Eur. Phys. J. E (2018) **41**: 87
DOI 10.1140/epje/i2018-11694-7

THE EUROPEAN PHYSICAL JOURNAL E

Regular Article

Implementing efficient concerted rotations using Mathematica and C code[⋆]

Luca Tubiana[1,a], Miroslav Jurásek[2,3], and Ivan Coluzza[4]

[1] Computational Physics Department, University of Vienna, Sensengasse 8/10, 1090, Vienna, Austria
[2] Faculty of Science, Masaryk University, Kotlářská 2, 602 00 Brno, Czech Republic
[3] CEITEC - Central European Institute of Technology, Kamenice 5, 625 00 Brno, Czech Republic
[4] CIC biomaGUNE Parque Cientfico y Tecnolgico de Gipuzkoa, Paseo Miramn 182, 20014 Donostia / San Sebastin, Gipuzkoa, Spain

Received 22 February 2018 and Received in final form 21 June 2018
Published online: 20 July 2018

Abstract. In this article we demonstrate a general and efficient metaprogramming implementation of concerted rotations using Mathematica. Concerted rotations allow the movement of a fixed portion of a polymer backbone with fixed bending angles, like a protein, while maintaining the correct geometry of the backbone and the initial and final points of the portion fixed. Our implementation uses Mathematica to generate a C code which is then wrapped in a library by a Python script. The user can modify the Mathematica notebook to generate a set of concerted rotations suited for a particular backbone geometry, without having to write the C code himself. The resulting code is highly optimized, performing on the order of thousands of operations per second.

1 Introduction

Recent research demonstrates a renewed interest for the investigation of polypeptides, particularly intrinsically disordered proteins (IDP) [1–4], with the intention of understanding the physical principles shaping their conformational space and to design synthetic proteins as well as protein-like polymers. In all these cases the exploration of the configuration space of the protein backbone poses a serious limitation [5]. This is often tackled by using coarse-grained models, characterized by a reduced amount of structural details and focusing on a small number of interactions between the residues [6–16].

Monte Carlo schemes would allow for a quick survey of the configurations space for a coarse-grained protein backbone, but the systematic rejection of pivot moves when the protein is compact and the backbone distortions introduced by other moves such as crankshaft rotations limit their applicability [10, 17–21].

A solution to the above problem is offered by the use of concerted rotations, first developed by Theodorou *et al.* [22] to investigate melts of polymers. This and other similar approaches were later adopted by different groups to move local portions of protein backbones [23–30]. In these algorithms a set of dihedral angles is changed in a concerted way so that a limited portion of the backbone is moved without distorting it and without affecting the position and orientation of its ends. Unfortunately, their relatively difficult implementation compared to other moves, coupled to the computational demands of their calculation, often keep them from being adopted in the initial developing phase of coarse-grained models.

An elegant approach to the problem of concerted rotations has recently been proposed by Zamuner *et al.* [31] who have generalized concerted rotations by mapping them on a differential manifold. The manifold is identified by the constraints imposed by the fixed positions of the initial and final atoms of the chain portion to be moved, and can be described using a series of Denavit-Hartenberg bases.

In this manuscript, we show how the analytic nature of Zamuner's algorithm makes it possible to implement it efficiently in a language like C, by using a metaprogramming approach, in which the final code is written by another program. In particular, we use Wolfram Mathematica [32] to derive all the equations and write them as C strings. These are then wrapped together by a Python script which builds a C library that can be easily called from any program. While we implement concerted rotations of the ϕ and ψ dihedrals of a protein backbone, other backbone geometries can simply be implemented by

[⋆] Contribution to the Topical Issue "Advances in Computational Methods for Soft Matter Systems" edited by Lorenzo Rovigatti, Flavio Romano, John Russo.

[a] e-mail: `luca.tubiana@univie.ac.at`

changing a few lines of the Mathematica notebook. The Mathematica notebook and the wrapper to generate the code are available on GitHub [33].

Our implementation inherits the advantages of Zamuner's approach while significantly improving its computational speed. In particular, while other methods [23–30] either result in small distortions of the backbone or modify only a selected set of angles, the implementation described here performs a concerted rotation of seven dihedral angles without distorting the backbone, and can be easily extended to work on a mixture of bending and dihedral angles or even on angles and bond lengths together. This possibility allows for applications in the fields of loop refinement, structure prediction under experimental data constraints [34, 35], and backbone reconstruction with structural alphabets [36–38].

The paper is organized as follows. In sect. 2 we recapitulate Zamuner's approach and demonstrate how this can be implemented in Mathematica. We provide details of our implementation, which is available on GitHub [33]. In sect. 3 we show that the obtained moves respect the detailed balance and are numerically stable, and we compute the concerted rotation amplitude that optimizes the sampling efficiency.

2 Methods

2.1 Concerted rotations of protein backbones: Zamuner's approach

In this section, we quickly recap Zamuner's approach to help the reader, and show how the algorithm can be applied to a protein backbone. For a complete treatment see Zamuner *et al.* [31].

A concerted rotation must keep fixed the first and last beads of the chain portion which is moved, and therefore satisfy 6 constraint equations. As a consequence, a series of $n \geq 7$ variables among dihedral angles, bending angles, and bond lengths have to be varied to perform the move. The six constraints thus identify a differential manifold M in a space of dimension n. Both the starting configuration and the final configuration pertains to M. A good strategy to obtain a new configuration ξ_f on M from a given one ξ_o is then to compute the tangent space to M in ξ_o, $T_{\xi_o}(M)$, propose a displacement along it to a point η_f and project this point on M to obtain a new configuration which respects all the constraints. In order to apply this strategy one needs to i) write the constraint equations identifying M in a consistent way, ii) compute the tangent space in any given point of M, and iii) project a point from a tangent space of M to the manifold itself.

2.1.1 Writing the constraint equations

In order to write the constraint equations identifying M we represent the backbone using the Denavit-Hartenberg (DH) convention. Given a polymer $P_N = \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N\}$ we can assign $N' \leq N$ right-handed orthonormal bases $\mathbf{e}_{x,i}$, $\mathbf{e}_{y,i}$, $\mathbf{e}_{z,i}$ to as many bonds of the polymer. This is done as follows: for each basis i we set $\mathbf{e}_{z,i} = \mathbf{b}_i/\|\mathbf{b}_i\|$, where $\mathbf{b}_i = \mathbf{r}_{j(i)+1} - \mathbf{r}_{j(i)}$ and $j(i)$ is the index j of the bond of the chain corresponding to the i-th basis; $\mathbf{e}_{x,i} = \mathbf{e}_{z,i-1} \times \mathbf{e}_{z,i}$, and $\mathbf{e}_{y,i} = \mathbf{e}_{z,i} \times \mathbf{e}_{x,i}$. The orientation of $\mathbf{e}_{x,0}$ and $\mathbf{e}_{y,0}$ for the first basis is arbitrary. The origin of each reference frame $\mathcal{O}_i$, $\mathbf{o}_i$, is set to the atom $\mathbf{r}_{j(i)}$ if $\mathbf{e}_{z,i}$ and $\mathbf{e}_{z,i-1}$ are co-planar, otherwise it is located along $\mathbf{e}_{z,i}$ in correspondence of the point in which the line passing through $\mathbf{e}_{z,i}$ is closer to the line passing through $\mathbf{e}_{z,i-1}$ (identified by the equation $\mathbf{r}_{j(i-1)} + \alpha\mathbf{e}_{x,i-1} = \mathbf{r}_{j(i)} + \beta\mathbf{e}_{z,i}$). Note that two successive bases need not be on consecutive bonds along the polymer.

Having set the bases, we can express a vector $\mathbf{a}_i$ in the reference frame $\mathcal{O}_i$ in the reference $\mathcal{O}_{i-1}$ simply as

$$\mathbf{a}_{i-1} = \mathbf{R}^i_{i-1}\mathbf{a}_i + \mathbf{S}^i_{i-1}, \tag{1}$$

where $\mathbf{R}^i_{i-1}$ is the transformation matrix expressing the basis $\mathbf{e}_{x,i}$, $\mathbf{e}_{y,i}$, $\mathbf{e}_{z,i}$ in $\mathbf{e}_{x,i-1}$, $\mathbf{e}_{y,i-1}$, $\mathbf{e}_{z,i-1}$ and $\mathbf{S}^i_{i-1}$ is the displacement vector giving the position of the origin of $\mathcal{O}_i$ in the reference frame $\mathcal{O}_{i-1}$. If we now define the vector $\mathbf{a}'_i = (\mathbf{a}_i, 1)^T$, and a matrix

$$\mathbf{T}^i_{i-1} = \begin{pmatrix} \mathbf{R}^i_{i-1} & \mathbf{S}^i_{i-1} \\ 0 & 1 \end{pmatrix}, \tag{2}$$

any roto-translation like eq. (1) can be rewritten in a compact form as

$$\mathbf{a}'_{i-1} = \mathbf{T}^i_{i-1}\mathbf{a}'_i. \tag{3}$$

Using this notation one can express a vector in the reference frame $\mathcal{O}_j$ in the frame $\mathcal{O}_i$ as follows:

$$\mathbf{a}'_i = \mathbf{T}^{i+1}_i \mathbf{T}^{i+2}_{i+1} \cdots \mathbf{T}^j_{j-1}\mathbf{a}'_j \equiv \mathbf{T}^j_i \mathbf{a}'_j. \tag{4}$$

Equation (4) can be used to express the constraint that the first and last bonds involved in the concerted rotation remain fixed. This constraint is equivalent to ask that the composed matrices $\mathbf{T}^{n_b+1}_{1,old}$ and $\mathbf{T}^{n_b+1}_{1,new}$, corresponding to the backbone configurations before and after the move, maintain the position and orientation of a vector expressed in the reference frame attached to the last bond when this is expressed in the reference frame corresponding to the first bond:

$$\mathbf{a}'_1 = \mathbf{T}^{n_b+1}_{1,old}\mathbf{a}'_{n_b+1} = \mathbf{T}^{n_b+1}_{1,new}\mathbf{a}'_{n_b+1} \Rightarrow \mathbf{T}^{n_b+1}_{1,old} - \mathbf{T}^{n_b+1}_{1,new} = 0, \tag{5}$$

where 1 and $n_b + 1$ indicate the first and last bond respectively.

In order to proceed, one needs to express eq. (5) in terms of a set of scalar quantities associated with the protein backbone, and in particular as a function of the dihedral angles. The Denavit-Hartenberg (DH) convention provides the instruments to do so, associating 4 parameters to each basis. The rotational part of each matrix $\mathbf{T}^i_{i-1}$, $\mathbf{R}^i_{i-1}$, can be univocally identified by two angles. These are the *link twist*, α_i, defined as the angle between $\mathbf{e}_{z,i}$ and $\mathbf{e}_{z,i-1}$ measured about $\mathbf{e}_{x,i}$; and the *joint angle*, θ_i, between the vectors $\mathbf{e}_{x,i}$ and $\mathbf{e}_{x,i-1}$ measured about

$\mathbf{e}_{z,i-1}$. The translation part of $\mathbf{T}^i_{i-1}$ can be decomposed into one projection along $\mathbf{e}_{z,i}$, r, called *link length* and one on the $\mathbf{e}_{x,i}\mathbf{e}_{y,i}$ plane, d, called *link offset*. Using the DH parameters the matrix $\mathbf{R}^i_{i-1}$ and the vector $\mathbf{S}^i_{i-1}$ can be written explicitly as

$$\mathbf{R}^i_{i-1} = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) \end{pmatrix}, \tag{6}$$

$$\mathbf{S}^i_{i-1} = \left(d_i\cos(\theta_i),\, d_i\sin(\theta_i),\, r_i \right)^T. \tag{7}$$

When the reference frames $\mathcal{O}_i$ and $\mathcal{O}_{i-1}$ are placed on two consecutive bonds the DH parameters assume a well-defined physical meaning: the link twist α_i is the bending angle between the two bonds, θ_i their torsion, r_i is a bond length and $d_i = 0$ since the origin of frame $\mathcal{O}_i$ is placed along $\mathbf{e}_{z,i-1}$. Therefore, if we build a DH base for every bond in the backbone, the torsion angles θ_i will correspond to the different dihedrals of the protein backbone: ϕ, ψ and ω. The link lengths r_i will correspond to the lengths of different N–Cα, Cα–C, C–N bonds and the α_i angles to the angles given by the scalar product between these bonds.

Plugging the DH parameters into eq. (4), one obtains a set of 16 equations in $4N$ parameters, with N the number of DH basis involved in the move. In order to preserve the position and orientation of the last bond we need only 6 equations, specifically those corresponding to the translation part of the matrix $\mathbf{T}_1^{n_b+1}$ to maintain the position and either the upper or lower part of the rotation matrix $\mathbf{R}_1^{n_b+1}$ for the orientation. We chose the upper part as in Zamuner *et al.* The constraint equations are therefore

$$\begin{aligned}
\mathit{\Delta}\mathbf{T}_1(\alpha_{new},\theta_{new},d_{new},r_{new}) &\equiv [\mathbf{T}_1^{n_b+1}(\alpha_{old},\theta_{old},d_{old},r_{old}) \\
&\quad -\mathbf{T}_1^{n_b+1}(\alpha_{new},\theta_{new},d_{new},r_{new})]_{12}, \\
\mathit{\Delta}\mathbf{T}_2(\alpha_{new},\theta_{new},d_{new},r_{new}) &\equiv [\mathbf{T}_1^{n_b+1}(\alpha_{old},\theta_{old},d_{old},r_{old}) \\
&\quad -\mathbf{T}_1^{n_b+1}(\alpha_{new},\theta_{new},d_{new},r_{new})]_{13}, \\
\mathit{\Delta}\mathbf{T}_3(\alpha_{new},\theta_{new},d_{new},r_{new}) &\equiv [\mathbf{T}_1^{n_b+1}(\alpha_{old},\theta_{old},d_{old},r_{old}) \\
&\quad -\mathbf{T}_1^{n_b+1}(\alpha_{new},\theta_{new},d_{new},r_{new})]_{23}, \\
\mathit{\Delta}\mathbf{T}_4(\alpha_{new},\theta_{new},d_{new},r_{new}) &\equiv \mathbf{T}_1^{n_b+1}(\alpha_{old},\theta_{old},d_{old},r_{old}) \\
&\quad -\mathbf{T}_1^{n_b+1}(\alpha_{new},\theta_{new},d_{new},r_{new})]_{14}, \\
\mathit{\Delta}\mathbf{T}_5(\alpha_{new},\theta_{new},d_{new},r_{new}) &\equiv \mathbf{T}_1^{n_b+1}(\alpha_{old},\theta_{old},d_{old},r_{old}) \\
&\quad -\mathbf{T}_1^{n_b+1}(\alpha_{new},\theta_{new},d_{new},r_{new})]_{24}, \\
\mathit{\Delta}\mathbf{T}_6(\alpha_{new},\theta_{new},d_{new},r_{new}) &\equiv [\mathbf{T}_1^{n_b+1}(\alpha_{old},\theta_{old},d_{old},r_{old}) \\
&\quad -\mathbf{T}_1^{n_b+1}(\alpha_{new},\theta_{new},d_{new},r_{new})]_{34},
\end{aligned} \tag{8}$$

where we indicate with α, θ, d, r the whole set of $4N$ parameters. When there are $n \geq 7$ free parameters the constraint equations can be solved.

In coarse-grained protein models only the ϕ and ψ dihedrals are considered to be free, while the dihedral angle ω of the peptidic bond as well as all bond lengths and bending angles are fixed. Taking into account this aspect, we build a concerted rotation which moves 3 consecutive residues by varying 7 consecutive dihedral angles ϕ and ψ. This can be achieved by attaching 7 DH reference frames to consecutive N–Cα and Cα–C bonds, as depicted in fig. 1, so that their joint angles θ_i will correspond to $\phi_i + \pi$ and ψ_i respectively.

Since the other backbone quantities are always fixed, we end up with two transformation matrices, one for ϕ, in which $d_\phi = 0$ and r_ϕ is given by the length of the bond N–Cα, so that $\mathbf{S}_\phi = \{0, 0, r_\phi\}$; and the other one for ψ, in which the translation vector $\mathbf{S}_\psi = \mathbf{S}_{\mathrm{N}}^{\mathrm{C}\alpha}$ is computed as described above, either at run-time, to improve numerical stability, or once and for all, to optimize for execution speed. The link twist for the ψ matrix is simply $\alpha_\psi = \pi - \widehat{\mathrm{NC\alpha C}}$, while the one for the ϕ matrix can be easily computed from the peptide geometry.

Using the matrices $\mathbf{T}_1(\phi)$ and $\mathbf{T}_2(\psi)$ we can thus impose a constraint, for example by starting from a ϕ dihedral and vary 7 consecutive dihedral angles:

$$\mathit{\Delta}(\mathbf{T}_1\mathbf{T}_2\mathbf{T}_1\mathbf{T}_2\mathbf{T}_1\mathbf{T}_2\mathbf{T}_1) = 0. \tag{9}$$

The matrix product as well as the other equations can easily be computed analytically with Mathematica, as shown in sect. 2.2.

2.1.2 Identifying the tangent space and performing the move

The tangent space to the manifold M is given by the kernel of the application $\boldsymbol{\mathit{\Delta}}\mathbf{T}(\alpha_{new},\theta_{new},d_{new},r_{new})$: $R^n \to R^6$. For the constraint in eqs. (9), $n = 7$ and M is a one-dimensional manifold in R^7. Recalling Dini's implicit function theorem, we can identify the tangent space by picking one dihedral angle at random as the free variable and expressing the other six as a function of this one. If we indicate with $\boldsymbol{y}$ the constrained variables and with x the free one, $\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}x}(\xi)$ can be expressed in any non-singular point as

$$\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}x}(\xi) = J^{-1}(\xi)\frac{\partial\boldsymbol{\mathit{\Delta}}\mathbf{T}}{\partial x}, \tag{10}$$

where ξ is the point of the manifold, J^{-1} is the inverse of the Jacobian $J = \frac{\partial\boldsymbol{\mathit{\Delta}}\mathbf{T}(\xi)}{\partial\mathbf{y}}$ and $\boldsymbol{\mathit{\Delta}}\mathbf{T}$ the 6-dimensional constraint function. From this relation the tangent vector to $M(\xi)$ in R^7, $\boldsymbol{\nu}(\xi)$, can be simply written with the constrained components given by relation (10) and the component corresponding to the free variable set to 1. Of course, in order to apply eq. (10), the determinant of J must be different from zero. In that case the approach of Zamuner *et al.*, followed here, is to simply take a different angle as the free variable.

Numerically, it is particularly demanding to invert a matrix. Therefore to obtain $\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}x}(\xi)$ we prefer to solve the corresponding linear system given by

$$\frac{\partial\boldsymbol{\mathit{\Delta}}\mathbf{T}}{\partial x} = J(\xi)\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}x}(\xi). \tag{11}$$

The move proper consists in choosing a free variable, computing the tangent vector $\boldsymbol{\eta}(\xi_o)$ in the original point

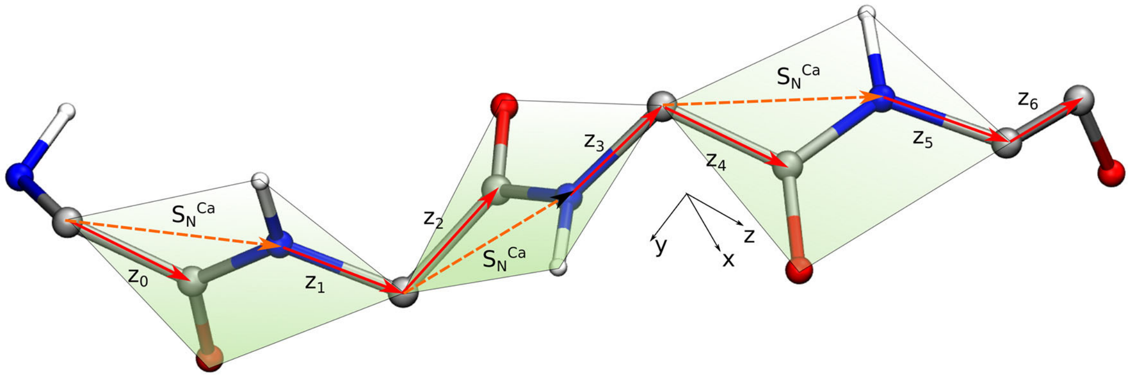

Fig. 1. Peptide bonds involved in a local concerted rotation. The bonds involved in the move are marked in red. Those going from Cα to N atoms correspond to ϕ dihedral angles. Those going from C to Cα to ψ angles. All the distances and angles inside a peptide are considered fixed. The vectors **S** joining to successive origins $\mathbf{o}_i$ and $\mathbf{o}_{i+1}$ are reported when they are not zero.

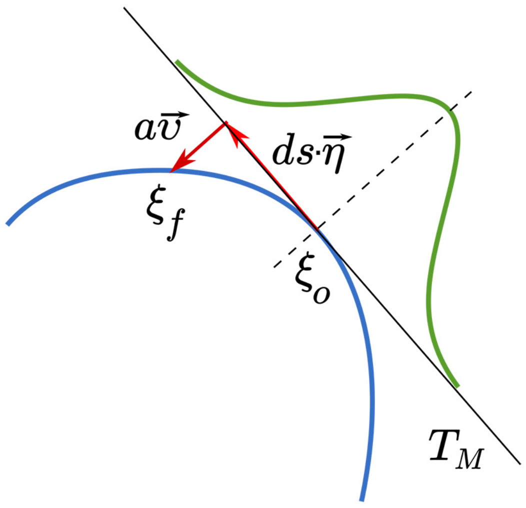

Fig. 2. Concerted rotations are mapped by Zamuner's approach to a random step on the tangent space and a root-finding step to find a new point on the constraints manifold. The amplitude of the random step, ds, is picked from a Gaussian distribution of mean zero and variance σ. As noted by Zamuner *et al.* [31], when the step ds is too large the root-finding algorithm might not converge, as the point $\xi_o + \mathrm{d}s\boldsymbol{\eta}$ becomes too far from the manifold.

and projecting along this direction by a random amount, ds, that controls the amplitude of the move. The new point ξ_f, identifying the proposed configuration is obtained from the point $\xi_o + \mathrm{d}s\boldsymbol{\eta}(\xi_o)$ with a root-finding scheme in the space orthogonal to $T_M(\xi_o)$. This procedure is reported schematically in fig. 2.

In our implementation, we use a simple Gram-Schmidt algorithm to build a base for the orthogonal space and then use the multiroot package of the GSL library [39] to find a root of the equation $\boldsymbol{\Delta}\mathbf{T}(\xi_o + \mathrm{d}s\boldsymbol{\eta}(\xi_o) + a\boldsymbol{\nu}) = 0$, with $\boldsymbol{\nu}$ in the orthogonal space to $T_M(\xi_o)$.

Following Zamuner *et al.* [31], we take the value of ds from a Gaussian distribution of width σ. This width thus controls the amplitude of the move. Its effects are discussed in detail in sect. 3.

In order to weight correctly the various configurations we need to take into account the determinant of the Jacobian in both the original point ξ_o and the final point ξ_f, as well as the probability of extracting the step sizes ds and ds' for the forward and backward move from a Gaussian distribution. The new configuration has therefore to be accepted with probability

$$p = \min\left(1, \exp\left(\frac{\mathrm{d}s^2 - \mathrm{d}s'^2}{\sigma^2}\right)\frac{|J(\xi_f)|}{|J(\xi_o)|}\right), \qquad (12)$$

where $|J|$ indicates the determinant of the Jacobian. In principle, the approach of Zamuner *et al.* can be extended to avoid this computation, by re-weighting the variables involved in the move so that they already appear with the correct probability. In our case though, all constraints, Jacobians, and derivatives, are precomputed using Mathematica, so the Jacobians calculation presents no computational overhead.

2.2 Metaprogramming: using Mathematica to pre-generate the code

The mathematical steps needed to compute the constraint equation, its partial derivatives, and its Jacobian can all be automated with a software like Mathematica. Using Mathematica, we compute symbolically the function $\boldsymbol{\Delta}\mathbf{T}$, (8), its partial derivatives with respect to each dihedral angle, and the seven Jacobians obtained by fixing each different dihedral. In total, this approach gives a set of polynomial equations in $\cos(\phi_i)$, $\sin(\phi_i)$, $\cos(\psi_i)$ and $\sin(\psi_i)$.

All equations are processed using Mathematica `HornerForm` function to optimize their expression for numerical evaluation. After this simplification sines and cosines are substituted with generic variables and the resulting equations are printed as C strings in separate text files.

Each text file is then read by a pre-processor, a Python script, which wraps a function around each equation. These functions take in input the sines and cosines of the angles as well as the other DH parameters and substitute

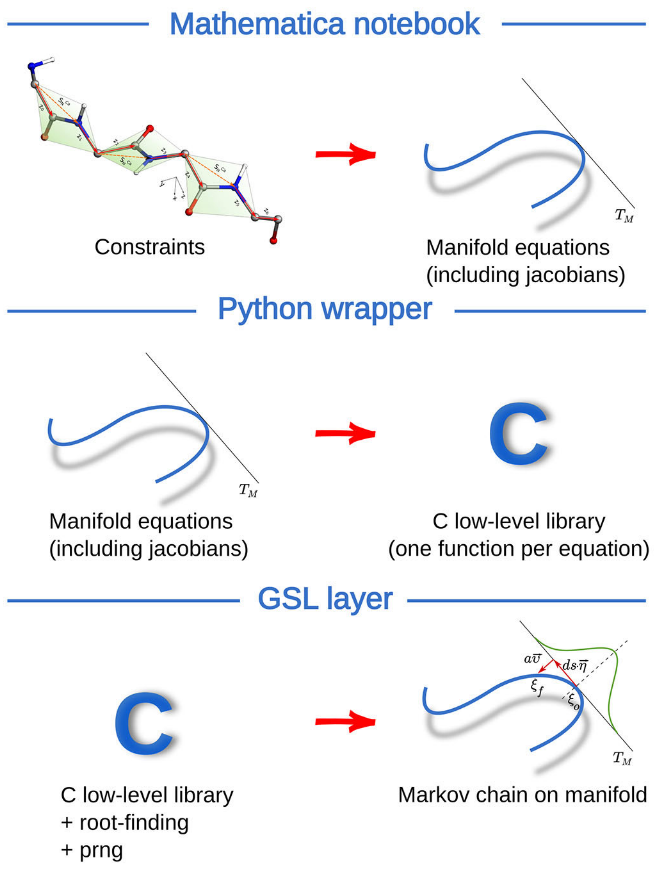
Mathematica notebook
Constraints
Manifold equations (including jacobians)
T_M
Python wrapper
Manifold equations (including jacobians)
C
C low-level library (one function per equation)
GSL layer
C
C low-level library + root-finding + prng
$a\vec{v}$
$ds\vec{\eta}$
ξ_f
ξ_o
Markov chain on manifold

Fig. 3. A schematic representation of our metaprogramming methodology. First, the Mathematica notebook is used to produce a set of equations representing the constraints manifold. In the second step, the Python wrapper encapsulates each equation in a function and create a C library. Finally, the GSL layer calls these equations to implement Zamuner's algorithm, providing an effective Markov chain on the constraints manifold, see fig. 2.

them in the equation, whose value is returned. The preprocessor also creates the interface for the whole concerted rotation move.

These equations, put together by the Python script, immediately give us all the building blocks we need to write eqs. (9) and (11). In order to solve eq. (11) we then use the linear solver algorithms from GSL. The procedure is schematized in fig. 3. The implementation can be downloaded from GitHub [33].

We note here that while our approach is dedicated to concerted rotations of protein backbone dihedrals, this methodology can easily be applied to other cases. In particular, the user can simply change the free variables in the Mathematica notebook provided on the repository [33] to implement a move based on bending angles or a mixture of bending and dihedral angles. The generality of the implementation also makes it possible to modify 3 non-consecutive peptides, as described by Zamuner *et al.* [31]. In order to do so, it is sufficient to substitute the functions that convert the protein backbone to DH parameter and viceversa to deal with the more general case. On the other hand, the approach can be specialized. Instead of using the general matrix $\mathbf{T}$ and insert the DH parameters at runtime, one can already fix all constant parameters from the beginning in Mathematica, thus saving numerical operations. For example for a protein one could input $\mathbf{T}_1(\phi)$ and $\mathbf{T}_2(\psi)$ in Mathematica instead of using the general matrix as we do here.

3 Results

We tested the robustness and speed of our implementation by simulating a phantom 20 amino-acid long protein backbone. To sample its configurations we used a combination of concerted rotations and pivot moves to equilibrate the termini of the chain. Pivot moves were performed only on the first and last 3 amino-acids. As can be seen from fig. 4(a), this combination of moves samples uniformly the dihedrals, as expected. Using a reconstruction algorithm based on the dihedral angles to move the affected portion of the backbone, local concerted rotations remain numerically accurate for at least 10^6 iterations, at which point there is usually a variation of some bond lengths of the order 10^{-9} nm.

As described in the methods, the amplitude of the move depends on the length of the step along the tangent space, ds, which is chosen from a Gaussian distribution of width σ. Larger values of σ thus correspond intuitively to larger variations between the original set of dihedrals and the new one. At the same time, a larger σ causes the pre-rotated point to be further from the constraints manifold. This increases the number of iterations required to find a solution as well as the probability of rejecting the move, either because of the detailed balance or due to a lack of convergence of the root-finding algorithm.

It is particularly interesting to characterize the impact of σ on the efficiency of concerted rotations. To do so, we first measured its effect on the move speed and on its rejection rate. To have a clean measure of the algorithm's execution time, we worked directly in dihedral space. The results, reported in fig. 4(b), show that both quantities grow quickly for σ between 0.05 and 2, while for $\sigma > 2$ the growth slows down. In this interval the time required to perform the move on an Intel Xeon 2.3 GhZ processor goes from 0.37 ms for $\sigma = 0.05$ to 2.5 ms for $\sigma = 2.15$, while the rejection rate goes from 27% to 43%. We can combine these data to obtain the number of successful concerted moves per second. This is reported in fig. 4(c), and shows that the move is particularly efficient for values of σ lower than 0.5, with more than a thousand successful rotations per seconds.

We then proceeded to assess the impact of σ on the ability of the move to sample the dihedral space. To do so, we considered again a phantom chain of 20 amino acids and measured its cumulative Ramachandran plot, excluding the first and last residue. As observed before, at equilibrium all dihedrals are equiprobable and thus the Ramachandran plot must correspond to a uniform distribution; one can thus estimate the sampling efficiency of

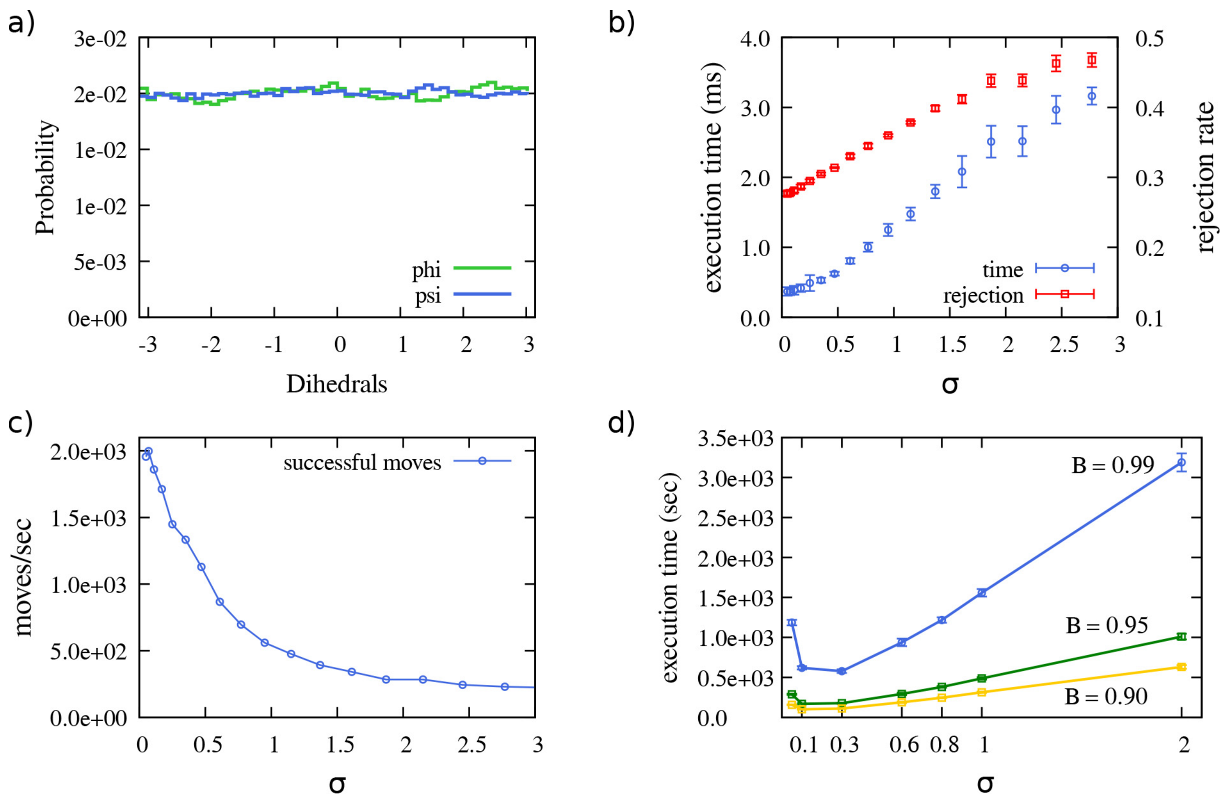

Fig. 4. (a) Cumulative histograms for the dihedrals of a 20 a.a. long peptide without steric hindrance, moved using concerted rotations and pivot moves for the ends, for 5 million steps with $\sigma = 0.1$. (b) Average execution time and rejection rate for concerted rotations as a function of σ, the parameter controlling the amplitude of concerted rotations, derived from five independent simulations. (c) Number of successful concerted rotations per second as a function of σ. (d) Time needed to reach a given value of the Bhattacharyya coefficient as a function of σ.

concerted rotations by measuring the number of iterations required to reach a uniform ϕ, ψ distribution.

To quantify the convergence of our Ramachandran plot we used the Bhattacharyya coefficient [40], which measures the discrepancy between two normalized histograms. It is defined as

$$B = \sum_{i=1}^{N} \sqrt{q_i p_i}, \tag{13}$$

where the sum runs over all the N bins and q_i, p_i are the values of bin i in the two histograms. If the two histograms are identical, $B = 1$. The reason for this can be understood intuitively through geometry, as originally noted by Bhattacharyya. Since the two histograms are normalized,

$$\sum_{i=1}^{N} p_i = 1$$

for both of them. The square roots of the terms in the sum, $\sqrt{p_i}$, can thus be considered as the components of a normalized vector on an orthonormal basis of size N. The coefficient B is therefore equal to the scalar product between two N-dimensional unit vectors, and is equal to one when they are collinear [40].

In our case, we calculated B between our Ramachandran plot and a uniform distribution, for which $p_i = M/N$ with M the number of measures performed. For all the values of σ we tested B converges to unity, although with different times (see appendix A). We can thus use B to identify an optimal value of σ to use in simulations, by fixing a threshold for B and measuring the time needed to reach it as a function of σ. The results are reported in fig. 4(d) for three different thresholds, $B = 0.9, 0.95, 0.99$. All three curves display a minimum between $\sigma = 0.1$ and $\sigma = 0.3$. Increasing σ beyond this value causes a significant growth in the simulation time required to explore the configurational space of a phantom backbone.

Finally, we investigated whether the optimal value of σ changes when taking into account a more realistic peptide backbone with excluded-volume interactions. To do so, we implemented a simple self-avoiding peptide chain, 20 a.a. long, with excluded-volume interactions between oxygen and amide hydrogen. Both oxygen and hydrogen were represented as hard spheres with a radius derived from their Van der Waals radii. To ensure the peptide chain impenetrability we introduced an additional excluded-volume interaction between α carbons with their diameter scaled up to 0.4 nm, omitting next-neighbor interactions[1].

[1] Due to the backbone reconstruction algorithm implemented in our test code, the NCαC angle can be susceptible to the cumulation of numerical errors which can amount, in the presence of steric hindrance, to a maximum of ~ 0.1 degrees

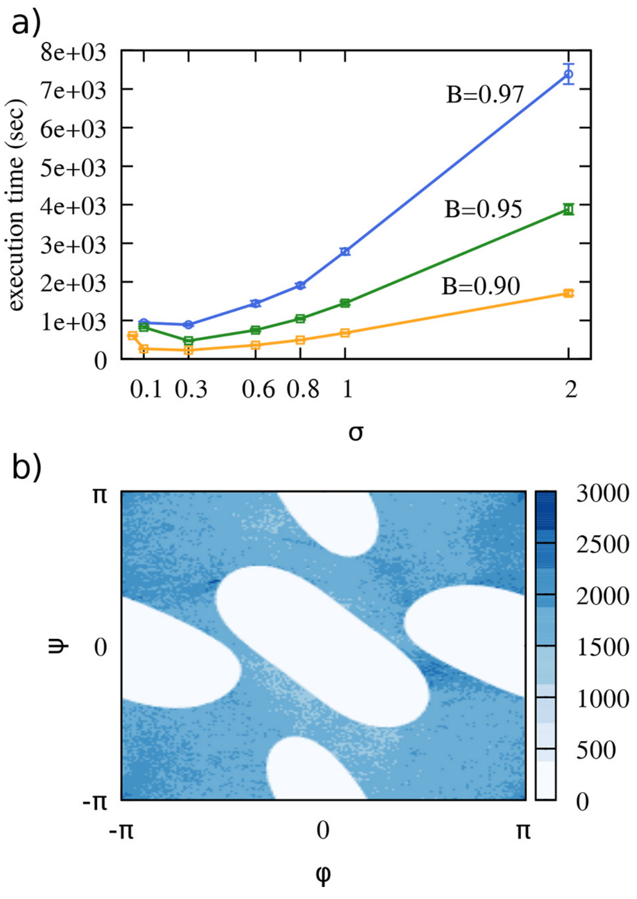

Fig. 5. (a) Optimal σ for a polypeptide chain, taking into account the excluded volume. The convergence of the coefficient B is further discussed in appendix A. (b) Histogram of Ramachandran plot for a 20 a.a. long poly-peptide with the steric hindrance of oxygen, hydrogen and Cα atoms after 10^8 iterations for $\sigma = 0.3$.

To calculate the optimal σ in this case, we built a reference probability distribution against which we could compute the Bhattacharyya coefficient using eq. (13). We did so by running a long simulation (10^8 steps) using only pivot moves and saving the Ramachandran plot for all dihedrals in the chain in a 180×180 bins histogram. We considered a bin to be accessible if it is visited at least 5 times (to account for possible numerical uncertainties). From this histogram, we produced an ideal normalized distribution by assigning to each accessible bin the probability $\frac{1}{number\ of\ accessible\ bins}$ and 0 to all unaccessible bins.

Despite a notable change in the peptide model, the optimal value for σ remains approximately in the same range from 0.1 to 0.3, although slightly shifted toward 0.3 (see fig. 5(a)). On the other hand, the overall performance of the move in constrained space has decreased both due to a rejection of configurations in excluded areas as well as to the time required to compute the excluded-volume constraints. The Ramachandran plot obtained for $B = 0.98$ is reported in fig. 5(b).

after 10^6 moves. While this deviation is well within the statistical uncertainty over these bending angles, to compute the Ramachandran plot we enforced their value to be 111° through a harmonic potential.

4 Conclusions

In this manuscript, we showed how the code for concerted rotations can be written using a metaprogramming approach, in which a symbolic calculation package like Wolfram Mathematica precomputes the analytic part of the move. The resulting equations are printed in C format and then arranged in a set of usable C files by a Python script.

To guide the adoption of our move for Monte Carlo simulations of polypeptide chains, we estimated the optimal value of the concerted rotations step, σ, which we found to be between 0.1 and 0.3 irrespective of steric interactions. We note that other applications, such as loop refinement, might however benefit from different, or adaptive, choices of σ, based on the distance between the ends of the loop to be refined or the amount of distortions to be corrected.

Our implementation is very efficient as most of the calculation have already been performed symbolically. Furthermore, the speed of the move can be increased by adding known backbone constraint into the equations in Mathematica, thus saving operations at the price of a lower flexibility of the produced code. For example, the angles and bond lengths for the protein backbone can be fixed already in the symbolic equations. An example Mathematica notebook for concerted rotations of protein backbones together with the generated code and series of small test programs is available on GitHub [33].

Open Access funding provided by University of Vienna. LT acknowledges support from the Mahlke-Oberman Stiftung and the European Union's Seventh Framework Programme for research, technological development and demonstration (grant No. 609431). MJ acknowledges support from the Czech Science Foundation (grant 17-11571S) and the CEITEC 2020 (LQ1601) project with financial contribution made by the Ministry of Education, Youths and Sports of the Czech Republic within special support paid from the National Programme for Sustainability II funds. Computational resources were provided by the CESNET LM2015042 and the CERIT Scientific Cloud LM2015085, provided under the programme Projects of Large Research, Development, and Innovations Infrastructures. This project has received funding from the European Union's Horizon 2020 research and innovation Programme under grant agreement No. 692068.

Author contribution statement

LT designed the approach using Mathematica to write C code, and wrote the first implementation. LT and MJ developed and tested the C implementation. MJ performed all analyses on the move efficiency. All the authors were involved in the preparation of the manuscript. All the authors have read and approved the final manuscript.

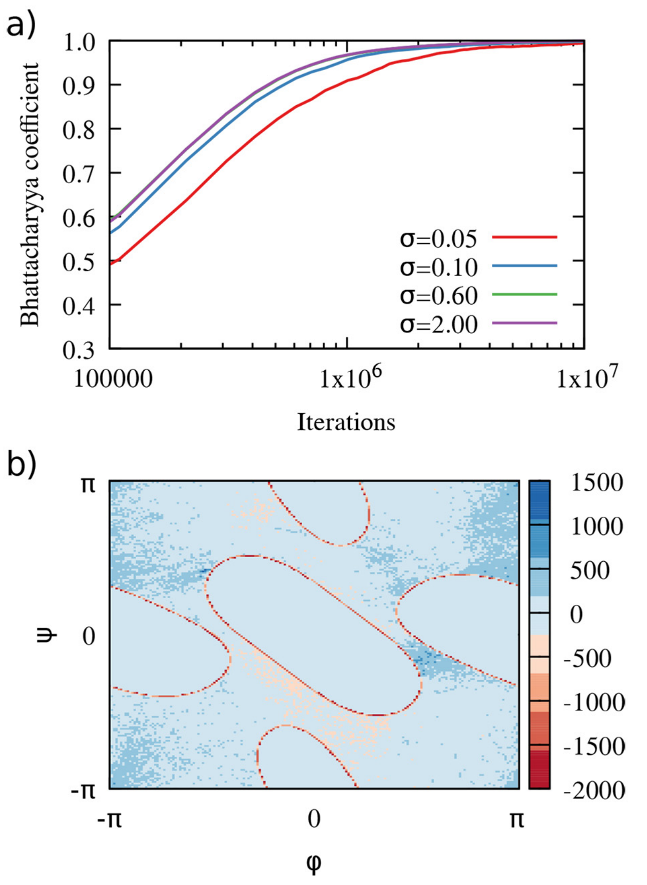

Fig. 6. (a) Convergence of B for different values of σ for phantom polypeptide chains. (b) Difference between the Ramachandran histogram and the corresponding reference histogram for a 20 a.a. long polypeptide with steric hindrance of oxygen, hydrogen and Cα atoms, after 10^8 iterations for $\sigma = 0.3$.

Appendix A. Convergence of Bhattacharyya coefficients

In this appendix, we quickly discuss the convergence of the Bhattacharyya coefficients in the two cases discussed in the main text. For phantom polypeptide chains, the coefficient B always converges to 1, for all values of σ. The time needed to reach convergence is reported in fig. 6(a).

When taking into account a more realistic polypeptide chain, including steric interactions between alpha carbons, hydrogens, and oxygens, we noticed that $B = 0.98$ after 10^8 MC steps. The inability to reach $B = 1$ is due to the fact that our reference distribution is only approximated. Being a finite histogram, it is, in fact, a rasterization of the true Ramachandran plot. We report in fig. 6 the difference between the reference distribution and the distribution obtained at $B = 0.98$ using concerted rotations and pivot moves of the chain ends. We notice that the great majority of the discrepancies are distributed along the border between accessible and unaccessible regions, since the corresponding bins are difficult to assign to either of them.

References

1. Peter Tompa, Trends Biochem. Sci. **27**, 527 (2002).
2. H. Jane Dyson, Peter E. Wright, Nat. Rev. Mol. Cell Biol. **6**, 197 (2005).
3. Anthony L. Fink, Curr. Opin. Struct. Biol. **15**, 35 (2005).
4. Vladimir N. Uversky, A. Keith Dunker, Biochim. Biophys. Acta-Proteins Proteomics **1804**, 1231 (2010).
5. Predrag Kukic, Arvind Kannan, Maurits J.J. Dijkstra, Sanne Abeln, Carlo Camilloni, Michele Vendruscolo, PLoS Comput. Biol. **11**, e1004435 (2015).
6. Valentina Tozzini, Curr. Opin. Struct. Biol. **15**, 144 (2005).
7. Cecilia Clementi, Curr. Opin. Struct. Biol. **18**, 10 (2008).
8. Paul C. Whitford, Jeffrey K. Noel, Shachi Gosavi, Alexander Schug, Kevin Y. Sanbonmatsu, Jose N. Onuchic, Proteins: Struct. Funct. Bioinform. **75**, 430 (2009).
9. Adam Liwo, Stanisław Ołdziej, Cezary Czaplewski, Dana S. Kleinerman, Philip Blood, Harold A. Scheraga, J. Chem. Theory Comput. **6**, 890 (2010).
10. Gregory R. Bowman, Vincent A. Voelz, Vijay S. Pande, Curr. Opin. Struct. Biol. **21**, 4 (2011).
11. Ivan Coluzza, PLoS ONE **6**, e20853 (2011).
12. M.M. Lin, A.H. Zewail, Ann. Phys. **524**, 379 (2012).
13. W.G. Noid, J. Chem. Phys. **139**, 090901 (2013).
14. James T. MacDonald, Lawrence A. Kelley, Paul S. Freemont, PLoS ONE **8**, e65770 (2013).
15. Ivan Coluzza, PLoS ONE **9**, e112852 (2014).
16. Ivan Coluzza, J. Phys.: Condens. Matter **29**, 143001 (2017).
17. B. Mehlig, A.L.C. Ferreira, D.W. Heermann, Phys. Lett. B **291**, 151 (1992).
18. B.M. Forrest, U.W. Suter, J. Chem. Phys. **101**, 2616 (1994).
19. Dmitry G. Gromov, Juan J. de Pablo, J. Chem. Phys. **103**, 8247 (1995).
20. Anne Voegler Smith, Carol K. Hall, Proteins: Struct. Funct. Genet. **44**, 344 (2001).
21. F. Calvo, J.P.K. Doye, Phys. Rev. E **63**, 0109021 (2001).
22. L.R. Dodd, T.D. Boone, D.N. Theodorou, Mol. Phys. **78**, 961 (1993).
23. Daniel Hoffmann, Ernst-Walter Knapp, Eur. Biophys. J. **24**, 387 (1996).
24. G. Favrin, A. Irbäck, F. Sjunnesson, J. Chem. Phys. **114**, 8154 (2001).
25. Jakob P. Ulmschneider, William L. Jorgensen, J. Chem. Phys. **118**, 4261 (2003).
26. Evangelos A. Coutsias, Chaok Seok, Matthew P. Jacobson, Ken A. Dill, J. Comput. Chem. **25**, 510 (2004).
27. Marcos R. Betancourt, J. Chem. Phys. **123**, 174905 (2005).
28. Daniel J. Mandell, Evangelos A. Coutsias, Tanja Kortemme, Nat. Methods **6**, 551 (2009).
29. Sandro Bottaro, Wouter Boomsma, Kristoffer E. Johansson, Christian Andreetta, Thomas Hamelryck, Jesper Ferkinghoff-Borg, J. Chem. Theory Comput. **8**, 695 (2012).
30. Amelie Stein, Tanja Kortemme, PLoS ONE **8**, e63090 (2013).
31. Stefano Zamuner, Alex Rodriguez, Flavio Seno, Antonio Trovato, PLoS ONE **10**, e0118342 (2015).

32. Mathematica, Version 9.0, Wolfram Reasearch Inc., Champaign, IL, 2018.
33. Luca Tubiana, Miroslav Jurasek, luca-tubiana/concerted_rotations: `https://doi.org/10.5281/zenodo.1183330` (February 2018).
34. Andrea Cavalli, Xavier Salvatella, Christopher M. Dobson, Michele Vendruscolo, Proc. Natl. Acad. Sci. U.S.A. **104**, 9615 (2007).
35. Andrea Cavalli, Carlo Camilloni, Michele Vendruscolo, J. Chem. Phys. **138**, 03B603 (2013).
36. T. Alwyn Jones, Soren Thirup, EMBO J. **5**, 819 (1986).
37. Cristian Micheletti, Flavio Seno, Amos Maritan, Proteins: Struct. Funct. Bioinform. **40**, 662 (2000).
38. Alessandro Pandini *et al.*, BMC Bioinform. **11**, 97 (2010).
39. Mark Galassi, Jim Davies, James Theiler, Brian Gough, Gerard Jungman, Patrick Alken, Michael Booth, Fabrice Rossi, *GNU Scientific Library Reference Manual* (Network Theory Ltd., 2001).
40. Frank J. Aherne, Neil A. Thacker, Peter I. Rockett, Kybernetika **34**, 363 (1998).