Part of the supporting material of: Sequence-dependent persistence lengths of DNA (Mitchell, Glowacki, Grandchamp, Manning, and Maddocks)

J. of Chemical Theory and Computation (2016)
DOI: 10.1021/acs.jctc.6b00904

S.1 Computation of coarse-grain approximations to the tangent

The unit tangents $\mathbf{t}_i^{[k]}$ (where for simplicity we only take k odd) to the straight lines that are the best least squares linear approximation to a consecutive run of $(k+1)$ base-pair locations $\mathbf{r}_i, \ldots, \mathbf{r}_{i+k}$ can be computed for any configuration, $\mathbf{t}_i^{[k]}$ as follows. First calculate the (geometrical) centre of mass $\mathbf{c}_i^k = (\sum_{j=i}^{j=i+k} \mathbf{r}_j)/(k+1)$. Then $\mathbf{t}_i^{[k]}$ is the unit eigenvector (with positive projection on the chord $\mathbf{r}_{i+k} - \mathbf{r}_i$) corresponding to the largest eigenvalue of the (local gyration) matrix $\sum_{j=i}^{j=i+k} (\mathbf{r}_j - \mathbf{c}_i^k) \otimes (\mathbf{r}_j - \mathbf{c}_i^k)$. The case $k=1$ reduces analytically to the unit tangent to the junction chord between two consecutive base pair origins $\mathbf{t}_i^{[1]} = (\mathbf{r}_{i+1} - \mathbf{r}_i)/\left\|\mathbf{r}_{i+1} - \mathbf{r}_i\right\|$, while nonlocal coarse grain choices $k > 1$ must be computed numerically.

S.2 Details regarding the cgDNAmc code

S.2.1 Downloading the software

The C++ code *cgDNAmc*, along with two libraries it depends upon, *algebra3d* and *cgDNArecon*, is freely available with online instructions on how to download, compile, and run it.[1] The user has to supply any desired problem-specific, post-processing code fragments implementing specialised techniques such as the sliding-window average used in modelling cryo-EM experimental data.

The remainder of this section describes our Monte Carlo implementation in further detail. The simulations described here are not particularly intensive, nevertheless we have taken some efforts to make *cgDNAmc* code efficient. Benchmark results presented below were obtained on a mid-range laptop computer.

S.2.2 Direct Monte Carlo sampling

As described in the main text, a key step in our direct Monte Carlo sampling is the Cholesky decomposition $\mathbf{K} = \mathbf{L}\mathbf{L}^T$ of the sparse stiffness matrix $\mathbf{K}$. For efficient sampling, the key property of the Cholesky factorization is that if $\mathbf{K}$ has bandwidth m (meaning that all nonzero entries are within m rows of the diagonal, so for us $m = 17$), then $\mathbf{L}$ also has bandwidth m After this step, a new energy $E(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T\mathbf{y}$ with $\mathbf{y} = \mathbf{L}^T(\mathbf{w} - \hat{\mathbf{w}})$ yields a probability density function on $\mathbf{y}$ that is the product of uncoupled univariate normal distributions:

$$p_{\mathbf{y}}(\mathbf{y}) = \prod_{i=1}^{12n-6} \left(\frac{\beta}{2\pi}\right)^{\frac{1}{2}} e^{-\frac{\beta}{2}\mathrm{y}_i^2}. \tag{S.1}$$

To make a single draw $\mathbf{y}$ from this distribution, each component y_i is taken as a random number from the normal distribution with mean 0 and standard deviation $\beta^{-\frac{1}{2}}$. Note that units of the stiffness matrix $\mathbf{K}$ in the cgDNA model are such that $\beta = 1$. For the sake of efficiency uniform deviates are generated using the `xorshift1024*` implementation[2] of the `xorshift` algorithm to normal deviates using the `ZIGNOR` implementation[3] of the Ziggurat algorithm

The draw of the internal coordinates $\mathbf{w}$ corresponding to $\mathbf{y}$ is obtained from the equation $\mathbf{y} = \mathbf{L}^T(\mathbf{w} - \hat{\mathbf{w}})$ by solving $\mathbf{L}^T\mathbf{z} = \mathbf{y}$ for $\mathbf{z}$ (taking advantage of the upper triangular, banded structure of $\mathbf{L}$ using an appropriate solver from LAPACK and then setting $\mathbf{w} = \mathbf{z} + \hat{\mathbf{w}}$.

An alternative approach to obtain direct sampling would involve a spectral decomposition of $\mathbf{K}$ in place of the Cholesky factorisation, i.e.

$$\mathbf{K} = \mathbf{P}\mathbf{D}\mathbf{P}^T, \tag{S.2}$$

[1] see `http://lcvmwww.epfl.ch/cgDNA`
[2] `http://arxiv.org/abs/1404.0390`
[3] http://www.doornik.com/research/ziggurat.pdf

with $\mathbf{P}$ orthogonal and $\mathbf{D}$ diagonal. Here a similar change of variable $\mathbf{y} = \mathbf{D}^{\frac{1}{2}}\mathbf{P}^T\mathbf{w}$ can be used so that $\mathbf{w} = \mathbf{P}\mathbf{D}^{-\frac{1}{2}}\mathbf{y}$. This has been successfully exploited by Czapla et al. for the case where $\mathbf{K}$ is block diagonal. However in our setting with a (potentially large) banded $\mathbf{K}$ that approach is significantly less efficient, since the matrix $\mathbf{P}\mathbf{D}^{\frac{1}{2}}$ would not be sparse, and a dense matrix-vector multiply must be carried out in the construction of each draw. To give an example a simulation calculating $\langle \mathbf{t}_i^{[0]} \cdot \mathbf{t}_0^{[0]} \rangle$ for 1 million configurations of the λ_3 sequence of length 300 bp using Cholesky decomposition takes just above 3 minutes, while using spectral decomposition the running time is around 2 hours.

S.2.3 Rigid base pair marginals

We remark that many expectations of interest involve only the *inter* part of the configuration variable $\mathbf{w}$ so that the number of degrees of freedom can be reduced by one half by computing the marginal distribution for the inter variables. As the original distribution is Gaussian its marginals are also Gaussian, but the resulting marginal stiffness matrix is now dense, so that sparse computations can no longer be used. As a consequence a calculation of $\langle \mathbf{t}_0^{[0]} \cdot \mathbf{t}_i^{[0]} \rangle$ for 1 million configurations of the λ_3 fragment using the marginal distribution takes around 23 minutes, nearly 7 times slower than generating ensembles in the full $\mathbf{w}$ space and discarding all the *intra* variables.

S.2.4 Reconstruction of 3D shapes

The first step in calculating our observables is reconstructing a 3D shape of a molecule from a given internal coordinate vector $\mathbf{w}$ as detailed in As mentioned in the previous section, the calculation of tangent-tangent correlations, arclengths and Flory vectors require only the *inter* part of $\mathbf{w}$. As a result we only reconstruct base pair positions $\mathbf{r}_i$ and orientations $\mathbf{R}_i$, which takes only half the time of reconstructing a full 3D configuration of rigid bases. The reconstruction procedure, implemented by the *cgDNArecon* library, involves evaluating half rotations, composing rotations, applying rotations to vectors and adding vectors.

A careful numerical study of efficiency of different parametrisations of rotations (namely Cayley vectors, unit quaternions and rotation matrices) using the *algebra3d* library has been performed. An explicit half-rotation formula for unit quaternions proved to be 60% faster than a similar formula for Cayley vectors. (For rotation matrices, the analogous calculation would require, e.g., an iterative algorithm of computing the principal square root and so was not considered). As expected, for composition of rotations, quaternion multiplication was faster than matrix multiplication, with our observed difference being 20%. On the other hand, in the case of applying a rotation to a vector, the matrix-vector product was 5 times faster than a specialised quaternion multiplication. In fact the fastest way to apply a rotation given as unit quaternion to a vector was to convert the quaternion to a rotation matrix first (this takes only twice the time of the matrix-vector product). Efficiency of converting between all three parametrisations was also analysed. This suggested, for example, that the formula for computing a rotation matrix for a given Cayley vector of times slower than conversion of a Cayley vector to quaternion and subsequent conversion of the quaternion to a rotation matrix.

Considerations similar to the above suggested two approaches to the reconstruction procedure. The first one uses directly the Cayley vectors of the configuration variable $\mathbf{w}$ to calculate half rotations and converts to rotation matrices for all subsequent calculations. The other one, that finally proved to be 30% faster, begins with converting the Cayley vectors to quaternions, then computes half rotations using quaternions, and finally converts quaternions to matrices when rotations need to be applied to vectors.

S.2.5 Remarks on parallelisation

We first note that in *cgDNAmc* pseudo-random numbers are generated sequentially to ensure reproducibility of results. Also the reconstruction procedure is inherently sequential. The conversion of the decoupled normal deviates $\mathbf{y}$ to an internal coordinate vector $\mathbf{w}$ depends on the underlying LAPACK routine, that might already be optimized to use available multiple cores, but the *cgDNAmc* code has no other explicit parallelisation. In part this is because each configuration can be generated and analysed independently of all others, so that the suggested solution for generating large ensembles is to run multiple independent simulations at the same time, with a different seed for the pseudo-random number generator in each instance. By linearity, expectations from multiple runs can be aggregated as a weighted average with weights proportional to the number of configurations generated in each independent run. As an example we achieved a 2.4 speed up in this way by running four independent threads on a single laptop.

S.2.6 Run-times of key steps of algorithm

A simple profile of run times for the key steps of a simulation that calculates five expectations using 1 million configurations of the 300 bp λ_3 oligomer is:

Operation	Run time [s]	% of simulation
Generation of $\mathbf{y}$	59.08	12.66%
Transformation to $\mathbf{w}$	74.75	16.02%
Shape reconstruction	41.41	8.87%
Calculating $\langle \mathbf{t}_0^{[0]} \cdot \mathbf{t}_i^{[0]} \rangle$	6.63	1.42%
Calculating $\langle \mathbf{t}_0^{[11]} \cdot \mathbf{t}_i^{[11]} \rangle$	273.28	58.55%
Calculating $\langle s_i^{[1]} \rangle$	3.72	0.80%
Calculating $\langle s_i^{[11]} \rangle$	0.61	0.13%
Calculating Flory vecs	6.12	1.30%
Other	1.14	0.24%
Entire simulation	466.74	100.00%

The time necessary to evaluate most of the expectations is a negligible fraction of the total, except for the generalized-chord expectation $\langle \mathbf{t}_0^{[11]} \cdot \mathbf{t}_i^{[11]} \rangle$, where the computation of the principal eigenvector of the local gyration matrix is quite costly.