in 3D involve a plane and a unique orthogonal direction along which no rotation takes place, namely the axis of rotation. The nonzero elements of $\mathbf{S}_3$ represent rotations in the corresponding planes, e.g., the element s_{12} produces rotation in the xy plane. The plane of rotation can be readily expressed as the span of

$$\vec{v}_1 = \frac{1}{\sqrt{2}}\,\mathrm{Re}(\vec{s}_1 + \vec{s}_2), \quad \vec{v}_2 = \frac{1}{\sqrt{2}}\,\mathrm{Im}(\vec{s}_1 - \vec{s}_2),$$

where $\mathrm{Re}(\cdot)$ and $\mathrm{Im}(\cdot)$ denote the real and imaginary parts of their arguments, respectively. Setting $\vec{v}_3 = \vec{s}_3$ gives an orthonormal set of vectors in $\mathbb{R}^3$ that satisfy $\vec{v}_i \cdot \vec{v}_j = \delta_{ij}$ for $i, j = 1, 2, 3$.

These connections between matrix representations and the geometric meaning of both $\mathbf{S}_3$ and $\mathbf{R}_3$ also translate to 4D rotations, where any rotation matrix satisfies $\mathbf{R}_4 = e^{\mathbf{S}_4}$ for a skew-symmetric matrix

$$\mathbf{S}_4 = \begin{pmatrix} 0 & s_{12} & s_{13} & s_{14} \\ -s_{12} & 0 & s_{23} & s_{24} \\ -s_{13} & -s_{23} & 0 & s_{34} \\ -s_{14} & -s_{24} & -s_{34} & 0 \end{pmatrix}.$$

The matrix $\mathbf{S}_4$ can be uniquely decomposed as [15]

$$\mathbf{S}_4 = \alpha\mathbf{A} + \beta\mathbf{B} \tag{2}$$

for $\alpha, \beta \geqslant 0$, $\alpha \neq \beta$, with two rank-2 skew symmetric matrices $\mathbf{A}$ and $\mathbf{B}$, which satisfy

$$\mathbf{A}^3 = -\mathbf{A}, \quad \mathbf{B}^3 = -\mathbf{B}, \quad \mathbf{AB} = \mathbf{BA} = 0. \tag{3}$$

For simple rotations, where one of the angles α, β vanishes, $\mathbf{S}_4$ also has rank 2, but in general it has full rank. The corresponding Rodrigues rotation formula reads [15]

$$\mathbf{R}_4(\alpha,\beta) = e^{\mathbf{S}_4} = \mathbf{I}_4 + \sin(\alpha)\mathbf{A} + [1-\cos(\alpha)]\mathbf{A}^2 + \sin(\beta)\mathbf{B} + [1-\cos(\beta)]\mathbf{B}^2. \tag{4}$$

Reference [15] introduces an algorithm to calculate the angles α and β, as well as the matrices $\mathbf{A}$ and $\mathbf{B}$, from a given skew-symmetric matrix $\mathbf{S}_4$. In what follows, we consider the inverse problem of constructing a pair of matrices $\mathbf{A}$ and $\mathbf{B}$ that satisfy Eq. (3), which then allows us to use Eq. (4) to generate random 4D rotation matrices for small-angle rotations. Our results can be used to efficiently perform a random walk in 4D rotation

space, as is required, e.g., for the generation of equimomental systems of point masses [4,11].

B. Constructing the matrix A

Motivated by the fact that a 4D rotation can be represented by a pair of 3D rotations, we consider two three-dimensional vectors, $\vec{a}_1$ and $\vec{a}_2$, which should satisfy the relation

$$(\vec{a}_1 \cdot \vec{a}_2)^2 = \det(\mathbf{A})$$

for the skew-symmetric matrix

$$\mathbf{A} = \begin{pmatrix} 0 & a_{12} & a_{13} & a_{14} \\ -a_{12} & 0 & a_{23} & a_{24} \\ -a_{13} & -a_{23} & 0 & a_{34} \\ -a_{14} & -a_{24} & -a_{34} & 0 \end{pmatrix}.$$

There are multiple but equivalent ways of associating the elements of $\vec{a}_1$ and $\vec{a}_2$ with the matrix elements a_{ij}. Here, we choose $\vec{a}_1$ analogous to the vector $\vec{u}$ in the three-dimensional case, which uniquely determines $\vec{a}_2$ and we get

$$\vec{a}_1 = \begin{pmatrix} -a_{23} \\ a_{13} \\ -a_{12} \end{pmatrix}, \quad \vec{a}_2 = \begin{pmatrix} a_{14} \\ a_{24} \\ a_{34} \end{pmatrix}. \tag{5}$$

Analogous to the 3D case, a_{ij} represents rotation in the ij plane and the dot product of $\vec{a}_1$ and $\vec{a}_2$ combines elements representing orthogonal planes. Furthermore, one can show that $\mathbf{A}^3 = -\mathbf{A}$ holds whenever $\vec{a}_1$ and $\vec{a}_2$ satisfy

$$\vec{a}_1 \cdot \vec{a}_2 = 0, \quad \vec{a}_1 \cdot \vec{a}_1 + \vec{a}_2 \cdot \vec{a}_2 = 1. \tag{6}$$

Requiring $\vec{a}_1$ and $\vec{a}_2$ to be perpendicular ensures that $\mathbf{A}$ is a rank-2 matrix, i.e., a simple rotation involving a single plane of rotation. The second requirement of $\vec{a}_1$ and $\vec{a}_2$ forming a six-dimensional unit vector $(\vec{a}_1^\top, \vec{a}_2^\top)^\top$ ensures the decomposition of $\mathbf{S}_4$ according to Eq. (2), as we shall see in Sec. II C.

Naively, one could think that the construction of $\mathbf{A}$ requires six numbers, i.e., one for every element above the diagonal, but it turns out that only four are needed. As discussed in Appendix A, this is because we can always choose an arbitrary, auxiliary unit vector $\vec{a}_1^*$ from the elements of the 2-sphere, which are fully determined by the polar and azimuthal angles. This accounts for two degrees of freedom. We can then identify two vectors perpendicular to $\vec{a}_1^*$ that span a plane, in which a second auxiliary unit vector $\vec{a}_2^*$ is then constructed with an additional degree of freedom. Finally, a fourth degree of freedom R serves as the "mixing ratio" between $\vec{a}_1^*$ and $\vec{a}_2^*$, which results in two vectors $\vec{a}_1 = \vec{a}_1^*\sqrt{R}$ and $\vec{a}_2 = \vec{a}_2^*\sqrt{1-R}$ that satisfy the constraints in Eq. (6) and define a matrix $\mathbf{A}$ according to Eq. (5).

C. Constructing the matrix B from a known matrix A

Generating a 4D rotation requires, in general, six pieces of information [7]. We have already shown that four numbers are needed to construct the rank-2 matrix $\mathbf{A}$ of a simple rotation and the decomposition of Eq. (2) contains two independent angles, so the matrix $\mathbf{B}$ must already be encoded in $\mathbf{A}$. In fact, the two matrices share the same four unique eigenvectors, which, analogous to the 3D case, can be used to construct an orthonormal set of vectors that span two orthogonal planes of rotation. The matrix $\mathbf{A}$ describes rotations in one plane and $\mathbf{B}$ in the other and the eigenvalues of $\mathbf{A}$ and $\mathbf{B}$ determine which plane is associated with which matrix.

In what follows, we shall clarify how the above-mentioned relations between the two matrices can be used to construct $\mathbf{B}$ from a given $\mathbf{A}$, such that their linear combination [Eq. (2)] gives rise to a rotation matrix according to Eq. (4). To this end, let us consider a skew-symmetric matrix

$$\mathbf{B} = \begin{pmatrix} 0 & b_{12} & b_{13} & b_{14} \\ -b_{12} & 0 & b_{23} & b_{24} \\ -b_{13} & -b_{23} & 0 & b_{34} \\ -b_{14} & -b_{24} & -b_{34} & 0 \end{pmatrix}$$

that satisfies the property $\mathbf{B}^3 = -\mathbf{B}$. According to Sec. II B, there exist two vectors

$$\vec{b}_1 = \begin{pmatrix} -b_{23} \\ b_{13} \\ -b_{12} \end{pmatrix}, \quad \vec{b}_2 = \begin{pmatrix} b_{14} \\ b_{24} \\ b_{34} \end{pmatrix},$$

for which

$$\vec{b}_1 \cdot \vec{b}_2 = 0, \quad \vec{b}_1 \cdot \vec{b}_1 + \vec{b}_2 \cdot \vec{b}_2 = 1 \tag{7}$$

must hold for $\mathbf{B}$ to have rank 2. However, $\vec{b}_1$ and $\vec{b}_2$ cannot be chosen independently of $\vec{a}_1$ and $\vec{a}_2$, because of the third condition in Eq. (3). Under the constraints of Eqs. (6) and (7), the resulting equations can be solved to give

$$b_{12} = \mp a_{34}, \quad b_{13} = \pm a_{24}, \quad b_{14} = \mp a_{23},$$
$$b_{23} = \mp a_{14}, \quad b_{24} = \pm a_{13}, \quad b_{34} = \mp a_{12},$$

or, equivalently,

$$\vec{b}_1 = \pm \vec{a}_2, \quad \vec{b}_2 = \pm \vec{a}_1.$$

Note that choosing the top or bottom sign corresponds to choosing between $\mathbf{B}$ and $\mathbf{B}^\top = -\mathbf{B}$, which boils down to a positive or negative rotation in the angle β. We can therefore, without loss of generality, fix the signs of the elements of $\mathbf{B}$ and allow β to take both positive and negative values.

D. Sampling the angles α and β

For a random rotation $\mathbf{R}_3(\gamma)$ in 3D, the rotation angle $\gamma \in [0, 2\pi)$ is far from being uniformly distributed. Instead, one has $\gamma \sim [1 - \cos(\gamma)]/2\pi$ [17], where the notation $z \sim p(z)$ implies that z is distributed according to $p(z)$. In the 4D case, the angles $\alpha, \beta \in [0, 2\pi)$ entering Eq. (2) follow the joint probability distribution function (PDF) [17]

$$p(\alpha, \beta) = \frac{[\cos(\alpha) - \cos(\beta)]^2}{4\pi^2} \tag{8}$$

and are therefore not independent, i.e., $p(\alpha, \beta) \neq p(\alpha)\, p(\beta)$. However, with the help of the auxiliary variables u and v, satisfying $\alpha = u + v$ and $\beta = v - u$, one can uncouple the angles of rotation. This variable substitution decomposes $p(\alpha, \beta)$ into a product of independent and identical distributions for $u, v \in [0, 2\pi]$ with the functional form (see Appendix B)

$$p(z) = \frac{\sin(z)^2}{\pi}. \tag{9}$$

In principle, it is possible to expand the corresponding cumulative distribution function (CDF) in the limit of $z \ll 1$ and sample small-angle values for α and β using inverse transform sampling, as demonstrated in Appendix B. However, it turns out that the properties of a large-scale rotation generated by multiple, consecutive, small-angle rotations are mostly unaffected by the small-angle distributions of α and β. In fact, for the large-scale rotations to be uniformly distributed over SO(4), one only has to ensure that (i) sufficiently many small-angle rotations are used to cover all of the 3-sphere and (ii) the small-angle rotations are reversible. The latter requirement is automatically fulfilled when the vectors $\vec{a}_1$ and $\vec{a}_2$ are sampled from a uniform distribution on the 2-sphere, because then every 4D rotation (characterized by two orthogonal planes associated with $\vec{a}_1$ and $\vec{a}_2$) is as probable as its counter rotation (given by $-\vec{a}_1$ and $-\vec{a}_2$). We can therefore choose arbitrary distributions to sample small-angle values for α and β.

III. EFFICIENT ALGORITHM FOR THE GENERATION OF SMALL-ANGLE 4D ROTATION MATRICES

We are now ready to formulate an algorithm to generate random, small-angle, 4D rotation matrices, which can be used in Markov-chain Monte Carlo sampling schemes to explore the space of 4D rotations. In Sec. II D, we already mentioned that the distributions, from which small-angle values of α and β are drawn, can be chosen arbitrarily. Here we will thus make use of uniformly distributed pseudorandom numbers to minimize computational costs. One way of generating the random vectors $\vec{a}^*_{i=1,2}$ needed to construct the matrices $\mathbf{A}$ and $\mathbf{B}$ would be to apply a random SO(3) element to a pair of orthogonal vectors, e.g., $(1,0,0)^\top$ and $(0,1,0)^\top$. However, the efficiency of such a construction scheme would largely depend on the method used to generate the SO(3) elements.

Instead, we rely on the following algorithm, which is efficient in the sense that it only requires a total of six random numbers [corresponding to the dimension of SO(4)].

(1) Generate two uniformly distributed random numbers, $R_1 \sim \mathcal{U}_{[-1,1]}$ and $R_2 \sim \mathcal{U}_{[0,2\pi]}$, where $\mathcal{U}_{[s,t]}$ indicates a continuous uniform distribution on the interval $[s,t]$. The auxiliary unit vector $\vec{a}^*_1$ is then given by (see Appendix A)

$$\vec{a}^*_1 = \begin{pmatrix} \sqrt{1-R_1^2}\cos(R_2) \\ \sqrt{1-R_1^2}\sin(R_2) \\ R_1 \end{pmatrix}.$$

(2) Generate an additional random number $R_3 \sim \mathcal{U}_{[0,2\pi]}$. The auxiliary unit vector

$$\vec{a}^*_2 = \begin{pmatrix} R_1\cos(R_2)\cos(R_3) + \sin(R_2)\sin(R_3) \\ R_1\sin(R_2)\cos(R_3) - \cos(R_2)\sin(R_3) \\ -\sqrt{1-R_1^2}\cos(R_3) \end{pmatrix}$$

is then perpendicular to $\vec{a}^*_1$ (see Appendix A). Note that if random number generation is fast compared to the evaluation of trigonometric functions, one can obtain the sine and cosine of R_2 and R_3 by drawing two points on a unit circle instead.

(3) Generate a fourth random number $R_4 \sim \mathcal{U}_{[0,1]}$, which defines the "mixing ratio" between the auxiliary vectors, i.e.,

$$\vec{a}_1 = \vec{a}^*_1\sqrt{R_4}, \quad \vec{a}_2 = \vec{a}^*_2\sqrt{1-R_4}.$$

(4) Generate two additional random numbers, $R_5, R_6 \sim \mathcal{U}_{[0,1]}$, and set

$$\alpha = \varepsilon R_5, \quad \beta = \varepsilon R_6,$$

for $\varepsilon \ll 1$.

(5) Compute the rotation matrix

$$\mathbf{R}_4 = e^{\mathbf{S}_4} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{34} & r_{44} \end{pmatrix}$$

associated with $\mathbf{S}_4 = \alpha\mathbf{A} + \beta\mathbf{B}$ via the Rodrigues formula [Eq. (4)]. This corresponds to the following elementwise calculations:

$$\begin{aligned}
r_{11} &= \left(a_{1,y}^2 + a_{1,z}^2 + a_{2,x}^2\right)\Delta + \cos(\beta), \\
r_{12} &= h_{12}\Delta - a_{1,z}\sin(\alpha) - a_{2,z}\sin(\beta), \\
r_{21} &= h_{21}\Delta + a_{1,z}\sin(\alpha) + a_{2,z}\sin(\beta), \\
r_{22} &= \cos(\alpha) - \left(a_{1,y}^2 + a_{2,x}^2 + a_{2,z}^2\right)\Delta, \\
r_{13} &= h_{13}\Delta + a_{1,y}\sin(\alpha) + a_{2,y}\sin(\beta), \\
r_{31} &= h_{31}\Delta - a_{1,y}\sin(\alpha) - a_{2,y}\sin(\beta), \\
r_{23} &= h_{23}\Delta - a_{1,x}\sin(\alpha) - a_{2,x}\sin(\beta), \\
r_{32} &= h_{32}\Delta + a_{1,x}\sin(\alpha) + a_{2,x}\sin(\beta), \\
r_{33} &= \cos(\alpha) - \left(a_{1,z}^2 + a_{2,x}^2 + a_{2,y}^2\right)\Delta, \\
r_{14} &= h_{14}\Delta + a_{2,x}\sin(\alpha) + a_{1,x}\sin(\beta), \\
r_{41} &= h_{41}\Delta - a_{2,x}\sin(\alpha) - a_{1,x}\sin(\beta), \\
r_{24} &= h_{24}\Delta + a_{2,y}\sin(\alpha) + a_{1,y}\sin(\beta), \\
r_{42} &= h_{42}\Delta - a_{2,y}\sin(\alpha) - a_{1,y}\sin(\beta), \\
r_{34} &= h_{34}\Delta + a_{2,z}\sin(\alpha) + a_{1,z}\sin(\beta), \\
r_{43} &= h_{43}\Delta - a_{2,z}\sin(\alpha) - a_{1,z}\sin(\beta), \\
r_{44} &= \left(a_{2,x}^2 + a_{2,y}^2 + a_{2,z}^2\right)\Delta + \cos(\beta),
\end{aligned}$$

where $\vec{a}_i = (a_{i,x}, a_{i,y}, a_{i,z})^\top$, $\Delta = \cos(\alpha) - \cos(\beta)$, and

$$\begin{aligned}
h_{12} &= h_{21} = a_{2,x}a_{2,y} - a_{1,x}a_{1,y}, \\
h_{13} &= h_{31} = a_{2,x}a_{2,z} - a_{1,x}a_{1,z}, \\
h_{23} &= h_{32} = a_{2,y}a_{2,z} - a_{1,y}a_{1,z}, \\
h_{14} &= h_{41} = a_{1,z}a_{2,y} - a_{1,y}a_{2,z}, \\
h_{24} &= h_{42} = a_{1,x}a_{2,z} - a_{1,z}a_{2,x}, \\
h_{34} &= h_{43} = a_{1,y}a_{2,x} - a_{1,x}a_{2,y}.
\end{aligned}$$

A possible way of speeding up the algorithm is to solely rely on simple rotations. With $\beta = 0$, $\sin(\beta) = 0$, and $\cos(\beta) = 1$, one less random number is required in step (4) and the expressions for r_{ij} in step (5) get simplified.

The algorithm can also be used to generate uniformly distributed SO(4) elements by replacing the fourth step with the following (see Appendix B).

TABLE I. Runtimes of the different algorithms to generate SO(4) rotation matrices in numerical tests.

Algorithm	Angles	Median runtime
Sec. III	$\alpha, \beta \sim p(\alpha, \beta)$	2.332 µs
Sec. III	$\alpha, \beta \sim \mathcal{U}_{[0,\varepsilon]}$	207.980 ns
Sec. III	$\alpha \sim \mathcal{U}_{[0,\varepsilon]}, \beta = 0$	182.310 ns
Eq. (1)	$\alpha, \beta \sim \mathcal{U}_{[0,\varepsilon]}$	1.618 µs

(4) Generate two additional random numbers, $R_5, R_6 \sim \mathcal{U}_{[0,1]}$, and find the root of

$$f(z_i) = 2z_i - \sin(2z_i) - 8\pi R_i$$

for $z_5 = u$ and $z_6 = v$. The angles of rotation are then given by

$$\alpha = u + v, \quad \beta = v - u.$$

IV. TESTING THE ALGORITHM

It is straightforward to numerically verify that the matrices generated by the algorithm proposed in Sec. III satisfy

$$\mathbf{R}_4 \mathbf{R}_4^\top = \mathbf{I}_4, \quad \det(\mathbf{R}_4) = 1,$$

and are therefore elements of SO(4). In this section, we compare its runtime performance to Eq. (1) and verify that the subsequent application of multiple small-angle rotations gives rise to a uniformly distributed large-scale rotation.

A. Runtime performance

The algorithms in Eq. (1) and Sec. III were both implemented in Julia [18] and timed using the `@benchmark` macro provided by the BenchmarkTools.jl package [19]. To generate the random O(4) elements $\mathbf{Q}_4$ entering Eq. (1), we translated a Python implementation [20] of an algorithm used to generate SO(4) elements [10] into efficient JULIA code. We tested both small-angle sampling and sampling of the joint angle distribution $p(\alpha, \beta)$ [Eq. (8)]. The runtime measurements of each algorithm were performed using an installation of JULIA v.1.8.0 on a machine with a 1.2 GHz Quad-Core Intel Core i7 processor.

Our results are collected in Table I. Using the algorithm of Sec. III to generate small-angle double rotations is an order of magnitude faster than Eq. (1) and only marginally slower than restricting oneself to simple rotations with $\beta = 0$.

B. Asymptotic uniformity

If the rotation angles α and β are randomly drawn from their joint PDF [Eq. (8)], we can verify whether the resulting random 4D rotation matrices are uniformly distributed. This can be realized, e.g., by generating a set of random matrices $\{\mathbf{R}_4^{(k)}\}_{k=1}^K$ using the algorithm in Sec. III with a modified fourth step and inspecting the distribution of random points

$$\vec{t}\,'_k = \mathbf{R}_4^{(k)} \vec{t} \tag{10}$$

that emerges when the matrices are used to rotate a fixed unit 4-vector $\vec{t}$. For small-valued rotation angles, multiple rotations

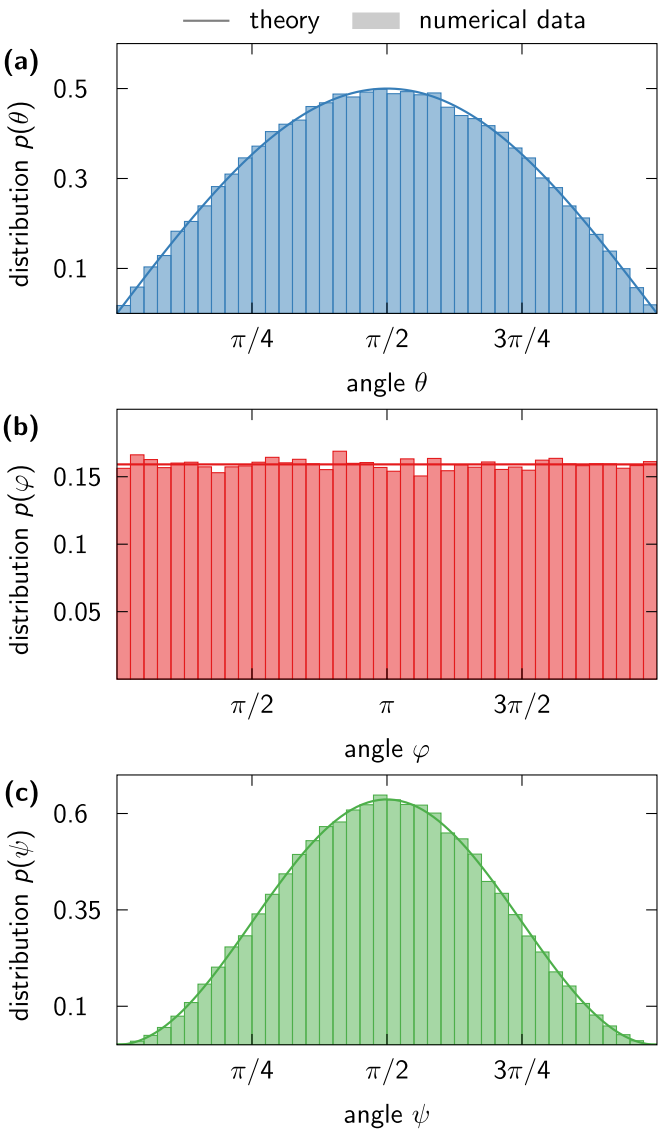

FIG. 1. Distributions of hyperspherical angles corresponding to a uniform distribution on S^3. A set $\{\vec{t}'_k\}_{k=1}^K$, $K = 10^5$, of random points on the unit 3-sphere were generated by repeatedly applying $N = 10^5$ random, small-angle ($\varepsilon = 0.05$) rotation matrices to a fixed 4-vector $\vec{t} = (0, 0, 0, 1)^\top$. The numerical data (histograms) nicely reproduce the angle distributions (a) $p(\theta) = \sin(\theta)/2$, (b) $p(\varphi) = 1/2\pi$, and (c) $p(\psi) = 2\sin(\psi)^2/2$, which one expects to find if the points $\vec{t}'_k$ uniformly cover the surface of the 3-sphere.

are needed to compose a large-scale rotation, so Eq. (10) has to be replaced with

$$\vec{t}\,'_k = \mathbf{R}_4^{(1,k)} \mathbf{R}_4^{(2,k)} \ldots \mathbf{R}_4^{(N,k)} \vec{t}. \tag{11}$$

In hyperspherical coordinates, we have (see Appendix C)

$$\vec{t}\,'_k = \begin{pmatrix} \sin(\psi_k)\sin(\theta_k)\cos(\varphi_k) \\ \sin(\psi_k)\sin(\theta_k)\sin(\varphi_k) \\ \sin(\psi_k)\cos(\theta_k) \\ \cos(\psi_k) \end{pmatrix}$$

because $|\vec{t}| = 1$, so verifying whether the points $\vec{t}'_k$ are uniformly distributed on S^3 amounts to comparing the

distributions of $\{\theta_k\}_{k=1}^K$, $\{\varphi_k\}_{k=1}^K$, and $\{\psi_k\}_{k=1}^K$ to

$$p(\theta) = \frac{\sin(\theta)}{2}, \quad p(\varphi) = \frac{1}{2\pi}, \quad p(\psi) = \frac{2\,\sin(\psi)^2}{\pi}. \tag{12}$$

In Fig. 1, the hyperspherical coordinates of $K = 10^5$ random points, generated via Eq. (11) with $N = 10^5$ and $\varepsilon = 0.05$, are collected into histograms and displayed next to the theoretical predictions of Eq. (12). Visually, the numerical data nicely agree with the analytic PDFs. The comparison can also be made quantitative with the help of the Kolmogorov-Smirnov (KS) test, which considers the differences

$$\delta_+^{(k)} = k/K - P(z_k),$$
$$\delta_-^{(k)} = P(z_k) - (k-1)/K$$

between the empirical distribution function and the CDF $P(z)$ of the variable $z \in \{\theta, \varphi, \psi\}$. The KS statistic

$$\mathcal{S}_z = \sqrt{K} \max_{1\leqslant k\leqslant K} (\delta_+^{(k)}, \delta_-^{(k)}) \tag{13}$$

gives a measure for how likely it is that the sample $\{z_k\}_{k=1}^K$ was drawn from the PDF $p(z)$. For the distributions in Fig. 1, we have

$$P(\theta) = \sin(\theta/2)^2, \quad P(\varphi) = \varphi/2\pi,$$
$$P(\psi) = [\psi - \sin(\psi)\cos(\psi)]/\pi,$$

and the corresponding KS statistics evaluate to $\mathcal{S}_\theta \approx 0.90$, $\mathcal{S}_\varphi \approx 0.91$, and $\mathcal{S}_\psi \approx 1.04$. The largest KS statistic has an associated p value of 0.23, so we can be quite confident in our assumption that the cumulative rotation of $N = 10^5$ small-angle rotation matrices results in a uniformly distributed 4D rotation.

Figure 2 investigates how many small-angle rotation matrices N are actually needed to construct a uniformly distributed SO(4) element. We thereby considered 100 sets of points $\{\vec{t}\,'_k\}_{k=1}^K$ with $K = 1000$ and for each set we evaluated the KS statistic [Eq. (13)] for the hyperspherical coordinates θ, φ, and ψ. For $\varepsilon = 0.05$ [Fig. 2(a)], which was also used to generate the data in Fig. 1, we find that approximately 10 000 small-angle matrices will guarantee a uniform distribution, whereas for $\varepsilon = 0.5$ [Fig. 2(b)] only around 100 matrices are needed. When used with Monte Carlo methods, the parameter ε can therefore be seen as a magnitude for the step size, which can be tuned, e.g., to optimize the fraction of accepted moves.

V. APPLICATION TO EQUIMOMENTAL SYSTEMS

Based on Newton's equations of motion, two systems of point masses are dynamically equivalent (or equimomental) if the total mass $M = \sum_i m_i$ of both systems and the corresponding center-of-mass vectors $\vec{r}_{\text{com}} = (x_{\text{com}}, y_{\text{com}}, z_{\text{com}})^\top$ and inertia tensors $\mathcal{I}$ are identical. Reference [4] introduces a general framework for the generation of such equimomental systems. The central quantity of interest in this formalism is the pseudoinertia matrix

$$\Xi = \left(\begin{array}{ccc|c} \frac{1}{2}(-I_{xx}+I_{yy}+I_{zz}) & -I_{xy} & -I_{xz} & Mx_{\text{com}} \\ -I_{xy} & \frac{1}{2}(I_{xx}-I_{yy}+I_{zz}) & -I_{yz} & My_{\text{com}} \\ -I_{xz} & -I_{yz} & \frac{1}{2}(I_{xx}+I_{yy}-I_{zz}) & Mz_{\text{com}} \\ \hline Mx_{\text{com}} & My_{\text{com}} & Mz_{\text{com}} & M \end{array}\right), \tag{14}$$

which encodes the elements I_{ij} of the inertia tensor $\mathcal{I}$ and the total mass M (diagonal blocks), as well as the center-of-mass vector $\vec{r}_{\text{com}}$ scaled by M (off-diagonal blocks). The matrix Ξ can be readily constructed from exterior products as follows:

$$\Xi = \sum_{i=1}^N m_i \vec{p}_i \vec{p}_i^{\,\top}, \tag{15}$$

where N denotes the number of point masses and $\vec{p}_i = (p_{i,x}, p_{i,y}, p_{i,z}, 1)^\top$ is a homogeneous extended position vector. Without loss of generality, the Ξ matrix can be diagonalized by a suitable element $\mathbf{G}$ of the special Euclidean group SE(3), i.e.,

$$\Xi' = \mathbf{G}\Xi\mathbf{G}^\top. \tag{16}$$

Diagonalization by $\mathbf{G}$ amounts to a combined translation and rotation that sets $\vec{r}_{\text{com}} \to \vec{0}$ and orients the system of point masses in the principal frame. From here onwards, let us therefore assume that Ξ is diagonal. Defining the diagonal matrix $\mathbf{D} = (M^{-1}\Xi)^{1/2} = \text{diag}(a, b, c, 1)$ and the homogeneous extended position vectors $\vec{q}_i$ that satisfy $\vec{p}_i = \mathbf{D}\vec{q}_i$ leads to a convenient decomposition of Ξ, namely

$$\Xi = \sum_{i=1}^N m_i \vec{p}_i \vec{p}_i^{\,\top} = \sum_{i=1}^N m_i \mathbf{D}\vec{q}_i \vec{q}_i^{\,\top}\mathbf{D}^\top = M\mathbf{D}\mathbf{D}^\top. \tag{17}$$

In the last step, we made use of the identity

$$\sum_{i=1}^N m_i \vec{q}_i \vec{q}_i^{\,\top} = M\mathbf{I}_4, \tag{18}$$

which follows from the fact that $\mathbf{D}\mathbf{D}^\top = M^{-1}\Xi$ must hold, by construction. Now, the structure of Ξ makes it invariant to arbitrary 4D rotations of the $\vec{q}_i$ vectors, as seen by

$$\begin{aligned} \Xi &= M\mathbf{D}\mathbf{D}^\top = M\mathbf{D}\mathbf{R}_4\mathbf{R}_4^\top\mathbf{D}^\top \\ &= \sum_{i=1}^N m_i \mathbf{D}\mathbf{R}_4 \vec{q}_i \vec{q}_i^{\,\top}\mathbf{R}_4^\top\mathbf{D}^\top, \end{aligned} \tag{19}$$

which allows us to transform between equimomental systems using elements of SO(4). Equation (19) implies that the rotation matrix $\mathbf{R}_4$ shifts the homogeneous extended positions $\vec{p}_i$ and redistributes m_i of the point masses

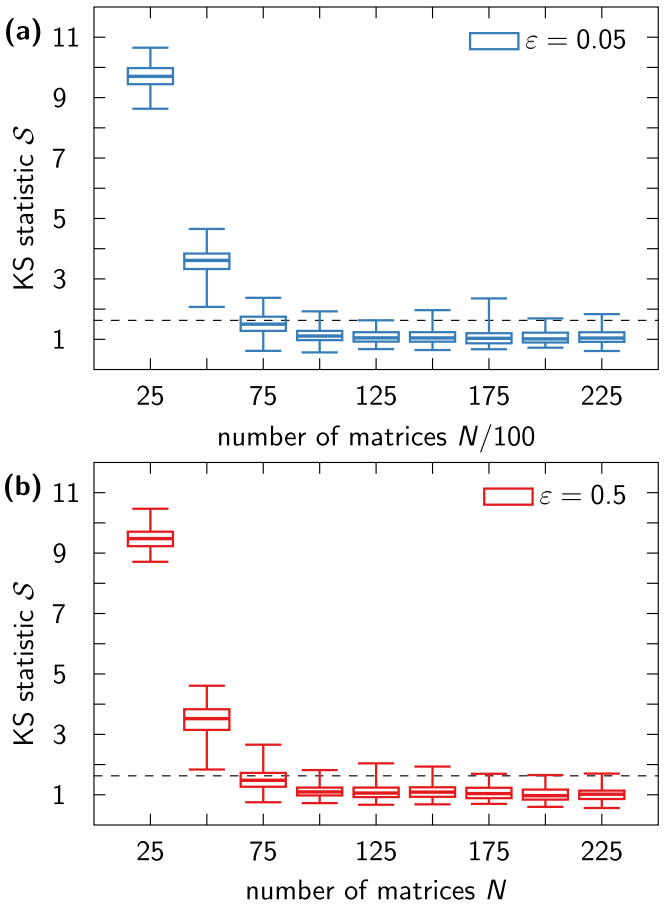

FIG. 2. Distributions of KS statistics $\mathcal{S} = \max(\mathcal{S}_\theta, \mathcal{S}_\varphi, \mathcal{S}_\psi)$ depending on the number of matrices N behind the cumulative rotation. For each $\mathcal{S}$ a corresponding p value can be calculated and the dashed lines mark $p = 0.01$. We can therefore claim with confidence that the collective rotation of N small-angle rotation matrices generates uniformly distributed SO(4) elements if the corresponding $\mathcal{S}$ distribution mostly lies below the dashed line, where $p > 0.01$. (a) For $\varepsilon = 0.05$, tens of thousands of small-angle rotation matrices are needed, whereas (b) $\varepsilon = 0.5$ only requires $N \approx 100$.

as follows:

$$\vec{p}_i \to \vec{p}_i' = \vec{w}_i / w_{i,4}, \quad m_i \to m_i' = m_i w_{i,4}^2,$$

where the scaling via the fourth element of $\vec{w}_i = \mathbf{DR}_4 \vec{q}_i$ ensures that $\vec{p}_i'$ is a proper homogeneous extended vector.

Reference [11] provides a particular application of the above framework to coarse-grained molecular dynamics simulations of cholesterol. The coarse-grained representation of cholesterol [21] in the Martini 2 force field [22] contains coupled triangular constraints that require exceedingly strict settings of the LINCS constraint-solving algorithm [23] to converge. By constructing particle arrangements equimomental to the constrained scaffold of cholesterol via small-angle 4D rotations generated by the algorithm of Sec. III, it was possible to speed up convergence of the LINCS algorithm without affecting the original parametrization of the force field.

VI. CONCLUSIONS

In this paper, we have presented a geometrically appealing way to construct a 4D rotation matrix from two orthogonal 3D vectors and two angles of rotation (Sec. II). The explicit control of the angle variables gives us an advantage over established methods for the generation of SO(4) elements, which we exploited to formulate an algorithm to exclusively generate small-angle rotation matrices in four dimensions (Sec. III). We also demonstrated (Sec. IV B) that the repeated application of multiple consecutive small-angle rotations, generated by our proposed algorithm, results in uniform sampling of 4D rotations, as required for an unbiased algorithm.

The algorithm is efficient in the sense that it only requires the minimal six pseudorandom numbers, is an order of magnitude faster than alternative schemes (Sec. IV A), and can be used in combination with Monte Carlo methods to solve optimization problems involving 4D rotations in physics [4] and engineering [5]. A practical example is the optimization of the constrained scaffold of a coarse-grained cholesterol model [11] (Sec. V), which is used in combination with the Martini force field in molecular dynamics simulations of biological systems.

ACKNOWLEDGMENTS

We thank Professor Emeritus Dr. A. J. Hanson and Dr. A. Szabo for discussions. This research was supported by the Max Planck Society. B.F. thanks the Alexander von Humboldt Foundation for funding.

APPENDIX A: GENERATING UNIFORMLY DISTRIBUTED POINTS ON A 3D SPHERE

In spherical coordinates only two angles, $\theta \in [0, \pi]$ and $\varphi \in [0, 2\pi)$, are needed to define a point on the unit sphere $S^2 = \{\vec{r} \in \mathbb{R}^3 \mid \|\vec{r}\| = 1\}$. The position of the point is given by

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \sin(\theta)\cos(\varphi) \\ \sin(\theta)\sin(\varphi) \\ \cos(\theta) \end{pmatrix} \tag{A1}$$

in Cartesian coordinates. Many algorithms exist to generate uniformly distributed points on S^2. Here, we essentially draw random values for θ and φ from the appropriate distributions.

A uniform distribution on S^2 is given in spherical coordinates by $p(\theta, \phi) = \sin(\theta)/4\pi$, which implies that the azimuthal angle φ is uniformly distributed on the interval $[0, 2\pi]$, i.e., $\varphi \sim \mathcal{U}_{[0,2\pi]}$. The polar angle θ is distributed according to $p(\theta) = \sin(\theta)/2$, so in order to draw from this distribution, we employ inverse transform sampling and construct $\theta \sim p(\theta)$ from a uniformly distributed random number $R \in [0, 1]$ as follows:

$$\theta = \arccos(1 - 2R).$$

Note that for $R \sim \mathcal{U}_{[0,1]}$ we have $1 - 2R \sim \mathcal{U}_{[-1,1]}$, so

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \sqrt{1 - R_1^2}\cos(R_2) \\ \sqrt{1 - R_1^2}\sin(R_2) \\ R_1 \end{pmatrix}, \tag{A2}$$

with $R_1 \sim \mathcal{U}_{[-1,1]}$ and $R_2 \sim \mathcal{U}_{[0,2\pi]}$, uniformly distributed on S^2. Here, we exploited the fact that $\sin[\arccos(x)] = \sqrt{1-x^2}$ must hold for $-1 \leqslant x \leqslant 1$.

If Eq. (A2) is used as a proxy for the auxiliary vector $\vec{a}_1^*$, then there are infinitely many candidates for $\vec{a}_2^*$ lying in a plane spanned by the orthonormal vectors

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos(\theta)\cos(\varphi) \\ \cos(\theta)\sin(\varphi) \\ -\sin(\theta) \end{pmatrix}, \quad \begin{pmatrix} x'' \\ y'' \\ z'' \end{pmatrix} = \begin{pmatrix} \sin(\varphi) \\ -\cos(\varphi) \\ 0 \end{pmatrix}.$$

We can randomly choose a candidate of unit length from said plane with a random rotation, giving

$$\vec{a}_2^* = \cos(R_3)\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} + \sin(R_3)\begin{pmatrix} x'' \\ y'' \\ z'' \end{pmatrix}$$

for $R_3 \sim \mathcal{U}_{[0,2\pi]}$.

APPENDIX B: SAMPLING SMALL ROTATION ANGLES

Equation (8) in the main text can be rewritten as follows [17]:

$$p(\alpha,\beta) = \frac{1}{\pi^2}\sin\left(\frac{\alpha+\beta}{2}\right)^2 \sin\left(\frac{\alpha-\beta}{2}\right)^2,$$

which invites the coordinate substitutions $u = (\alpha-\beta)/2$ and $v = (\alpha+\beta)/2$ to uncouple the variables. Both u and v are then distributed according to Eq. (9) in the main text. We can expand the diamond-shaped domain of $(u,v) = [(\alpha-\beta)/2, (\alpha+\beta)/2]$, which emerges for $\alpha,\beta \in [0,2\pi]$, to the square $[-\pi,\pi]\times[0,2\pi]$ or, equivalently, $[0,2\pi]\times[0,2\pi]$.

The CDF of Eq. (9) is given by

$$P(z) = \frac{2z - \sin(2z)}{4\pi}$$

for $z \in [0,2\pi]$ and can be used to generate random values for u and v via inverse transform sampling, which amounts to solving

$$P(z) = R_z \tag{B1}$$

for $z \in \{u,v\}$ and a set of random uniformly distributed numbers $R_z = R_u, R_v \sim \mathcal{U}_{[0,1]}$. In the small-angle limit, Eq. (B1) reduces to

$$\varepsilon^3 R_z = \frac{z^3}{3\pi} + O(z^4)$$

for $\varepsilon \ll 1$, which gives rise to the following random angles of rotations:

$$\alpha = \varepsilon\left(R_u^{1/3} + R_v^{1/3}\right), \quad \beta = \varepsilon\left(R_v^{1/3} - R_u^{1/3}\right).$$

However, as discussed in Sec. II D and demonstrated on explicit examples in Sec. IV, the distributions behind α and β do not affect the properties of the large-scale rotations resulting from multiple, consecutive, small-angle rotations.

APPENDIX C: HYPERSPHERICAL COORDINATES

While spherical coordinates are ideal for systems in $\mathbb{R}^3$ with rotation symmetries, they can also be generalized to so-called hyperspherical coordinates in higher dimensions. In 4D, an angle coordinate $\psi \in [0,2\pi]$ is added to the spherical coordinates r, θ, and φ, such that the Cartesian coordinates of a 4-vector are given by

$$\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} r\sin(\psi)\sin(\theta)\cos(\varphi) \\ r\sin(\psi)\sin(\theta)\sin(\varphi) \\ r\sin(\psi)\cos(\theta) \\ r\cos(\psi) \end{pmatrix}.$$

Conversely, one can calculate r, θ, φ, and ψ from a given 4-vector $(x,y,z,w)^\top$ as follows:

$$\begin{aligned} r &= \sqrt{x^2+y^2+z^2+w^2}, \\ \psi &= \arccos(w/r), \\ \theta &= \arccos(z/\sqrt{r^2-w^2}), \\ \varphi &= \begin{cases} \arccos(x/\sqrt{r^2-w^2-z^2}), & y \geqslant 0, \\ 2\pi - \arccos(x/\sqrt{r^2-w^2-z^2}), & y < 0. \end{cases} \end{aligned}$$

Note that, in this paper, we only consider 4-vectors of unit length, so $r \equiv 1$.

[1] M. Karlsson, Four-dimensional rotations in coherent optical communications, J. Light. Technol. **32**, 1246 (2014).

[2] D. Oliver, *The Shaggy Steed of Physics* (Springer-Verlag, New York, 2004).

[3] A. Borowicz, On using quaternionic rotations for indpendent component analysis, in *Proceedings of the 2018 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), Poznan, Poland* (IEEE, Piscataway, NJ, 2018), pp. 114–119.

[4] L. P. Laus and J. M. Selig, Rigid body dynamics using equimomental systems of point-masses, Acta Mech. **231**, 221 (2020).

[5] J. Wu, Y. Sun, M. Wang, and M. Liu, Hand-eye calibration: 4-D procrustes analysis approach, IEEE Trans. Instrum. Meas. **69**, 2966 (2020).

[6] S. Sarabandi and F. Thomas, Approximating displacements in $\mathbb{R}^3$ by rotations in $\mathbb{R}^4$ and its application to pointcloud registration, IEEE Trans. Robot. **38**, 2652 (2022).

[7] P. Lounesto, *Clifford Algebras and Spinors* (Cambridge University Press, Cambridge, UK, 2001).

[8] C. A. León, J.-C. Massé, and L.-P. Rivest, A statistical model for random rotations, J. Multivariate Anal. **97**, 412 (2006).

[9] F. Mezzadri, How to generate random matrices from the classical compact groups, Not. Am. Math. Soc. **54**, 592 (2007).

[10] G. W. Stewart, The efficient generation of random orthogonal matrices with an application to condition estimators, SIAM J. Numer. Anal. **17**, 403 (1980).

[11] B. Fábián, S. Thallmair, and G. Hummer, Optimal bond-constraint topology for molecular dynamics simulations of cholesterol, J. Chem. Theory Comput. **19**, 1592 (2023).

[12] A. J. Hanson, *Visualizing Quaternions* (Morgan Kaufmann Publishers, Burlington, MA, 2006).

[13] A. Perez-Gracia and F. Thomas, On Cayley's factorization of 4D rotations and applications, Adv. Appl. Clifford Algebras **27**, 523 (2017).

[14] S. Sarabandi, A. Perez-Gracia, and F. Thomas, On Cayley's factorization with an application to the orthonormalization of noisy rotation matrices, Adv. Appl. Clifford Algebras **29**, 49 (2019).

[15] M. Erdoğdu and M. Özdemir, Simple, double and isoclinic rotations with a viable algorithm, Math. Sci. Appl. E-Notes **8**, 11 (2020).

[16] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey *et al.*, SciPy 1.0: Fundamental algorithms for scientific computing in Python, Nat. Methods **17**, 261 (2020).

[17] H. Rummler, On the distribution of rotation angles: How great is the mean rotation angle of a random rotation? Math. Intell. **24**, 6 (2002).

[18] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, Julia: A fresh approach to numerical computing, SIAM Rev. **59**, 65 (2017).

[19] https://github.com/JuliaCI/BenchmarkTools.jl.

[20] See https://github.com/scipy/scipy/blob/v1.4.1/scipy/stats/_multivariate.py for the original code.

[21] M. N. Melo, H. I. Ingólfsson, and S. J. Marrink, Parameters for Martini sterols and hopanoids based on a virtual-site description, J. Chem. Phys. **143**, 243152 (2015).

[22] S. J. Marrink, H. J. Risselada, S. Yefimov, D. P. Tieleman, and A. H. de Vries, The MARTINI force field: Coarse grained model for biomolecular simulations, J. Phys. Chem. B **111**, 7812 (2007).

[23] B. Hess, H. Bekker, H. J. C. Berendsen, and J. G. E. M. Fraaije, LINCS: A linear constraint solver for molecular simulations, J. Comput. Chem. **18**, 1463 (1997).