Generating equilateral random polygons in confinement II

Y. Diao[†], C. Ernst[*], A. Montemayor[*], and U. Ziegler[*]

[*]Department of Mathematics and Computer Science
Western Kentucky University
Bowling Green, KY 42101, USA
[†]Department of Mathematics and Statistics
University of North Carolina Charlotte
Charlotte, NC 28223

Abstract. In this paper we continue an earlier study [10] on the generation algorithms of random equilateral polygons confined in a sphere. Here, the equilateral random polygons are rooted at the center of the confining sphere and the confining sphere behaves like an absorbing boundary. One way to generate such a random polygon is the accept/reject method in which an unconditioned equilateral random polygon rooted at origin is generated. The polygon is accepted if it is within the confining sphere, otherwise it is is rejected and the process is repeated. The algorithm proposed in this paper offers an alternative to the accept/reject method, yielding a faster generation process when the confining sphere is small. In order to use this algorithm effectively, a large, reusable data set needs to be pre-computed only once. We derive the theoretical distribution of the given random polygon model and demonstrate, with strong numerical evidence, that our implementation of the algorithm follows this distribution. A run time analysis and a numerical error estimate are given at the end of the paper.

AMS classification scheme numbers: PACS 87.14.G, 87.15.A, 87.10.-e, 87.15.B, 02.50.Cw. MSC 92D20,92B05,46N30,92C40.

Submitted to: *J. Phys. A: Math. Gen.*

1. Introduction

Knots are commonly found in nucleic acids and proteins. DNA knots appear as the product of random cyclization reactions of DNA molecules in solution [30, 33] and in confinement [2], as products of enzyme mediated biochemical reactions such as those mediated by site-specific recombinases [5, 7, 8, 14, 15, 31, 35, 40], topoisomerases [6, 9, 16, 42] and condensins [18, 28, 34] and as nanotechnological devices [26, 32]. Knots are also found along the backbone of some proteins. Recent studies of some protein crystal structures from viruses [44], bacteria [19] and humans [41] have revealed knotted structures in a wide variety of enzymes such as RNA methyltransferases [27], kinases [44] and transmembrane protein [19]. Knots have posed a new paradigm in protein folding and may have important functional and evolutionary implications [21, 22, 36, 41]. A few examples of linked protein rings have been reported and they are believed to provide stability to the complex they are part of. These include the proteins that form the capsid of certain viruses (e.g. [43]), proteins in thermofilic organisms [4] as well as some engineered proteins [3].

In order to carry out numerical studies needed for mathematically understanding and analyzing these biological systems it is necessary to model circular molecules. A few commonly used models for such circular molecules include the various random polygon models that may or may not take volume exclusion into consideration, the wormlike chain model and the bead model. However, for a large scale numerical study, a fast and reliable algorithm to generate non-correlated ensembles of random circular molecules (whatever the model one uses) is needed and is often a very difficult problem to overcome. The simplest representation of a circular molecule is by an equilateral random polygon in 3-space with n edges [12, 13], where each edge represents one or more monomers [24]. The equilateral random walks, as it turns out, have been studied extensively for many years, since they are also used to model long polymer chains in chemistry. There are a few algorithms that can generate large ensembles of equilateral random polygons relatively fast. These include the crankshaft algorithm [1, 20, 25] and the hedgehog algorithm [20, 29], and the generalized hedgehog algorithm [39]. Although it has not been mathematically proven that any of these methods generate the equilateral random polygons according to their probability distribution (though numerical evidences strongly indicate that they do), it has been shown that the crankshaft algorithm [1, 20, 25, 38] and the generalized hedgehog algorithm [39] are ergodic in the sense that any possible configuration of an equilateral random polygon can be in theory reached by this algorithm.

In this paper, we continue our earlier studies [10, 11] that focus on equilateral random polygons under a confinement condition. This study is motivated by the following biological problem. It is known that macromolecular self-assembly processes are key players in the complex network of interactions that take place in every organism. One of the ubiquitous self-assembly processes is the packing of genomic material (long

DNA chains) inside living organisms. Even in the case of a simple organism such as viruses, the DNA packing is of high density. For example, in the prototypic case of the P4 bacteriophage virus, the $3\mu m$-long double-stranded DNA is packed within a viral capsid with a caliper size of about $50nm$, corresponding to a 70-fold linear compaction [17]. How such a high density packing is achieved and what its effects are on the overall properties of the packed DNA is of great interest to scientists across several disciplines including mathematics and biophysics. In both cases, considerable effort has been spent in recent years to define and characterize the level of entanglement of the packaged DNA. This problem is one of the main factors that motivate the study of random polygons with a confinement condition, as such random polygons serve as a coarse model of packed DNA.

There are several issues involving any (random polygon) model-based numerical study of the DNA packing problem. The first issue is how to define the models to reflect the various packing properties the DNA or polymer chains may have. Once a confined random polygon model is defined, the next issue is determining the probability distributions of the random polygons based on the model, and the third issue is the actual generation of the random polygons in accordance with these (theoretical) probability distributions.

The focus of this paper is on the third issue for one particular confined random polygon model. In our earlier study [10, 11] we focused on equilateral random polygons that are confined in a sphere of fixed radius, where the confining sphere behaves somewhat similar to a "reflective surface". In this paper, the confined equilateral random polygon model is defined such that the confining sphere behaves more like an "absorbing boundary". More precisely, in [10] an equilateral random polygon is generated sequentially starting and ending at the origin by reversing the order (vertex by vertex) of equilateral random walks with fixed end points. Once a vertex X_{k+1} is generated, the next vertex X_k is generated subject to two conditions:
(1) $X_0 = O$, X_1, ... X_k, X_{k+1} form an equilateral random walk with fixed end points $X_0 = O$ and X_{k+1} and
(2) X_k is within the confining sphere.
Whereas in this paper, condition (2) above is replaced by the following condition:
(2′) X_j is within the confining sphere for $1 \leq j \leq k$.

This seemingly small difference in the definition in fact makes a big difference in the distributions of the random polygons and also a big difference in the ways the polygons can be generated. In fact, the confined random equilateral polygon model defined using condition (2′) results in precisely the equilateral random polygons that are bounded within the confining sphere, hence their probability density function is the same as the probability density function in the non-confined case, normalized by a constant (which would be the reciprocal of the probability for such a random polygon to be within the confining sphere). This is not the case when condition (2) is used instead. Similar to [10], our aim here is to develop an algorithm that is capable of

producing large sets of relatively long confined equilateral random polygons according to their probability distributions in a reasonable runtime. As pointed out in [10], in the non-confined case, there are several known algorithms that work reasonably well, these include the crankshaft algorithm [20, 25], the hedgehog algorithm [20, 29] and the generalized hedgehog algorithm [39]. In order to generate a random equilateral polygon under the confinement conditions (1) and (2′), one can always use the simple accept/reject approach. That is, we can generate an equilateral random polygon without the confinement condition using one of the methods mentioned above, but only accept the polygons that are within the confinement sphere. The main problem with this approach is the run time constrain. If the polygon is long (relative to the radius of the confining sphere), then the rejection rate will be very high (since only exponentially few of the generated polygons would be bounded within the confining sphere). See the review article [23] about generating polygons in confinement with a more sophisticated approach of an accept/reject approach to generated polygons in confinement. To improve on the accept/reject approach is the main motivation for our study in this paper. Let us point out that it is widely accepted (confirmed by numerous numerical studies) that these methods do generate equilateral random polygons (in the non-confined case) with the correct distribution (even though none has been theoretically proven), therefore, these methods can still be of use to us. We use them to generate confined equilateral random polygons with the accept/reject approach and compare them with the polygons generated by our new method to validate/refute our proposed method.

The method proposed in [10] relied on a schema that used a database of evenly spaced sample points where the exact value was computed by the evaluation of required cumulative probability density function by analytic integration. These sample points then allowed us to approximate the cumulative probability density functions at those points where we did not sample. This yields algorithms with a runtime of about $O(n^{3.5})$ (where n is the length of the polygon). A similar idea is used in this paper. That is, our generation method also requires some precomputation. However, for the confined random equilateral polygons defined here, the probability distribution is more difficult to deal with numerically. Consequently, the computation time needed, both for the creation of the initial data set and the generation of the random polygons, is much longer.

The rest of the paper is organized as follows: In Section 2, we give the theoretical background of the conditional probability density distributions needed. This section repeats some results already established in [10] that are essential for this paper to be self-contained and allow the the reader to understand our methods. In Section 3, we introduce the basic principle of an algorithm for generating the equilateral random polygon. We also derive a closed form expression for the needed theoretical conditional probability distributions. In Section 4, we discuss how to numerically implement the algorithm and provide some numerical results on runtime for both algorithms, the effect of numerical error on the shape of a generated polygons and the vertex distributions of

the generated confined polygons.

2. Basic background, terminology and notations

As this paper is a sequel to [10], the background is similar and we use similar terminology and notations. The reader may refer to [10] for details that may be missing here.

Suppose U_1, U_2, ... , U_n are n independent random vectors uniformly distributed on S^2 (so the joint probability density function of the three coordinates of each U_j is simply $\frac{1}{4\pi}$ on the unit sphere). An equilateral random walk of n steps, denoted by EW_n^O, is defined as the sequence of points in the three dimensional space $\mathbf{R}^3$: $X_0 = O$, $X_k = U_1 + U_2 + \cdots + U_k$, $k = 1, 2, ..., n$. Each X_k is called a vertex of the EW_n^O and the line segment joining X_k and X_{k+1} is called an edge of EW_n^O (which is of unit length). If the last vertex X_n of EW_n^O is fixed at X, then we have a conditioned random walk $EW_n^O|_{X_n=X}$. In particular, EW_n^O becomes a polygon EP_n if $X_n = O$. In this case, it is called an equilateral random polygon and is denoted by EP_n. The confining sphere considered in this paper is the sphere $S_R(O)$ centered at the origin with a radius $R \geq 1$ and the random polygon is rooted at the origin. The later condition is only for simplicity at this stage of study with no biological justification provided. Let n be the length of the equilateral polygon, then the polygon has n vertices and the vertices are denoted (in the sequential order as they appear on the polygon) by $X_0 = O$, X_1, X_2, ..., X_{n-1}, $X_n = X_0 = O$. In [10], the equilateral random polygon confined in $S_R(O)$ is defined sequentially (in a backward order, that is, X_n, X_{n-1}, ..., X_1, X_0) subject to

Condition (r): once X_{k+1} is generated, the next vertex X_k is generated subject to the conditions that $X_0 = O$, X_1, ... X_k, X_{k+1} form an equilateral random walk with fixed end points $X_0 = O$ and X_{k+1} and that X_k is within the confining sphere $S_R(O)$.

In this paper, the equilateral random polygon confined in $S_R(O)$ is defined sequentially (in a backward order, that is X_n, X_{n-1}, ..., X_1, X_0) subject to

Condition (a): once X_{k+1} is generated, the next vertex X_k is generated subject to the condition that $X_0 = O$, X_1, ... X_k, X_{k+1} form an equilateral random walk with fixed end points $X_0 = O$ and X_{k+1} which is confined in $S_R(O)$.

An equilateral random polygon of length n is denoted by $EP_n^r(R)$ if it is generated using condition (r) and $EP_n^a(R)$ if it is generated using condition (a). The letters r and a are referring to the fact that the confining sphere acts in a way similar to a reflective surface in the first case and an absorbing surface in the second case. At this point, the difference between the two definitions may seem subtle so let us elaborate on the importance of this difference. Suppose that X_{k+1} has been generated (so it is fixed). To generate X_k, we generate an equilateral random walk (of length $k+1$ with end points at O and X_{k+1}). In the first definition, as long as this random walk satisfy the condition that $|X_k| \leq R$, no matter where the other vertices are, X_k (and only X_k!) is selected and this process is repeated for the choice of X_{k-1}. However, in the second

definition, X_k is rejected if any of the vertices X_2, ..., X_{k-1} is not bounded in $S_R(O)$. Clearly the exact positions of X_1, X_2, ..., X_{k-1} are not known when the position of X_k is generated but different sets of potential positions for X_1, X_2, ..., X_{k-1} strongly influence the conditional probability density function that determines the position of X_k.

In the following, we provide a list of the various probability and conditional probability density functions needed for the rest of the paper, as well as some known formulas about them. The derivations of the formulas can be found in [10] and the references therein. Throughout the rest of the paper, for any integer k, r_k stands for $|X_k|$, the distance between X_k (the k-th vertex of the random polygon or the random walk) and the origin. Most of the time r_k is a random variable when X_k is treated as a random point (on the random walk or polygon), however r_k can also mean a constant when the point X_k has been already chosen.

(i) Let X_k be the k-th vertex of an equilateral random walk starting at the origin. The probability density function of $r_k = |X_k|$ is denoted by $g_k(r_k)$, which is given by the following closed formula:

$$g_k(r_k) = \frac{2r_k}{\pi} \int_0^\infty x \sin r_k x \left(\frac{\sin x}{x}\right)^k dx. \tag{1}$$

(ii) Let X_k be the k-th vertex of an equilateral random walk with end points $X_0 = O$ and X_{k+1}. Then under the condition that $|X_{k+1}| = r_{k+1}$ is fixed (that is, the end point X_{k+1} can be anywhere on the sphere of radius r_{k+1} centered at O), the conditional probability density function of r_k is denoted by $h_k(r_k|r_{k+1})$ and we have the following formula [10]:

$$h_k(r_k|r_{k+1}) = \frac{r_{k+1}}{2r_k} \frac{g_k(r_k)}{g_{k+1}(r_{k+1})}. \tag{2}$$

The corresponding cumulative probability distribution $P(|X_k| \leq r_k|r_{k+1})$ will be denoted by $H_k(r_k|r_{k+1})$.

(iii) Let $EW^O_{k+1}|_{X_{k+1}}$ be an equilateral random walk of length $k+1$ with fixed end points $X_0 = O$ and X_{k+1} such that $|X_{k+1}| = r_{k+1} \leq R$. Let B_k be the event that $EW^O_{k+1}|_{X_{k+1}}$ is confined in $S_R(O)$. The probability density function of $|X_k| = r_k$ given B_k and $|X_{k+1}| = r_{k+1}(\leq R)$ is denoted by $\lambda_k(r_k|B_k, r_{k+1})$ and its corresponding cumulative probability distribution function is denoted by $\Lambda_k(r_k|B_k, r_{k+1})$.

3. An $EP^a_n(R)$ generating algorithm based on the distribution functions $\lambda_k(r_k|B_k, r_{k+1})$ and $\Lambda_k(r_k|B_k, r_{k+1})$

3.1. The description of the algorithm.

Let us suppose we have a formula for the distribution functions $\lambda_k(r_k|B_k, r_{k+1})$ and $\Lambda_k(r_k|B_k, r_{k+1})$ and an effective way to compute them, then we use the following algorithm to generate an $EP^a_n(R)$.

Initial step: The starting and ending point of the polygon is set to be the origin by default. X_{n-1} is chosen uniformly on the unit sphere centered at the origin. The uniformity is guaranteed by the symmetry of the random polygon with respect to the confining sphere.

Recursive steps: Starting with $j = 2$. Given that in the previous step, X_{n-j+1} has been chosen. Thus X_{n-j+1}, hence $r_{n-j+1} = |X_{n-j+1}|$, is fixed. r_{n-j} is thus a random variable with probability density function $\lambda_{n-j}(r_{n-j}|B_{n-j}, r_{n-j+1})$. It can therefore be chosen to be the solution of the equation $\Lambda_{n-j}(r_{n-j}|B_{n-j}, r_{n-j+1}) = u$, where u is a random number uniformly chosen from $[0, 1]$. Once r_{n-j} is chosen, X_{n-j} is chosen uniformly on the intersection circle of the unit sphere centered at X_{n-j+1} and the sphere centered at O with radius r_{n-j}. This process is repeated for all $j \leq n-2$, and terminates once X_2 has been chosen.

Final step: At this point X_1 is simply chosen uniformly from the intersection circle of the unit sphere centered at X_2 and the unit sphere centered at O. This allows the random walk to return to the origin.

3.2. The derivation of an explicit formulation of $\lambda_k(r_k|B_k, r_{k+1})$ and $\Lambda_k(r_k|B_k, r_{k+1})$.

We now derive an explicit formulation of the probability density function $\lambda_k(r_k|B_k, r_{k+1})$ and the corresponding cumulative probability distribution function $\Lambda_k(r_k|B_k, r_{k+1})$. Notice that by definition $\Lambda_k(r_k|B_k, r_{k+1}) = P(|X_k| \leq r_k|B_k, r_{k+1})$.

Let j be a positive integer, $r \geq 0$ and $\tau \geq 0$ be real numbers. Given that $r_{j+1} = \tau$ then we denote by $I_j(\tau)$ the interval that contains all possible values of r_j, that is

$$I_j(\tau) = [|\tau - 1|, \min(R, j, \tau + 1)].$$

$I_j(\tau, r)$ stands for the interval of all possible values of r_j subject to two conditions $r_{j+1} = \tau$ and $r_j \leq r$.

$$I_j(\tau, r) = [|\tau - 1|, \min(R, j, r, \tau + 1)]$$

Next we define

$$V_k(\tau, r) = \int_{I_k(\tau,r)} \int_{I_{k-1}(\tau_k)} \cdots \int_{I_2(\tau_3)} d\tau_2 \cdots d\tau_{k-1} d\tau_k. \tag{3}$$

Notice that since $I_k(\tau, R) = I_k(\tau)$, we have

$$V_k(\tau, R) = \int_{I_k(\tau)} \int_{I_{k-1}(\tau_k)} \cdots \int_{I_2(\tau_3)} d\tau_2 \cdots d\tau_{k-1} d\tau_k. \tag{4}$$

The following theorem gives an explicit expression for the needed conditional cumulative probability distributions.

Theorem 1 *Given that X_{k+1} is fixed with $|X_{k+1}| = r_{k+1}$ (so r_{k+1} is also fixed), then we have*

$$\Lambda_k(r_k|B_k, r_{k+1}) = P(|X_k| \leq r_k|B_k, r_{k+1}) = \frac{V_k(r_{k+1}, r_k)}{V_k(r_{k+1}, R)}. \tag{5}$$

Proof. Recall all the probability events in consideration are under the condition that X_{k+1} is fixed with $|X_{k+1}| = r_{k+1} \leq R$ and the condition $|X_{k+1}| = r_{k+1}$ is abbreviated as r_{k+1}. Using Bayes Theorem we have

$$\begin{aligned} P(|X_k| \leq r_k|B_k, r_{k+1}) &= \frac{P(|X_k| \leq r_k, B_k|r_{k+1})}{P(B_k|r_{k+1})} \\ &= \frac{P(B_{k-1}, |X_k| \leq r_k \leq R|r_{k+1})}{P(B_k|r_{k+1})}. \end{aligned} \tag{6}$$

Using (2) we obtain

$$\begin{aligned} P(B_k|r_{k+1}) &= \int_{I_k(r_{k+1})} P(B_k|r_k) h_k(r_k|r_{k+1}) dr_k \\ &= \int_{I_k(r_{k+1})} P(B_{k-1}|r_k) h_k(r_k|r_{k+1}) dr_k \\ &= \int_{I_k(r_{k+1})} P(B_{k-1}|r_k) \frac{r_{k+1}}{2r_k} \frac{g_k(r_k)}{g_{k+1}(r_{k+1})} dr_k, \end{aligned}$$

where in the above the event B_k is replaced by B_{k-1} since the condition that $|X_k| = r_k$ and $r_k \in I_k(r_{k+1})$ trivially implies that $|X_k| \leq R$. Similarly (with $r_k \leq R$),

$$P(B_{k-1}|r_k) = \int_{I_{k-1}(r_k)} P(B_{k-2}|r_{k-1}) \frac{r_k}{2r_{k-1}} \frac{g_{k-1}(r_{k-1})}{g_k(r_k)} dr_{k-1},$$

hence

$$P(B_k|r_{k+1}) = \int_{I_k(r_{k+1})} \int_{I_{k-1}(r_k)} P(B_{k-2}|r_{k-1}) \frac{r_{k+1}}{2^2 r_{k-1}} \frac{g_{k-1}(r_{k-1})}{g_{k+1}(r_{k+1})} dr_{k-1} dr_k.$$

Repeating this procedure until we reach B_2, and since B_1 is trivially true, $g_2(r_2) = r_2/2$, we obtain

$$\begin{aligned} P(B_k|r_{k+1}) &= \int_{I_k(r_{k+1})} \int_{I_{k-1}(r_k)} \cdots \int_{I_2(r_3)} \frac{r_{k+1}}{2^{k-1} r_2} \frac{g_2(r_2)}{g_{k+1}(r_{k+1})} dr_2 \cdots dr_{k-1} dr_k \\ &= \frac{r_{k+1}}{2^k g_{k+1}(r_{k+1})} V_k(r_{k+1}, R). \end{aligned}$$

Similarly,

$$\begin{aligned} & P((B_{k-1}, |X_k| \leq r_k \leq R)|r_{k+1}) \\ &= \int_{I_k(r_{k+1}, r_k)} P(B_{k-1}|r'_k) h_k(r'_k|r_{k+1}) dr'_k \\ &= \int_{I_k(r_{k+1}, r_k)} P(B_{k-1}|r'_k) \frac{r_{k+1}}{2r'_k} \frac{g_k(r'_k)}{g_{k+1}(r_{k+1})} dr'_k \end{aligned}$$

$$
\begin{aligned}
&= \int_{I_k(r_{k+1},r_k)} \int_{I_{k-1}(r_k)} P(B_{k-2}|r_{k-1}) \frac{r_{k+1}}{2^2 r_{k-1}} \frac{g_{k-1}(r_{k-1})}{g_{k+1}(r_{k+1})} dr_{k-1} dr'_k \\
&= \cdots \\
&= \int_{I_k(r_{k+1},r_k)} \int_{I_{k-1}(r_k)} \cdots \int_{I_2(r_3)} \frac{r_{k+1}}{2^{k-1} r_2} \frac{g_2(r_2)}{g_{k+1}(r_{k+1})} dr_2 \cdots dr_{k-1} dr'_k \\
&= \frac{r_{k+1}}{2^k g_{k+1}(r_{k+1})} V_k(r_{k+1}, r_k).
\end{aligned}
$$

(5) now follows trivially from (6).

4. The implementation of the algorithm and numerical results

Using Theorem 1, we implement the algorithm outlined in the last section as follows. For the given confinement radius R and the polygon length n, we first precompute the integrals in the volume function defined in (3) for all values up n. For $k = 2, 3, \ldots, n-1$, define $\Lambda_k(r_k|r_{k+1}) = \frac{V_k(r_{k+1},r_k)}{V_k(r_{k+1},R)}$. Once X_{k+1} is selected, a random number $u \in U[0,1]$ is chosen and $r_k = |X_k|$ is selected as the solution to the equation $u = \Lambda_k(r_k|r_{k+1})$. The solution to the equation $u = \Lambda_k(r_k|r_{k+1})$ cannot be found analytically and must be accomplished by a numerical, iterative process which terminates when a desired precision is reached. More precisely, since $\Lambda_k(r_k|r_{k+1})$ is a non-decreasing function (of r_k, as r_{k+1} is fixed), a simple bisection (binary search) algorithm can be used on the interval $I_k(r_{k+1})$. Thus an efficient implementation of the overall algorithm must include an efficient implementation of the bisection algorithm to estimate the inverse cumulative distribution functions $\Lambda_k^{-1}(r_k|r_{k+1})$. In our implementation of the algorithm, the precision level of the computation is measured in the form of 10^{-m} for various positive integer m.

4.1. Comparison with the generalized hedgehog method.

To test the validity of the algorithm we decided to compare it with data collected using the accept/reject approach based on the hedgehog method. We set $R = 2$ and generated a polygon of length 20 using the hedgehog method. If the polygon was inside the confining sphere we kept the polygon; if any part of the polygon was outside the confinement we deleted the polygon and started over. We continued this process and collected 20000 polygons in confinement. We also collected 10000 polygons in confinement generated by the algorithm described in the previous section. For each set of polygons we computed the distance of each vertex from the origin and binned all distances to form a vertex density chart for the sphere of radius $R = 2$. In this computation we deleted the starting vertex $v_0 = v_{20}$ (since it has distance zero) and the vertices v_1 and v_{19} (since they have distance one) to remove bars in the graph that are artifacts of anchoring the polygons at the origin. The scale on the y-axis of the histogram is chosen so that the total area is one and the histogram represents a pdf function. Figure 1 shows that the two histograms are virtually identical providing strong numerical evidence that our algorithm generates random polygons in confinement

according to the theoretical probability distribution for the underlying model. It also shows that the vertex density is not uniform, but strongly declines for radii greater than one.

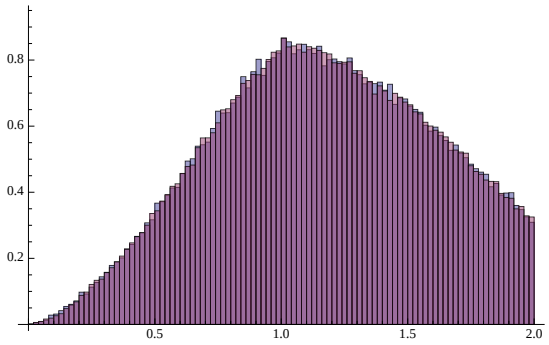

Figure 1. Two virtually identical histograms of polygons with 20 segments in a confinement sphere of radius two.

We investigated the runtime of our implementation in comparison to the existing generalized hedgehog method. We generated polygons with the generalized hedgehog method and only kept those which fit into the confining sphere. We measured the time needed to generate 1000 polygons of various lengths and confining radii of R = 1, 1.5, 2, and 2.5 for both the generalized hedgehog and our method. The results of the analysis are summarized in Figure 2. The x-axis shows the length of the polygons and the y-axis is the time used. (The y-axis uses a logarithmic scale to better represent all the data in the plot.) The runtime for the hedgehog method is shown with empty markers and the runtime for our method with filled markers. For each radius of the confining sphere the measured runtime for the hedgehog method grows faster than the runtime for our method and there is a cross over point at which our method generates polygons faster. All cross over points are shown in the same figure - from left to right for R = 1, 1.5, 2, and 2.5. The lengths of the polygons at the crossovers are roughly 11, 23, 41, and 68, respectively. We want to remark that the actual crossovers of runtimes are, of course, dependent on how the two algorithms are implemented. In our case the hedgehog algorithm is implemented with a C program while our algorithm is implemented in Mathematica making use of its high precision and ability to compute the needed volume functions given in equations 3 and 4. Implementation of both algorithms in the same programming environment would change the location of the crossovers. However it would not change the fact that our algorithm is much faster for polygons in a relatively small confinement sphere.

4.2. The effect of the integer m on the shape of the polygons.

Larger m values force the algorithm to make a smaller error when solving the equation $u = \Lambda_k(r_k|r_{k+1})$ for r_k. The effect of the choice of the integer m can be measured very effectively. For the generation of each vertex X_k we need to choose a random real

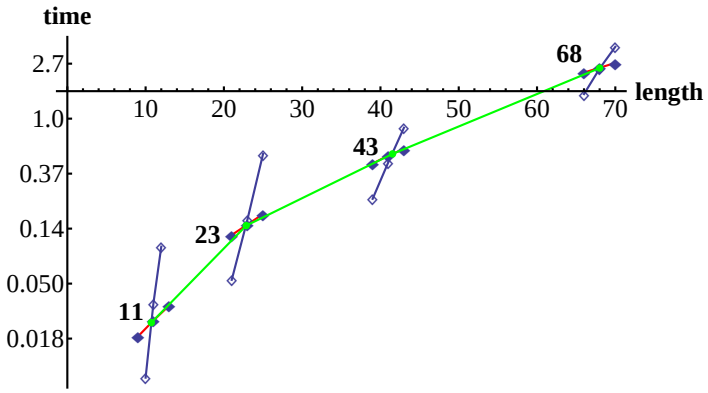

Figure 2. The diagram shows the polygon length at which the runtime of our method (filled markers) becomes faster than the generalized hedgehog method (empty markers). From left to right, the radii are 1, 1.5, 2, and 2.5. The x-axis shows the length of the polygons and the y-axis shows the average time per polygon (scaled using ln).

number u_k in the unit interval I. Thus we say that the polygon EP_n uses a list of seed values $\{u_i\}$ in its generation. For a fixed set of seed values $\{u_i\}$ we choose several values of m to generate polygons. By using the same seed values $\{u_i\}$ we directly measure the effect of the different m values by comparing the polygons that are generated. In the left of Figure 3 two polygons are shown that are generated using $m = 4$ and $m = 5$ and even at this low precision they cannot be visually distinguished. On the right in Figure 3 we see that the difference between any two corresponding vertices is at most 0.00042.

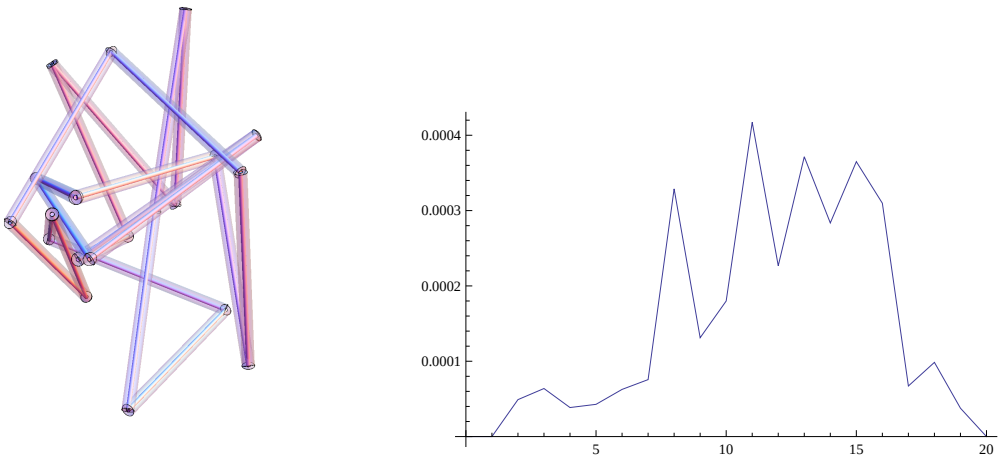

Figure 3. On the left two polygons with 20 segments are shown which were generated with the same seed values u_i using $m = 4$, $m = 5$ and $R = 1$. For $m = 4$ we use opaque cylinders of radius .03 and for $m = 5$ we use cylinders of radius .01. On the right the distance between corresponding vertices is shown.

To check that the above behavior is typical we computed sets of 100 polygons using

a fixed list of seed values for various values of k and R. In the table below we show the results for $k = 20$ and $R = 1$. For a given m value, we compute the vertex displacement of the polygons generated using m and $m+1$ respectively. The maximum of this vertex displacement, taken over the 100 polygons generated, is denoted by D. Table 1 shows the relationship between m and D in our study. Figure 4 is the log plot of the data in Table 1, which has a best fit line with a slope of about -0.99431. From this we conclude that for a fixed m we generate a polygon with a spatial accuracy of at least 10^{3-m}. For other values of k and R we have found similar results.

m	3	4	5	6	7	8	9	10
D	$2.6 \cdot 10^{-1}$	$2.1 \cdot 10^{-2}$	$3.0 \cdot 10^{-3}$	$1.5 \cdot 10^{-4}$	$3.6 \cdot 10^{-5}$	$4.6 \cdot 10^{-6}$	$1.3 \cdot 10^{-7}$	$4.6 \cdot 10^{-8}$

Table 1. The relationship between m and the maximum vertex displacement D.

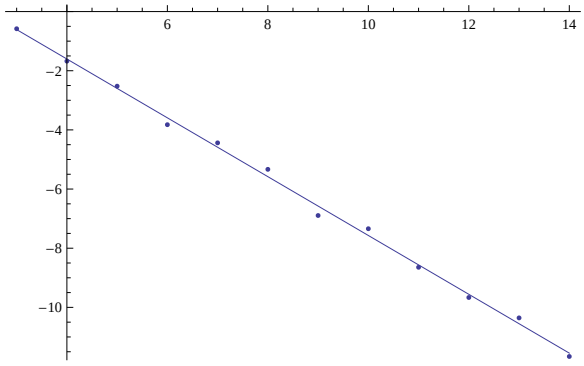

Figure 4. The log plot of the data presented in Table 1.

This shows that the numerical errors do not accumulate, and get out of control. Heuristically this can be explained as follows: Each rounding error might cause the computation of a radius that is slightly too large or slightly too small with equal probability. Thus these types of rounding error have the tendency to cancel each other. Similarly, a rounding error may lead us to a slightly different value of τ, say τ'. We now have two different cumulative probability density functions $\Lambda_k(r_k|B_k, r_{k+1} = \tau)$ and $\Lambda_k(r_k|B_k, r_{k+1} = \tau')$ for which we compute the inverse value for a fixed choice of u. Comparing the two plots of such graphs we can see that the two graphs sometimes intersect and we suspect that such intersections help to reduce the rounding error in r_k. The behavior shown in Figure 3 on the right shows these random fluctuations of the rounding error.

4.3. The runtime of the algorithm.

The runtime of the algorithm increases with the length of the generated polygons as shown in Figure 5. In the data set used for this study, for each length 10, 20, 30, 40,

and 50 and each R value between 1 and 4 with 0.5 increments, 1000 polygons were generated. For a few R values (1, 2.5, and 3) sets of 1000 polygons each with lengths 60, 70, 80, and 90 were also generated. The right side of Figure 5 shows the runtime for polygons up to length 90. The data shows that the runtime grows slower than an exponential function, but faster than a power function. The function shown with the data in Figure 5 is $f(x) = 1.6 \cdot 10^{-6} \cdot x^3 \cdot \ln(x)$ but there is no evidence that the fit is good for polygons with length > 90. The function merely provides context information about how the runtime grows for the given data.

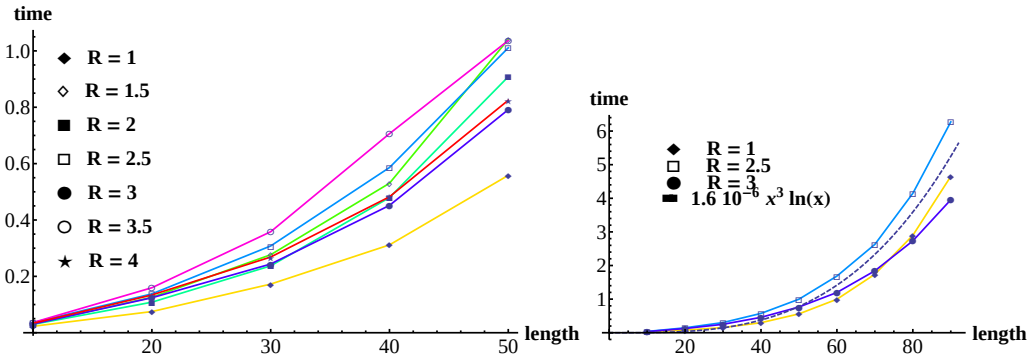

Figure 5. Left: Plot of runtime as a function of the length of the polygon with length up to 50. Different lines are used for different radii of the confining sphere. Right: Plot of runtime for polygons up to length 90 for $R = 1$, 2.5 and 3, together with the plot of the function $f(x) = 1.6 \cdot 10^{-6} \cdot x^3 \cdot \ln(x)$.

The change in the runtime with respect to the change in R is more complex than one might have expected. The runtime does not consistently increase with the increase of R. It seems that less time is needed for integer values of R than for fractional values of R as suggested by the left side plot of Figure 5. In fact, starting at length 30, the maximum runtime among the integer R values is less than the minimum runtime among the non-integer R values in our study. Oddly, generating polygons in confinement with $R = 3$ is faster than for $R = 2$, as suggested by Figure 6 where the runtime is plotted as a function of R (for several fixed polygon lengths).

4.4. The vertex distribution.

One important characteristic (of the random polygon model) to be considered is how the vertices are distributed within the confining sphere. Let v_i be the i-th vertex of a confined equilateral random polygon of total length n. Then $|v_i|$ (the distance of v_i to the origin) is a random variable. Let $f_i(|v_i|)$ be the probability density function of $|v_i|$. For simplicity of the presentation, instead of investigating $f_i(|v_i|)$ for each i, we choose to study their simple summation $\sum_{0 \le i < n} f_i(|v_i|)$ instead. The results presented in this section correspond to $n = 20$. Because $|v_0| = |v_{20}| = 0$, $|v_1| = |v_{19}| = 1$, their corresponding density functions are just the delta functions and are thus should be excluded from the study. Furthermore, $f_2(|v_2|)$ and $f_{18}(|v_{18}|)$ are both density

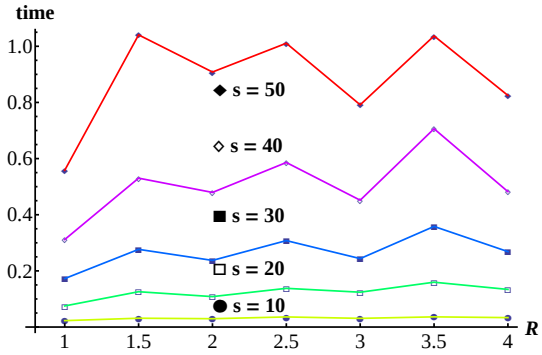

Figure 6. The runtime dependence on the radii of the confining sphere is shown for several fixed lengths (denoted by s in the figure).

functions with jump discontinuity and are thus also excluded from our study, otherwise the obtained distributions may contain discontinuity that may distract the reader's attention. Thus what we presented in this section is $\sum_{3\leq i\leq 17} f_i(|v_i|)$ for the case of $n = 20$, normalized by a factor of $1/15$. For the purpose of comparison, corresponding results using the generating method of [10] are also provided. Using each method, we generate 10000 confined polygons of length 20 for each R value ranging from 1.5 to 4.5 (with an increment of 0.5). The case $R = \infty$ (corresponding to no confinement) is also included (realized by using $R = 10$). For each set of polygons corresponding to a given R value, we partition $[0, R]$ into small equal length subintervals first, compute $|v_i|$ for each $3 \leq i \leq 17$ and count the total number of these within each subinterval. The resulting relative frequency histogram is an estimate of $F = (1/15)\sum_{3\leq i\leq 17} f_i(|v_i|)$. The combined results are shown in Figure 7. In the figure, for each R value, a pair of graphs is given, one with the histogram obtained using the method in [10] at the front (left), and the other with the histogram obtained using the method in this paper at the front (right). This way the reader sees the difference of the two (average) distributions easily. Clearly, the figure shows that the two distributions of F from these two methods converges to each other as R increases to infinity as expected. However, for small R values, the distribution F from the method in [10] has a much larger weight towards its right tail, meaning that more vertices are crowded around the boundary of the confining sphere, comparing with F obtained from the method in this paper. These phenomena coincide with what one would expect from a reflective boundary and an absorbing boundary.

5. Conclusions and ending remarks

Following our earlier study [10], in this paper we explored the generating method of equilateral random polygons confined within a sphere, but under a different set of defining conditions. We rigorously proved that our algorithm generates the random polygons according to their probability distributions. When the radius of the confining

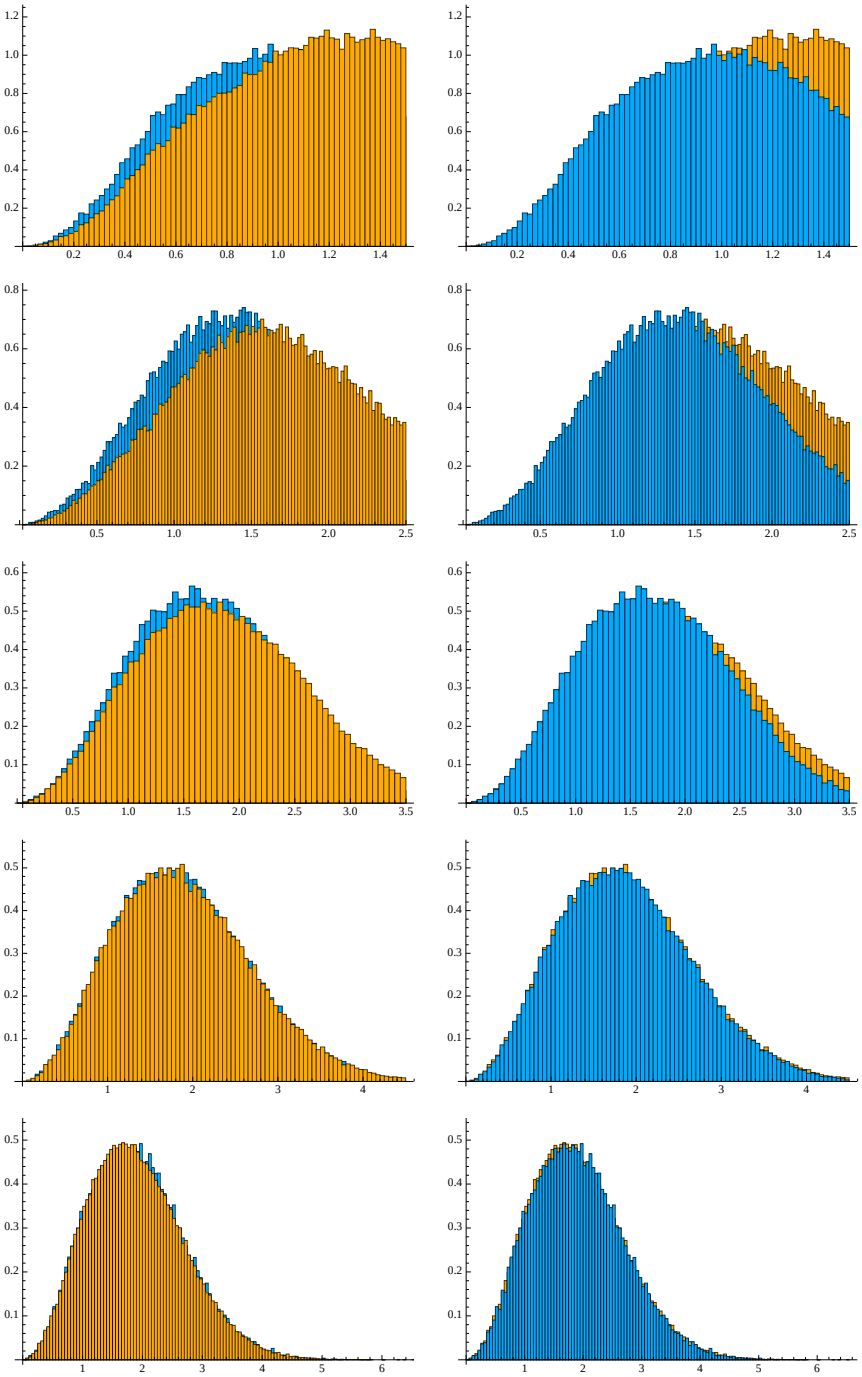

Figure 7. The histograms that approximate the average probability density function F for $R = 1.5$, 2.5, 3.5, 4.5 and 10 ordered from top to bottom. Darker shading relates to the model discussed in this paper; lighter shading relates to the model from [10].

.

sphere is small, our algorithm can generate the random polygons much faster than an algorithm that is based on the accept/reject approach. Many questions remain to be answered. For example, if the polygons are not rooted at the center of the confining sphere but their starting points are randomly selected from a certain distribution, how are such random polygons distributed within the confining sphere? Is it still possible for us to find a fast generating algorithm for such confined random polygons using confining conditions such as the ones used in this paper or the ones used in [10]? Another question is the following more biologically related question. Suppose that we know how the vertices of the random polygons are distributed within the confining sphere (for example such information may become available in a related biological study), how do we generate equilateral random polygons following this distribution? In particular, how do we generate equilateral random polygons such that their vertices are uniformly distributed within the confining sphere? We mention the uniform distribution since it is the simplest. Studying the simplest case may shed some light to the general case. These are a few directions that we intend to pursue in the future.

Acknowledgments

This work is supported in part by NSF Grants #DMS-0920880 and #DMS-1016460 (Y. Diao), and by NSF grant #DMS-1016420 (C. Ernst, A. Montemayor and U. Ziegler).

References

[1] Alvarado S, Calvo J and Millett K 2011, *The generation of random equilateral polygons*, J. Stat. Phys. **143** (1), 102–38.

[2] Arsuaga J, Vazquez M, Trigueros S, Sumners DW and Roca J 2002, *Knotting probability of DNA molecules confined in restricted volumes: DNA knotting in phage capsids,* Proc Natl Acad Sci USA **99** 5373–7.

[3] Blankenship J W and Dawson P E 2003, *Thermodynamics of a designed protein catenane,* J Mol Biol **327**(2) 537–48.

[4] Boutz D R, Cascio D, Whitelegge J, Perry L J and Yeates T O 2007, *Discovery of a thermophilic protein complex stabilized by topologically interlinked chains*, J Mol Biol **368**(5) 1332–44.

[5] Buck D and Flapan E 2007, *Predicting Knot or Catenane Type of Site-Specific Recombination Products*, J Mol Biol **374**(5) 1186–99.

[6] Buck G R and Zechiedrich E L 2004, *DNA disentangling by type-2 topoisomerases*, J Mol Biol **340**(5) 933–9.

[7] Darcy I K et al 2006, *Coloring the Mu transpososome*, BMC Bioinformatics **7** 435.

[8] ——and Scharein RG 2006, *TopoICE-R: 3D visualization modeling the topology of DNA recombination*, Bioinformatics **22**(14) 1790–1.

[9] Dean F B, Stasiak A, Koller T and Cozzarelli N R 1985, *Duplex DNA knots produced by Escherichia coli topoisomerase I. Structure and requirements for formation*, J Biol Chem **260** 4975–83.

[10] Diao Y, Ernst C, Montemayor A and Ziegler U 2011, *Generating equilateral random polygons in confinement*, J. Phys. A: Math. Theor. **44** 405202 doi:10.1088/1751-8113/44/40/405202.

[11] ——, Ernst C, Montemayor A and Ziegler U 2011, *Corrigendum for "Generating equilateral random polygons in confinement"*, *J. Phys. A: Math. Theor*, **44** 449501 doi:10.1088/1751-8113/44/44/449501 (2011).

[12] Edwards S F 1967, *Statistical mechanics with topological constraints I,* Proc Phys Soc **91** 513–9.
[13] ——1968, *Statistical mechanics with topological constraints II,* J Physics A **1** 15–28.
[14] Ernst, C and Sumners D W 1990, *A calculus for rational tangles with applications to DNA*, Math. Proc. Camb. Phil. Soc. 108(3), 489 - 515.
[15] ——1999, *Solving Tangle Equations arising in a DNA Recombination Model*, Math. Proc. Camb. Phil. Soc. 126, 23 - 36.
[16] Hsieh T 1983, *Knotting of the circular duplex DNA by type II DNA topoisomerase from Drosophila melanogaster,* J Biol Chem **258** 8413–20.
[17] Jardine P J and Anderson D L 2006, *DNA packaging in double-stranded DNA phages*, The bacteriophages, Ed. Richard Calendar, Oxford University Press, 49–65.
[18] Kimura K, Rybenkov V V, Crisona N J, Hirano T and Cozzarelli N R 1999, *13S condensin actively reconfigures DNA by introducing global positive writhe: implications for chromosome condensation,* Cell **98**(2) 239–48.
[19] King N P, Yeates E O and Yeates T O 2007,*Identification of rare slipknots in proteins and their implications for stability and folding*, J Mol Biol **373**(1) 153–66.
[20] Klenin K V, Vologodskii A V, Anshelevich V V, Dykhne A M and Frank-Kamenetskii M D 1988, *Effect of excluded volume on topological properties of circular DNA, J Biomolec Str and Dyn* **5** 1173–85.
[21] Lua R C and Grosberg A Y 2006,*Statistics of knots, geometry of conformations, and evolution of proteins,* PLoS Comput Biol **2**(5) e45.
[22] Mallam A L, Onuoha S C, Grossmann J C and Jackson S E 2008, *Knotted fusion proteins reveal unexpected possibilities in protein folding*, Mol Cell **30** 642–8.
[23] Micheletti C, Marenduzzo D, and Orlandini E 2011, *Polymers with spatial or topological constraints: Theoretical and computational results* Physics Reports **504** (1) 1–73.
[24] Michels J P J and Wiegel F W 1986, *On the Topology of a polymer ring,* Proc R Soc Lond A **403** 269–84.
[25] Millett K 1994, *Knotting of regular polygons in 3-space*, Ser. Knots Everything **7**, World Scientific, 31–46.
[26] Mueller J E, Du S M and Seeman N C 1991, *The design and synthesis of a knot from single-stranded DNA,* J Am Chem Soc **113** 6306–8.
[27] Nureki O et al 2002, *An enzyme with a deep trefoil knot for the active-site architecture*, Acta Crystallogr D Biol Crystallogr **58**(7) 1129–37.
[28] Petrushenko Z M, Lai C H, Rai R and Rybenkov V V 2006, *DNA reshaping by MukB. Right-handed knotting, left-handed supercoiling,* J Biol Chem **281**(8) 4606–15.
[29] Plunkett P et al 2007, *Total curvature and total torsion of knotted polymers*, *Macromolecules* **40** 3860–7.
[30] Rybenkov V V, Cozzarelli N R and Vologodskii A V 1993, *Probability of DNA knotting and the effective diameter of the DNA double helix,* Proc Natl Acad Sci USA **90** 5307–11.
[31] Saka Y and Vazquez M 2002, *TangleSolve: topological analysis of site-specific recombination*, Bioinformatics **18** 1011–2.
[32] Seeman N C 2003, *Biochemistry and structural DNA nanotechnology: an evolving symbiotic relationship,* Biochemistry **42** 7259–69.
[33] Shaw S Y and Wang J C 1993, *Knotting of a DNA chain during ring closure*, *Science* **260** 533–6.
[34] Stray J E et al 2005, *The Saccharomyces cerevisiae Smc2/4 condensin compacts DNA into (+) chiral structures without net supercoiling*, J Biol Chem **280**(41) 34723–34.
[35] Sumners D W, Ernst C, Spengler S J and Cozzarelli N R 1995, *Analysis of the mechanism of DNA recombination using tangles*, Quart. Rev. Biophysics 28(3), 253 - 313.
[36] Taylor W R 2000, *A deeply knotted protein structure and how it might fold,* Nature **406** 916–9.
[37] Thompson R 1966, *Evaluation of* $I_n(b) = \frac{2}{\pi}\int_0^\infty (\frac{\sin x}{x})^n \cos bx dx$ *and of Similar Integrals*, Mathematics of Computation **20** 330–2.
[38] Toussaint G 2005, *The Erdös-Nagy theorem and its ramifications*, Comput. Geom. **31**(3) 219–36.

[39] Varela R, Hinson K, Arsuaga J and Diao Y 2009, *A Fast Ergodic Algorithm for Generating Ensembles of Equilateral Random Polygons*, J. Phys. A: Math. Theor. **42** 095204.

[40] Vazquez M, Colloms S and Sumners DW 2005, *Tangle analysis of Xer recombination reveals only three solutions, all consistent with a single three-dimensional topological pathway*, J Mol Biol **346**(2) 493–504.

[41] Virnau P, Mirny L A and Kardar M 2006, *Intricate knots in proteins: Function and evolution,* PLoS Comput Biol **2**(9) e122.

[42] Wasserman S A and Cozzarelli N R 1991, *Supercoiled DNA-directed knotting by T4 topoisomerase*, J Biol Chem **266** 20567–73.

[43] Wikoff W R et al 2000, *Topologically linked protein rings in the bacteriophage HK97 capsid*, Science **289** 2129–33.

[44] Yeates T O, Norcross T S and King N P 2007, *Knotted and topologically complex proteins as models for studying folding and stability*, Curr Opin Chem Biol **11**(6) 595–603.