Precise algorithm to generate random sequential adsorption of hard polygons at saturation

G. Zhang

Precise algorithm to generate random sequential adsorption of hard polygons at saturation

G. Zhang*

Department of Physics, University of Pennsylvania, *Philadelphia, PA, 19104*

Abstract

Random sequential adsorption (RSA) is a time-dependent packing process, in which particles of certain shapes are randomly and sequentially placed into an empty space without overlap. In the infinite-time limit, the density approaches a "saturation" limit. Although this limit has attracted particular research interest, the majority of past studies could only probe this limit by extrapolation. We have previously found an algorithm to reach this limit using finite computational time for spherical particles, and could thus determine the saturation density of spheres with unprecedented accuracy. In this paper, we generalize this algorithm to generate saturated RSA packings of two-dimensional polygons. We also calculate the saturation density for regular polygons of three to ten sides, and obtain results that are consistent with previous, extrapolation-based studies.

PACS numbers: 05.10.-a, 45.70.-n, 05.20.-y

* gezhang@alumni.princeton.edu

I. INTRODUCTION

Random sequential adsorption (RSA) [1], also called random sequential addition [2], is a stochastic process widely used to model a variety of physical, chemical, and biological phenomena, including structure of cement paste [3], ion implantation in semiconductors [4], protein adsorption [5], particles in cell membranes [6], and settlement of animal territories [7]. Starting from a large, empty region in d-dimensional Euclidean space, particles of certain shapes are randomly and sequentially placed into the volume subject to a nonoverlap constraint: new particles are kept only if they do not overlap with any existing particles, and are discarded otherwise. One can stop this process at any time, obtaining configurations with time-dependent densities. As time increases, the density approaches a "saturation" or "jamming" limit, ϕ_s.

The RSA process of various particle shapes have been studied, including spheres in one through eight dimensions [8–13], squares and rectangles [14–18], polygons [19], ellipses [18, 20, 21], disk polymers [22], cubes [23], spheroids [24], superdisks [25], sphere polymers [26, 27], and four-dimensional hypercubes [28]. For non-spherical shapes, particle orientations may be random or fixed. Although previous researchers have studied a myriad of combinations of space dimensions, shapes, and orientations, the determination of ϕ_s have always been of particular interest. However, doing so is also particularly difficult since one cannot afford infinite computational time to reach the saturation limit. To overcome this problem, a very common strategy is to find out finite-time densities and then to extrapolate to the infinite-time limit [9, 12, 14, 15, 18–24, 26, 27].

Instead of extrapolation, ϕ_s for some systems can be ascertained by other strategies. For one-dimensional rods, analytical calculations found $\phi_s = 0.7475979202\ldots$ [8]. For disks, spheres, and d-dimensional hyperspheres, we have previously found a numerical algorithm to reach the saturation limit [13] with finite computational time. The algorithm takes advantage of the fact that when generating RSA packings of spheres of radius R, the distance between any two sphere centers cannot be smaller than $2R$. The part of space that is not within $2R$ distance to any existing sphere is called "available space," since a new sphere will be kept if and only if its center falls inside the available space. Thus, one can avoid insertion attempts in the unavailable part of the space [10–13, 16]. Saturation can be achieved by gradually increasing the resolution as additional spheres are inserted, eventually eliminating

all available spaces [10, 13]. Specifically, our algorithm consists of the following steps:

1. Perform a certain number of trial insertions to generate a near-saturation configuration.

2. Divide the simulation box into voxels (*i.e., d*-dimensional pixels) with side lengths comparable to R. Some voxels are completely included in an existing sphere of radius $2R$, and therefore cannot contain any available space. They are excluded from the voxel list.

3. Perform a certain number of trial insertions inside the remaining voxels.

4. Divide each voxel into 2^d sub-voxels by cutting it in half in each direction, and find possibly available sub-voxels.

5. The process of trial insertions and voxel division is repeated until the number of available voxels reaches zero, at which point we know that saturation is guaranteed since we only discard voxels that cannot contain any available space.

Ref. 13 ended with a proposal to extend this algorithm to generate saturated RSA packings of non-spherical shapes with random orientations. In this case, whether an incoming particle overlaps with existing ones depends on not only its location, but also its orientation. For a d-dimensional particle with d_f rotational degrees of freedom, one can construct a $(d + d_f)$-dimensional auxiliary space. Each point in this space would correspond to a trial insertion at a particular location with a particular orientation. One could thus use voxels to track the available parts of this higher-dimensional auxiliary space and generate saturated RSA packings. However, Ref. 13 did not propose any method to test for voxel availability, which is a nontrivial task.

In this paper, we use this idea to generate saturated RSA packings of 2D polygons with random orientations. We present a way to test for voxel availability based on worst-case error analysis in Sec. II B. We find that occasionally, the generalization of this algorithm needs a special tweak, detailed in Sec. II C. In Sec. III, we use this algorithm to find ϕ_s for regular polygons, which generally increases as the number of sides increases and approaches ϕ_s for disks.

II. ALGORITHMIC DETAILS

In order to use the algorithm described in Ref. 13, one need to supplement two subroutines: one to determine if two particles are overlapping and another to prove that certain voxels that cannot contain any available space. Here we describe these two subroutines.

A. polygon overlap test

To test if two polygons overlap, we first perform simple tests using their inscribed circles and circumscribed circles: If the inscribed circles overlap, then the two polygons must overlap. If the circumscribed circles do not overlap, then the two polygons cannot overlap. If these two simple tests fail to find a definitive answer, we then test if any two sides of the two polygons intersect with the following theorem [29]:

Define $O(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) = (y_2 - y_1)(x_3 - x_2) - (x_2 - x_1)(y_3 - y_2)$, where $\mathbf{r}_i = (x_i, y_i)$ is a two-dimensional point, then two line segments $(\mathbf{r}_1, \mathbf{r}_2)$ and $(\mathbf{r}_3, \mathbf{r}_4)$ intersect if and only if

$$O(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)O(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_4) < 0, \text{ and} \tag{1}$$

$$O(\mathbf{r}_3, \mathbf{r}_4, \mathbf{r}_1)O(\mathbf{r}_3, \mathbf{r}_4, \mathbf{r}_2) < 0. \tag{2}$$

B. voxel availability test

Our test for voxel availability is based on the aforementioned particle-overlapping test. Since a two-dimensional polygon has 1 rotational degree of freedom, the voxels are three-dimensional. Let (x, y) be the location of the center of a polygon and let θ be the angle between the orientation of the polygon and some reference orientation. A point in the voxel space can be represented by (x, y, θ), while a voxel centered at this point can be represented as $(x \pm \delta x, y \pm \delta y, \theta \pm \delta\theta)$, where δx, δy, and $\delta\theta$ are a half of the side length of the voxel in each direction. With this formulation, a voxel can be interpreted as a collection of trial insertions near (x, y, θ) with some error bounds δx, δy, and $\delta\theta$. To prove that a voxel cannot contain any available space (*i.e.,* to prove that such trial insertions always fail), we just need to perform a rigorous worse-case error analysis to prove that no matter how x, y, and θ vary in the ranges $[x - \delta x, x + \delta x]$, $[y - \delta y, y + \delta y]$, and $[\theta - \delta\theta, \theta + \delta\theta]$, the upcoming particle will always overlap with an existing one. Specifically, let $(\mathbf{r}_1, \mathbf{r}_2)$ be a side of an existing

polygon and let $(\mathbf{r}_3, \mathbf{r}_4)$ be a side of an upcoming polygon inside the voxel, then $\mathbf{r}_3$ and $\mathbf{r}_4$ carry uncertainties while $\mathbf{r}_1$ and $\mathbf{r}_2$ do not. Let (l_3, θ_3) and (x_3, y_3) be the polar coordinates and Cartesian coordinates of $\mathbf{r}_3$, we have $x_3 = x + l_3 \cos(\theta_3)$, and the associated error bound is

$$\delta x_3 = [x \pm \delta x + l_3 \cos(\theta_3 \pm \delta\theta)] - [x + l_3 \cos(\theta_3)] \tag{3}$$

$$\leq \delta x + l_3 \delta\theta. \tag{4}$$

Similarly,

$$\delta y_3 \leq \delta y + l_3 \delta\theta. \tag{5}$$

For simplicity, we can require that a voxel always have an equal side lengths in x and y directions, in this case $\delta x = \delta y$ and $\delta x_3 = \delta y_3$. We define $\delta_3 \equiv \delta x_3 = \delta y_3$. The error bounds for the Cartesian coordinates of vertex 4 are similar:

$$\delta x_4 \leq \delta x + l_4 \delta\theta \equiv \delta_4, \tag{6}$$

$$\delta y_4 \leq \delta y + l_4 \delta\theta = \delta_4, \tag{7}$$

The associated worse-case error in the O functions used in Eqs. (1)-(2) are thus

$$\delta O(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) = (y_2 - y_1)(x_3 \pm \delta_3 - x_2) - (x_2 - x_1)(y_3 \pm \delta_3 - y_2) \tag{8}$$

$$- (y_2 - y_1)(x_3 - x_2) - (x_2 - x_1)(y_3 - y_2) \tag{9}$$

$$\leq (|y_2 - y_1| + |x_2 - x_1|)\delta_3, \tag{10}$$

similarly

$$\delta O(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_4) \leq (|y_2 - y_1| + |x_2 - x_1|)\delta_4, \tag{11}$$

and

$$\delta O(\mathbf{r}_3, \mathbf{r}_4, \mathbf{r}_1) = (y_4 \pm \delta_4 - y_3 \pm \delta_3)(x_1 - x_4 \pm \delta_4) - (y_4 - y_3)(x_1 - x_4) \tag{12}$$

$$+ (x_4 \pm \delta_4 - x_3 \pm \delta_3)(y_1 - y_4 \pm \delta_4) - (x_4 - x_3)(y_1 - y_4) \tag{13}$$

$$\leq (\delta_3 + \delta_4)(|x_1 - x_4| + |y_1 - y_4| + 2\delta_4) + \delta_4(|y_4 - y_3| + |x_4 - x_3|), \tag{14}$$

similarly

$$\delta O(\mathbf{r}_3, \mathbf{r}_4, \mathbf{r}_2) \le (\delta_3 + \delta_4)(|x_2 - x_4| + |y_2 - y_4| + 2\delta_4) + \delta_4(|y_4 - y_3| + |x_4 - x_3|). \quad (15)$$

These error bounds allow us to prove certain voxels' unavailability: If Eqs. (1)-(2) hold, and if each O function's error bound is smaller than its absolute value, then we know these O functions cannot change sign no matter how x, y, and θ vary within their respective limit, and Eqs. (1)-(2) will always hold. We thus proved that the voxel cannot contain available space.

It is noteworthy that the errors could be smaller than the worse-case bounds we derived. Thus, a completely unavailable voxel could be miscategorized as an available one. This is nevertheless not a problem for two reasons: First, such miscategorization can cause us to retain unavailable voxels but can never cause us to discard available ones. Second, as we repeatedly divide the voxels and drive all error bounds to zero, such miscategorization will eventually disappear. The ending configuration is thus still guaranteed to be saturated.

C.   an unexpected problem and its solution

With the aforementioned subroutines supplementing the split-voxel algorithm, we are ready to generate saturated RSA configurations of 2D polygons. However, in doing so for 2D squares and regular hexagons, we found an unexpected problem: the number of voxels occasionally grows to extremely large numbers ($> 10^8$) for moderately-sized systems (≈ 3000 particles). Sometimes the number of voxels suddenly drops to zero after becoming extremely large, but sometimes our program crashes due to insufficient memory before the drop could happen. This is in contrast with the situation for disks, equilateral triangles, and regular pentagons, where the number of voxels always decays smoothly as they are split.

To understand its reason, we plotted 10 randomly selected voxel centers when the problem happened in Fig. 1. Surprisingly, all of the selected voxels are concentrated in a very small part of the configuration. In Fig. 1, the distance between points A and B is 0.999 992 times the side length of a square. Therefore, inserting a new square at this place is impossible. Nevertheless, the algorithm could not realize this impossibility until voxel-space resolution becomes extremely fine. Figure 1 also indicates that this problem can only occur when the polygon has at least one pair of parallel sides, and therefore explains why we only observed
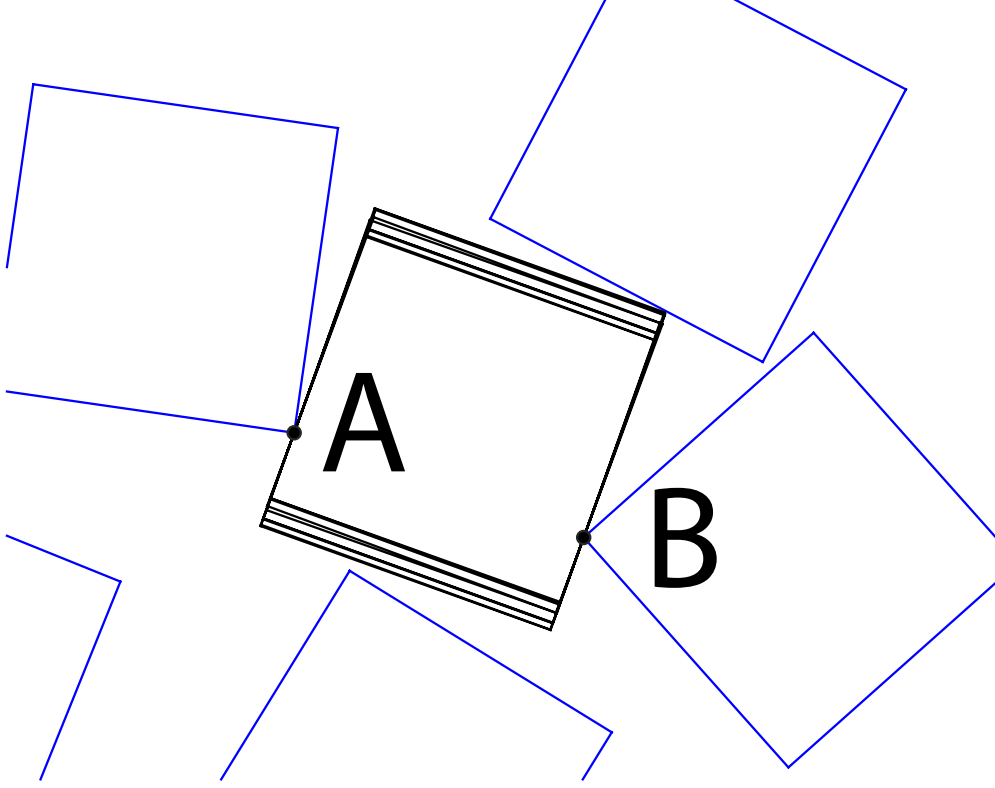
6

FIG. 1. Plot of 10 randomly selected voxel centers when the voxel-number-explosion problem happened when generating a saturated RSA packing of squares. Note that a voxel center has a three-dimensional coordinate (x, y, θ), and represents a trial insertion at location (x, y) and orientation θ. Hence, we can use a black square to represent a voxel center. Blue squares are adjacent existing particles. The distance between points A and B is 0.999 992 times the side length of a square. Therefore, inserting a new square at the place indicated by these voxels is impossible.

this problem only for certain shapes.

Our solution to this problem is to run very deep tests of voxel availabilities. Specifically, we define a level-0 test of voxel availability as our original test outlined in Sec. II B. We define a level-n $(n > 0)$ test of voxel availability as:

- If the voxel can be proved unavailable with the procedure outlined in Sec. II B, then declare the voxel unavailable.

- Otherwise, if the voxel center represents an incoming particle that does not overlap with existing particles, declare the voxel available.

- Otherwise, divide the voxel into 2^3 subvoxels, run a level-$(n-1)$ test of each subvoxel. If any subvoxel is available, declare the original voxel available.

- Otherwise, declare the original voxel unavailable.

The deep test retains a desired property of the original voxel availability test: an unavailable voxel may be misjudged as an available one if n is finite, but an available voxel will never be misjudged as an unavailable one. Therefore, one can safely employ a deep test on voxels and remove unavailable ones, without worrying about discarding any available space. If the particle has a pair of parallel sides, we randomly sample 100 voxels each time a voxel list is generated. If at least 50 of them are within a distance of R_{ins}, the radius of the inscribed circle of a particle, then this problem is suspected. We run a level-4 check on all voxels and discard unavailable ones. If all of the sampled voxels are within a distance of R_{ins}, then this problem is strongly suspected, and we run a level-12 check on all voxels and discard unavailable ones.

The deep test successfully solves this problem but can be very time consuming. To illustrate this point, we show the histogram of the time taken to generate a saturated RSA configuration of enneagons and decagons in Fig. 2. Only decagons are susceptible to this problem. The time distribution for decagons resembles that for enneagons, except that the former also exhibit a very long tail on the long-time side, corresponding to the extra time needed for the deep test.

One could argue that as Fig. 1 shows, this problem only happens in saturated locations. Thus, the simplest "solution" would be to just declare the configuration saturated whenever the problem is detected. If a strict proof of saturation is not required, this simple "solution" may be desirable, especially since deep testing voxel availability is very time-consuming. However, we choose the deep-test solution since in this work, we want to rigorously guarantee that each configuration is saturated.

III.   RESULTS AND DISCUSSION

To demonstrate the correctness and usefulness of this algorithm, we generate saturated RSA configurations of regular polygons, and compare ϕ_s with previous results. For each particle shape, we generate 1000 configurations with system size $L_{\text{particle}}/L_{\text{box}} = 0.01, 100$
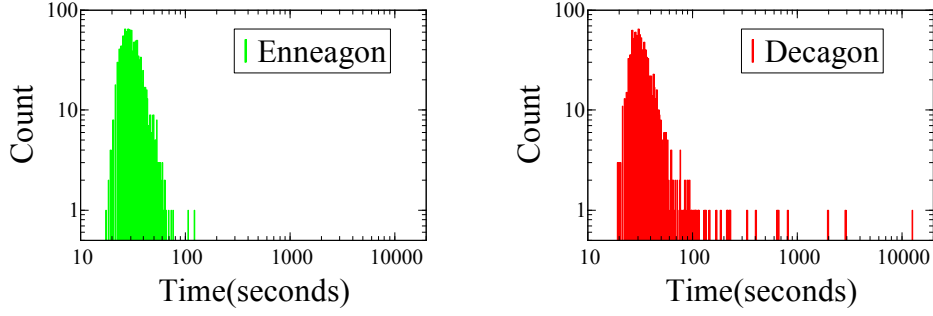
FIG. 2. Histogram of the time taken to generate an RSA configuration of (left) enneagons and (right) decagons of size $L_{\mathrm{particle}}/L_{\mathrm{box}} = 0.01$ on a computer with an Intel Xeon E5-2696 v3 central processing unit.

TABLE I. Saturation density ϕ_s for various particle shapes and system sizes. Here all error estimates are calculated from $\sigma = \sqrt{(< \phi_s^2 > - < \phi_s >^2)/(N_c - 1)}$, where $< \cdots >$ indicates averaging over all configurations, and N_c is the number of configurations.

| $L_{\mathrm{particle}}/L_{\mathrm{box}}$ | ϕ_s for | | | |
	Equilateral triangles	Squares	Regular pentagons	Regular hexagons
0.01	0.525892 ± 0.000064	0.527719 ± 0.000085	0.541319 ± 0.000087	0.539114 ± 0.000093
0.003	0.525993 ± 0.000058	0.527482 ± 0.000080	0.541241 ± 0.000088	0.539216 ± 0.000087
0.001	0.525820 ± 0.000066	0.527594 ± 0.000070	0.541344 ± 0.000072	0.539060 ± 0.000095
$L_{\mathrm{particle}}/L_{\mathrm{box}}$	ϕ_s for			
	Regular heptagons	Regular octagons	Regular enneagons	Regular decagons
0.01	0.542143 ± 0.000093	0.542329 ± 0.000094	0.544055 ± 0.000092	0.544104 ± 0.000094
0.003	0.542197 ± 0.000093	0.542494 ± 0.000090	0.544044 ± 0.000098	0.544278 ± 0.000100
0.001	0.541959 ± 0.000124	0.542328 ± 0.000098	0.544059 ± 0.000089	0.544259 ± 0.000124

configurations with $L_{\mathrm{particle}}/L_{\mathrm{box}} = 0.003$, and 10 configurations with $L_{\mathrm{particle}}/L_{\mathrm{box}} = 0.001$. Here L_{particle} is the distance between a particle's center and its vertex, and L_{box} is the side length of the simulation box. The resulting saturation density is summarized in Table I.

For all shapes, the difference in ϕ_s between different system sizes is comparable to the

error estimate, suggesting negligible finite-size effect. Indeed, this is consistent with Ref. 30, which found minimal finite-size effect for even smaller ($L_\text{particle}/L_\text{box} \approx 0.2$) saturated RSA configurations with periodic boundary configurations. This is also expected in light of Ref. 31, which found that the pair correlation functions of RSA configurations decay super-exponentially. With negligible finite-size effect, we think the best estimate of ϕ_s for each shape can be obtained by simply averaging the results for different system sizes. This yields $\phi_s = 0.525902 \pm 0.000036$ for triangles, $\phi_s = 0.527598 \pm 0.000045$ for squares, $\phi_s = 0.541301 \pm 0.000047$ for pentagons, $\phi_s = 0.539130 \pm 0.000053$ for hexagons, $\phi_s = 0.542100 \pm 0.000060$ for heptagons, $\phi_s = 0.542383 \pm 0.000054$ for octagons, $\phi_s = 0.544053 \pm 0.000054$ for enneagons, and $\phi_s = 0.544214 \pm 0.000062$ for decagons.

Previous researches by extrapolating finite-time RSA densities found the saturation density of squares to be $0.523 - 0.532$ [14] and 0.530 ± 0.001 (reported in both [17] and [18]). Our result is within the former range but slightly below the latter (by two and a half of their error bar). Could this indicate a mistake, for example, one that leads to the generation of unsaturated configurations? One way to double-check is to calculate RSA saturation densities of regular n-gons with large n, since as n increases, ϕ_s should approach that for disks, $0.547067 \cdots$ [30]. We thus calculated ϕ_s for 19-gons and 29-gons, and found 0.546210 ± 0.000080 and 0.546701 ± 0.000067, respectively. These densities indeed approach ϕ_s for disks, and thus does not suggest the existence of such a mistake.

We plot ϕ_s versus the number of sides of the polygon in Fig. 3, which shows that ϕ_s increases as the number of sides increases except that ϕ_s for hexagons is lower than that for pentagons. More generally, ϕ_s tend to be slightly higher than the trend when the number of sides is odd, and slightly lower otherwise. Overall, our results appear to be consistent with Ref. 19, which plotted (but not listed) ϕ_s for regular polygons obtained by infinite-time extrapolation.

IV. CONCLUSIONS

To summarize, we have developed in this paper a generalization of the split-voxel algorithm described in Ref. 13, based on worst-case error analysis method. We support the correctness of this method by finding the RSA saturation densities of 2D regular polygons with three to ten sides, and verifying their consistency with previous results.
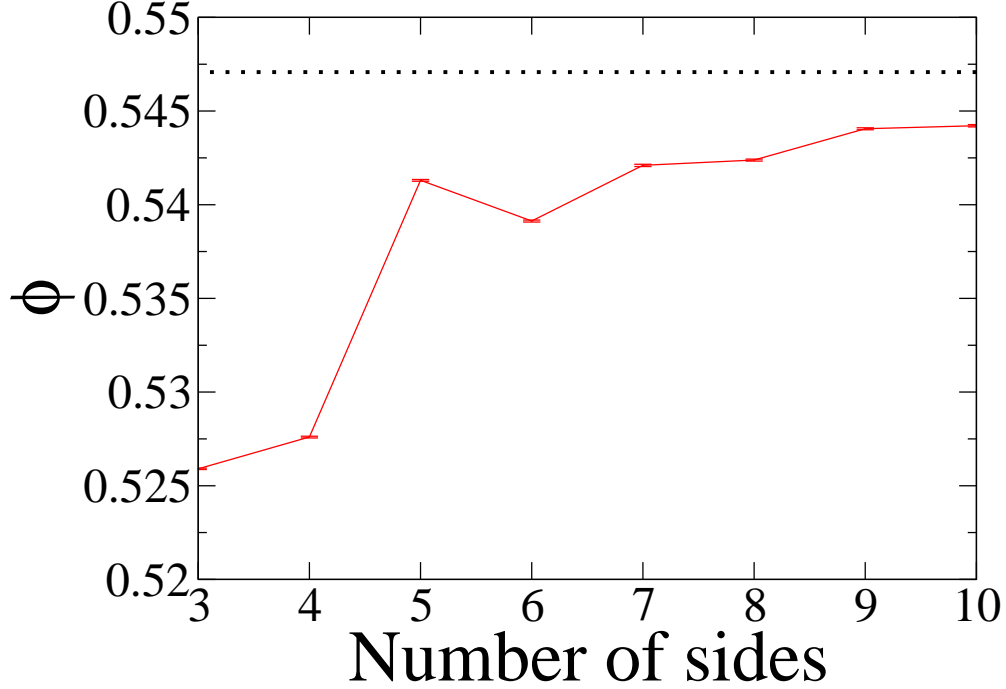
FIG. 3. (red solid line) RSA saturation density for regular polygons, as a function of the number of sides. Note that each data point is associated with an error bar that is barely visible. (black dotted line) RSA saturation density for disks.

A program implementing this algorithm is available as supplementary material [32].

ACKNOWLEDGMENTS

[1] J. Feder, J. Theor. Biol. **87**, 237 (1980).

[2] B. Widom, J. Chem. Phys. **44**, 3888 (1966).

[3] W. X. Xu and H. S. Chen, Computers & Structures **114**, 35 (2013).

[4] E. Roman and N. Majlis, Solid State Commun. **47**, 259 (1983).

[5] J. Feder and I. Giaever, J. Colloid Interface Sci. **78**, 144 (1980).

[6] L. Finegold and J. T. Donnell, Nature **278**, 443 (1979).

[7] M. Tanemura and M. Hasegawa, J. Theor. Biol. **82**, 477 (1980).

[8] A. Renyi, Sel. Trans. Math. Stat. and Prob. **4**, 203 (1963).

[9] D. W. Cooper, J. Colloid Interface Sci. **119**, 442 (1987).

[10] J. S. Wang, Int. J. Mod. Phys. C **5**, 707 (1994).

[11] J. S. Wang, Physica A **254**, 179 (1998).

[12] S. Torquato, O. U. Uche, and F. H. Stillinger, Phys. Rev. E **74**, 061308 (2006).

[13] G. Zhang and S. Torquato, Phys. Rev. E **88**, 053312 (2013).

[14] R. D. Vigil and R. M. Ziff, J. Chem. Phys. **91**, 2599 (1989).

[15] R. D. Vigil and R. M. Ziff, J. Chem. Phys. **93**, 8270 (1990).

[16] B. J. Brosilow, R. M. Ziff, and R. D. Vigil, Phys. Rev. A **43**, 631 (1991).

[17] P. Viot and G. Tarjus, Europhys. Lett. **13**, 295 (1990).

[18] P. Viot, G. Tarjus, S. M. Ricci, and J. Talbot, J. Chem. Phys. **97**, 5212 (1992).

[19] M. Cieśla and J. Barbasz, Phys. Rev. E **90**, 022402 (2014).

[20] J. Talbot, G. Tarjus, and P. Schaaf, Phys. Rev. A **40**, 4808 (1989).

[21] J. D. Sherwood, J. Phys. A **23**, 2827 (1999).

[22] M. Cieśla, G. Pająk, and R. M. Ziff, Phys. Chem. Chem. Phys. **17**, 24376 (2015).

[23] M. Cieśla and P. Kubala, J. Chem. Phys. **148**, 024501 (2018).

[24] J. D. Sherwood, J. Phys. A **30**, L839 (1999).

[25] O. Gromenko and V. Privman, Phys. Rev. E **79**, 042103 (2009).

[26] M. Cieśla and J. Barbasz, Surf. Sci. **612**, 24 (2013).

[27] M. Cieśla, Phys. Rev. E **87**, 052401 (2013).

[28] B. E. Blaisdell and H. Solomon, J. Appl. Probab. **19**, 382 (1982).

[29] "http://www.dcs.gla.ac.uk/ pat/52233/slides/geometry1x1.pdf,".

[30] M. Cieśla and R. M. Ziff, J. Stat. Mech. Theor. Exp. **4**, 043302 (2018).

[31] B. Bonnier, D. Boyer, and P. Viot, J. Phys. A: Math. Gen. **27**, 3671 (1994).

[32] See Supplemental Material at [URL will be inserted by publisher] for the program.