

Self-avoiding polygons on the square lattice

I Jensen and A J Guttmann

Department of Mathematics & Statistics, The University of Melbourne
Parkville, Vic. 3052, Australia

May 21, 1999

Abstract

We have developed an improved algorithm that allows us to enumerate the number of self-avoiding polygons on the square lattice to perimeter length 90. Analysis of the resulting series yields very accurate estimates of the connective constant $\mu = 2.63815852927(1)$ (biased) and the critical exponent $\alpha = 0.5000005(10)$ (unbiased). The critical point is indistinguishable from a root of the polynomial $581x^4 + 7x^2 - 13 = 0$. An asymptotic expansion for the coefficients is given for all n . There is strong evidence for the absence of any non-analytic correction-to-scaling exponent.

1 Introduction

A self-avoiding polygon (SAP) can be defined as a walk on a lattice which returns to the origin and has no other self-intersections. The history and significance of this problem is nicely discussed in [8]. Alternatively we can define a SAP as a sub-graph (of a lattice) whose vertices are of degree 0 or 2. Generally SAPs are considered distinct up to a translation, so if there are p_n SAPs of length n there are $2np_n$ walks (the factor of two arising since the walk can go in two directions). The enumeration of self-avoiding polygons on various lattices is an interesting combinatorial problem in its own right, and is also of considerable importance in the statistical mechanics of lattice models [8].

The basic problem is the calculation of the generating function

$$P(x) = \sum_n p_{2n} x^{2n} \sim A(x) + B(x)(1 - x^2/x_c^2)^{2-\alpha}, \quad (1)$$

where the functions A and B are believed to be regular in the vicinity of x_c . We discuss this point further in Sec. 3, as it pertains to the presence or otherwise of a non-analytic correction-to-scaling term. Despite strenuous effort over the past 50 years or so this problem has not been solved on any regular two dimensional lattice. However, for the hexagonal lattice the critical point, $x_c^2 = 1/(2 + \sqrt{2})$ as well as the critical exponent $\alpha = 1/2$ are known exactly [10], though non-rigorously. Very firm evidence exists from previous numerical work that the exponent α is universal and thus equals $1/2$ for all two dimensional lattices [6, 5, 9]. Thus the major remaining problem, short of an exact solution, is the calculation of x_c for various lattices. Recently the authors found a simple

mapping between the generating function for SAPs on the hexagonal lattice and the generating function for SAPs on the (3.12^2) lattice [9]. Knowledge of the exact value for x_c on the hexagonal lattice resulted in the exact determination of the critical point on the (3.12^2) lattice.

In order to study this and related systems, when an exact solution can't be found one has to resort to numerical methods. For many problems the method of series expansions is by far the most powerful method of approximation. For other problems Monte Carlo methods are superior. For the analysis of $P(x)$, series analysis is undoubtedly the most appropriate choice. This method consists of calculating the first coefficients in the expansion of the generating function. Given such a series, using the numerical technique known as differential approximants [7], highly accurate estimates can frequently be obtained for the critical point and exponents, as well as the location and critical exponents of possible non-physical singularities.

This paper builds on the work of Enting [3] who enumerated square lattice polygons to 38 steps using the finite lattice method. Using the same technique this enumeration was extended by Enting and Guttmann to 46 steps [4] and later to 56 steps [6]. Since then they extended the enumeration to 70 steps in unpublished work. These extensions to the enumeration were largely made possible by improved computer technology. In this work we have improved the algorithm and extended the enumeration to 90 steps while using essentially the same computational resources used to obtain polygons to 70 steps.

The difficulty in the enumeration of most interesting lattice problems is that, computationally, they are of exponential complexity. It would be a great breakthrough if a polynomial time algorithm could be found, while a linear time algorithm is, to all intents and purposes, equivalent to an exact solution. Initial efforts at computer enumeration of square lattice polygons were based on direct counting. The computational complexity was proportional to λ_1^n , where n is the length of the polygon, and $\lambda_1 \approx 2.638$. The dramatic improvement achieved [3] by the finite lattice method can be seen from its complexity, which is proportional to λ_2^n , where $\lambda_2 = 3^{\frac{1}{4}} \approx 1.316$. Our new algorithm, described below, has reduced both time and storage requirements by virtue of a complexity which is proportional to λ_3^n , where $\lambda_3 \approx 1.20$.

In the next section we will very briefly review the finite lattice method for enumerating square lattice polygons and give some details of the improved algorithm. The results of the analysis of the series are presented in Section 3 including a detailed discussion of a conjecture for the exact critical point.

2 Enumeration of polygons

The method used to enumerate SAP on the square lattice is an enhancement of the method devised by Enting [3] in his pioneering work. The first terms in the series for the polygon generating function can be calculated using transfer matrix techniques to count the number of polygons in rectangles $W + 1$ edges wide and $L + 1$ edges long. The transfer matrix technique involves drawing a line through the rectangle intersecting a set of $W + 2$ edges. For each configuration of occupied or empty edges along the intersection we maintain a (perimeter) generating function for loops to the left of the line cutting the intersection in that particular pattern. Polygons in a given rectangle are enumerated by moving the intersection so as to add one vertex at a time, as shown in Fig. 1. The

allowed configurations along the intersection are described in [3]. Each configuration can be represented by an ordered set of edge states $\{n_i\}$, where

$$n_i = \begin{cases} 0 & \text{empty edge,} \\ 1 & \text{lower part of loop closed to the left,} \\ 2 & \text{upper part of loop closed to the left.} \end{cases} \quad (2)$$

Configurations are read from the bottom to the top. So the configuration along the intersection of the polygon in Fig. 1 is $\{0112122\}$.

The rules for updating the partial generating functions as the intersection is moved are identical to the original work, so we refer the interested reader to [3] for further details regarding this aspect of the transfer matrix calculation.

Due to the obvious symmetry of the lattice one need only consider rectangles with $L \geq W$. Valid polygons were required to span the enclosing rectangle in the lengthwise direction. So it is clear that polygons with projection on the y -axis $< W$, that is polygons which are narrower than the width of the rectangle, are counted many times. It is however easy to obtain the polygons of width exactly W and length exactly L from this enumeration [3]. Any polygon spanning such a rectangle has a perimeter of length at least $2(W + L)$. By adding the contributions from all rectangles of width $W \leq W_{\max}$ (where the choice of W_{\max} depends on available computational resources, as discussed below) and length $W \leq L \leq 2W_{\max} - W + 1$, with contributions from rectangles with $L > W$ counted twice, the number of polygons per vertex of an infinite lattice is obtained correctly up to perimeter $4W_{\max} + 2$.

The major improvement of the method used to enumerate polygons in this paper is that we require valid polygons to span the rectangle in *both* directions. In other words we directly enumerate polygons of width exactly W and length L rather than polygons of width $\leq W$ and length L as was done originally. The only drawback of this approach is that for most configurations we have to use four distinct generating functions since the partially completed polygon could have reached neither, both, the lower, or the upper boundaries of the rectangle. The major advantage is that the memory requirement of the algorithm is exponentially smaller.

Realizing the full savings in memory usage requires two enhancements to the original algorithm. Firstly, for each configuration we must keep track of the current minimum number of steps N_{cur} that have been inserted to the left of the intersection in order to build up that particular configuration. Secondly, we calculate the minimum number of additional steps N_{add} required to produce a valid polygon. There are three contributions, namely the number of steps required to close the polygon, the number of steps needed (if any) to ensure that the polygon touches both the lower and upper boundary, and finally the number of steps needed (if any) to extend at least W edges in the length-wise direction. If the sum $N_{\text{cur}} + N_{\text{add}} > 4W_{\max} + 2$ we can discard the partial generating function for that configuration because it won't make a contribution to the polygon count up to the perimeter lengths we are trying to obtain. For instance polygons spanning a rectangle with a width close to W_{\max} have to be almost convex, so very convoluted polygons are not possible. Thus configurations with many loop ends (non-zero entries) make no contribution at perimeter length $\leq 4W_{\max} + 2$.

The number of steps needed to ensure a spanning polygon is straightforward to calculate. The complicated part of the new approach is the algorithm to calculate the number of steps required to close the polygon. There are very many special cases depending on

the position of the kink in the intersection and whether or not the partially completed polygon has reached the upper or lower boundary of the bounding rectangle. So in the following we will only briefly describe some of the simple contributions to the closing of a polygon. Firstly, if the partial polygon contains separate pieces these have to be connected as illustrated in Fig. 2. Separate pieces are easy to locate since all we have to do is start at the bottom of the intersection and moving upwards we count the number of 1's and 2's in the configuration. Whenever these numbers are equal a separate piece has been found and (provided one is not at the last edge in the configuration) the currently encountered 2-edge can be connected to the next 1-edge above. N_{add} is incremented by the number of steps (the distance) between the edges and the two edge-states are removed from the configuration before further processing. It is a little less obvious that if the configuration start (end) as $\{112\dots 2\}$ ($\{1\dots 122\}$) the two lower (upper) edges can safely be connected (note that there can be any number of 0's interspersed before the ...). Again N_{add} is incremented by the number of steps between the edges, and the two edge-states are removed from the configuration – leading to the new configuration $\{001\dots 2\}$ ($\{1\dots 200\}$) – before further processing. After these operations we may be left with a configuration which has just one 1- and one 2-edge, in which case we are done since these two edges can be connected to form a valid polygon. This is illustrated in Fig. 2 where the upper left panel shows how to close the partial polygon with the intersection $\{12112212\}$, which contain three separate pieces. After connecting these pieces we are left with the configuration $\{10012002\}$. We now connect the two 1-edges and note that the first two-edge is relabeled to a 1-edge (it has become the new lower end of the loop). Thus we get the configuration $\{00001002\}$ and we can now connect the remaining two edges and end up with a valid completed polygon. Note that in the last two cases, in addition to the steps spanning the distance between the edges, an additional two horizontal steps had to be added in order to form a valid loop around the intervening edges. If the transformation above doesn't result in a closed polygon we must have a configuration of the form $\{111\dots 222\}$. The difficulty lies in finding the way to close such configurations with the smallest possible number of additional steps. Suffice to say that if the number of non-zero entries is small one can easily devise an algorithm to try all possible valid ways of closing a polygon and thus find the minimum number of additional steps. In Fig. 2 we show all possible ways of closing polygons with 8 non-zero entries. Note that we have shown the generic cases here. In actual cases there could be any number of 0-edges interspersed in the configurations and this would determine which way of closing would require the least number of additional steps.

With the original algorithm the number of configurations required as W_{max} increased grew asymptotically as $3^{W_{\text{max}}}$ [6]. Our enumerations indicate that the computational complexity is reduced significantly. While the number of configurations still grows exponentially as $\lambda^{W_{\text{max}}}$ the value of λ is reduced from $\lambda = 3$ to $\lambda \simeq 2$ with the improved algorithm (Fig. 3 shows the number of configuration required as W_{max} increases). Furthermore, for any W we know that contributions will start at $4W$ since the smallest polygons have to span a $W \times W$ rectangle, so for each configuration we need only retain $4(W_{\text{max}} - W) + 2$ terms of the generating functions while in the original algorithm contributions started at $2W$ because the polygons were required to span only in the length-wise direction. Combining these improvements allows us to extend the series for the square lattice polygon generating function from 70 terms to 90 terms using the same computational resources. Obtaining a series this long with the original algorithm would have required at least 200

times as much memory, or close to a terabyte!

In Table 1 we have listed the new terms obtained in this work. The number of polygons of length ≤ 56 can be found in [6].

3 Analysis of the series

We analyzed the series for the polygon generating function by the numerical method of differential approximants [7]. In Table 2 we have listed estimates for the critical point x_c^2 and exponent $2 - \alpha$ of the series for the square lattice SAP generating function. The estimates were obtained by averaging values obtained from first order $[L/N; M]$ and second order $[L/N; M; K]$ inhomogeneous differential approximants. For each order L of the inhomogeneous polynomial we averaged over those approximants to the series which used at least the first 35 terms of the series (that is, polygons of perimeter at least 74), and used approximants such that the difference between N , M , and K didn't exceed 2. These are therefore "diagonal" approximants. Some approximants were excluded from the averages because the estimates were obviously spurious. The error quoted for these estimates reflects the spread (basically one standard deviation) among the approximants. Note that these error bounds should *not* be viewed as a measure of the true error as they cannot include possible systematic sources of error. We discuss further the systematic error when we consider biased approximants. Based on these estimates we conclude that $x_c^2 = 0.1436806289(5)$ and $\alpha = 0.5000005(10)$.

As stated earlier there is very convincing evidence that the critical exponent $\alpha = 1/2$ exactly. If we assume this to be true we can obtain a refined estimate for the critical point x_c^2 . In Fig. 4 we have plotted estimates for the critical exponent $2 - \alpha$ against estimates for the critical point x_c^2 . Each dot in this figure represents a pair of estimates obtained from a second order inhomogeneous differential approximant. The order of the inhomogeneous polynomial was varied from 0 to 10. We observe that there is an almost linear relationship between the estimates for $2 - \alpha$ and x_c^2 and that for $2 - \alpha = 3/2$ we get $x_c^2 \simeq 0.14368062928\dots$. In order to get some idea as to the effect of systematic errors, we carried out this analysis using polygons of length up to 60 steps, then 70, then 80 and finally 90 steps. The results were $x_c^2 = 0.1436806308$ for $n = 60$, $x_c^2 = 0.14368062956$ for $n = 70$, $x_c^2 = 0.14368062930$ for $n = 80$, and $x_c^2 = 0.14368062928$ for $n = 90$. This is a rapidly converging sequence of estimates, though we have no theoretical basis that would enable us to assume any particular rate of convergence. However, observing that the differences between successive estimates are decreasing by a factor of at least 5, it is not unreasonably optimistic to estimate the limit at $x_c^2 = 0.14368062927(1)$.

This leads to our final estimate $x_c^2 = 0.14368062927(1)$ and thus we find the connective constant $\mu = 1/x_c = 2.63815853034(10)$. It is interesting to note that some years ago we [1] pointed out that since the hexagonal lattice connective constant is given by the zero of a quadratic in x^2 , it is plausible that this might be the case also for the square lattice connective constant. On the basis of an estimate of the connective constant that was 4 orders of magnitude less precise, we pointed out then that the polynomial

$$581x^4 + 7x^2 - 13 = 0$$

was the only polynomial we could find with "small" integer coefficients consistent with our estimate. The relevant zero of this polynomial is 0.1436806292698685.. in complete

agreement with our new estimate — which, as noted above, contains four more significant digits! Unfortunately the other zero is at $x_c^2 = -0.1557288\dots$, and we see no evidence of such a singularity. Nevertheless, the agreement is so astonishingly good that we are happy to take this as a good algebraic approximation to the connective constant. An argument as to why we might not expect to see the singularity on the negative real axis from our series analysis would make the root of the above polynomial a plausible conjecture for the exact value, but at present such an argument is missing.

Two further analyses were carried out on the data. Firstly, a study of the location of non-physical singularities, and secondly, a study of the asymptotic form of the coefficients — which is relevant to the identification of any correction-to-scaling exponent. Singularities outside the radius of convergence give exponentially small contributions to the asymptotic form of the coefficients, so are notoriously hard to analyse. Nevertheless, we see clear evidence of a singularity on the negative real axis at $x^2 \approx -0.40$ with an exponent that is extremely difficult to analyse but could be 1.5, in agreement with the physical exponent. There is weaker evidence of a further conjugate pair of singularities. First order approximants locates these at $-0.015 \pm 0.36i$, while second order approximants locates them at $-0.035 \pm 0.31i$. There is also evidence of a further singularity on the negative real axis at $x_c^2 = -0.7$. We are unable to give a useful estimate of the exponents of these singularities.

We turn now to the asymptotic form of the coefficients. We have argued previously [2] that there is no non-analytic correction-to-scaling exponent for the polygon generating function. This is entirely consistent with Nienhuis's [10] observation that there is a correction-to-scaling exponent of $\Delta = \frac{3}{2}$. Since for the polygon generating function exponent $\alpha = \frac{1}{2}$, the correction term has an exponent equal to a positive integer, and therefore “folds into” the analytic background term, denoted $A(x)$ in Eqn.(1). This is explained in greater detail in [2]. We assert that the asymptotic form for the polygon generating function is as given by Eqn.(1) above. In evidence of this, we remark that from (1) follows the asymptotic form

$$p_{2n}x_c^{2n} \sim n^{-\frac{5}{2}}[a_1 + a_2/n + a_3/n^2 + a_4/n^3 + \dots]. \quad (3)$$

Using our algebraic approximation to x_c quoted above, we show in Table 3 the estimates of the amplitudes a_1, \dots, a_4 . From the table we see that $a_1 \approx 0.0994018$, $a_2 \approx -0.02751$, $a_3 \approx 0.0255$ and $a_4 \approx 0.12$, where in all cases we expect the error to be confined to the last quoted digit. The excellent convergence of all columns is strong evidence that the assumed asymptotic form is correct. If we were missing a term corresponding to, say, a half-integer correction, the fit would be far worse. This is explained at greater length in [2]. So good is the fit to the data that if we take the last entry in the table, corresponding to $n = 45$, and use the entries as the amplitudes, then $p_4 \dots p_{16}$ are given exactly by the above asymptotic form (provided we round to the nearest integer), and beyond perimeter 20 all coefficients are given to the same accuracy as the leading amplitude.

Finally, to complete our analysis, we estimate the critical amplitudes $A(x_c^2)$ and $B(x_c^2)$, defined in Eqn.(1). $A(x_c^2)$ has been estimated by evaluating Padé approximants to the generating function, evaluated at x_c^2 . In this way we estimate $A(x_c^2) \approx 0.036$, while $B(x_c^2)$ follows from the estimate of a_1 in Eqn.(3), since $B(x_c^2) = \frac{4\sqrt{\pi}a_1}{3} \approx 0.234913$.

4 Conclusion

We have presented an improved algorithm for the enumeration of self-avoiding polygons on the square lattice. The computational complexity of the algorithm is estimated to be 1.2^n . Implementing this algorithm has enabled us to obtain polygons up to perimeter length 90. Decomposing the coefficients into prime factors reveals frequent occurrence of very large prime factors, supporting the widely held view that there is no “simple” formula for the coefficients. For example, p_{78} contains the prime factor 7789597345683901619. Our extended series enables us to give an extremely precise estimate of the connective constant, and we give a simple algebraic approximation that agrees precisely with our numerical estimate. An alternative analysis provides very strong evidence for the absence of any non-analytic correction terms to the proposed asymptotic form for the generating function. Finally we give an asymptotic representation for the coefficients which we believe to be accurate for all positive integers.

5 Acknowledgments

We gratefully acknowledge valuable discussions with Ian Enting and financial support from the Australian Research Council.

References

- [1] A. R. Conway and A. J. Guttmann, *J. Phys. A.* **26**, 1519 (1993).
- [2] A. R. Conway and A. J. Guttmann, *Phys. Rev. Letts.* **77**, 5284 (1996).
- [3] I. G. Enting, *J. Phys. A.* **13**, 3713 (1980).
- [4] I. G. Enting and A. J. Guttmann, *J. Phys. A.* **18**, 1007 (1985).
- [5] I. G. Enting and A. J. Guttmann, *J. Phys. A.* **22**, 1371 (1989).
- [6] A. J. Guttmann and I. G. Enting, *J. Phys. A.* **21**, L165 (1988).
- [7] A. J. Guttmann, in *Phase Transitions and Critical Phenomena*, Vol. 13, eds. C Domb and J L Lebowitz, Academic Press, New York (1989).
- [8] B. D. Hughes, in *Random walks and random environments, Vol. I Random walks*, Clarendon Press, Oxford (1995).
- [9] I. Jensen and A. J. Guttmann, *J. Phys. A.* **31**, 8137 (1998).
- [10] B. Nienhuis, *Phys. Rev. Letts.* **49**, 1062 (1982).

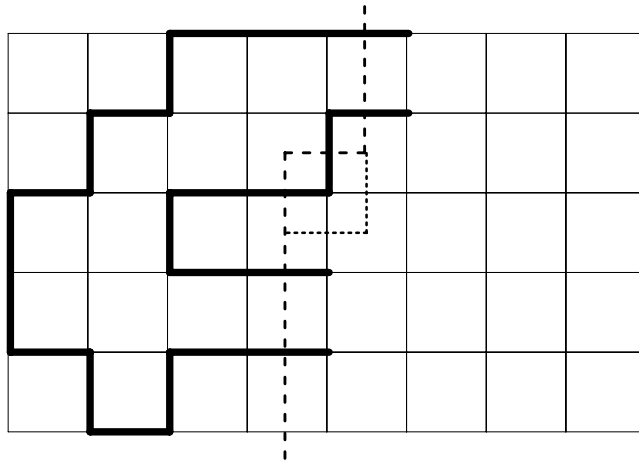


Figure 1: A snapshot of the intersection (dashed line) during the transfer matrix calculation on the square lattice. Polygons are enumerated by successive moves of the kink in the intersection, as exemplified by the position given by the dotted line, so that one vertex at a time is added to the rectangle. To the left of the intersection we have drawn an example of a partially completed polygon.

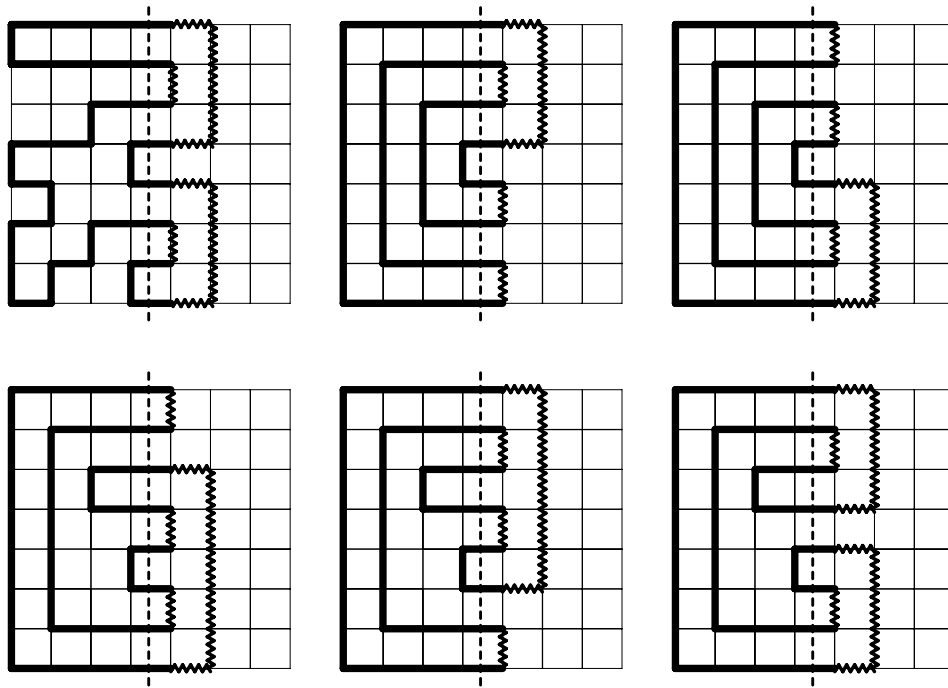


Figure 2: Examples of partially generated polygons (thick solid lines) to the left of the intersection (dashed line) and how to close them in a valid way (thick wavy line). Upper left panel shows how to close the configuration $\{12112212\}$. The upper middle and right panels show the two possible closures of the configuration $\{11112222\}$. The lower panels show the three possible closures of the configuration $\{11121222\}$.

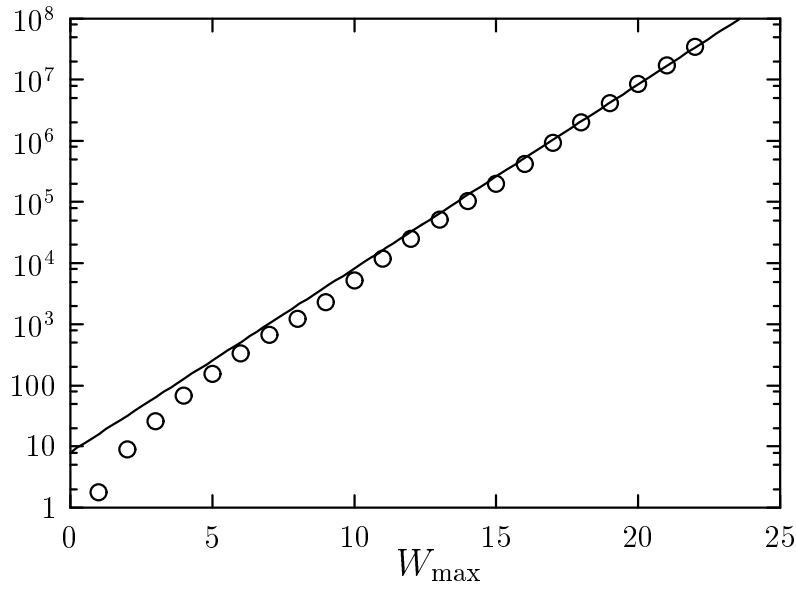


Figure 3: The number of configurations required as W_{\max} is increased. The straight line corresponds to a growth factor $\lambda = 2$.

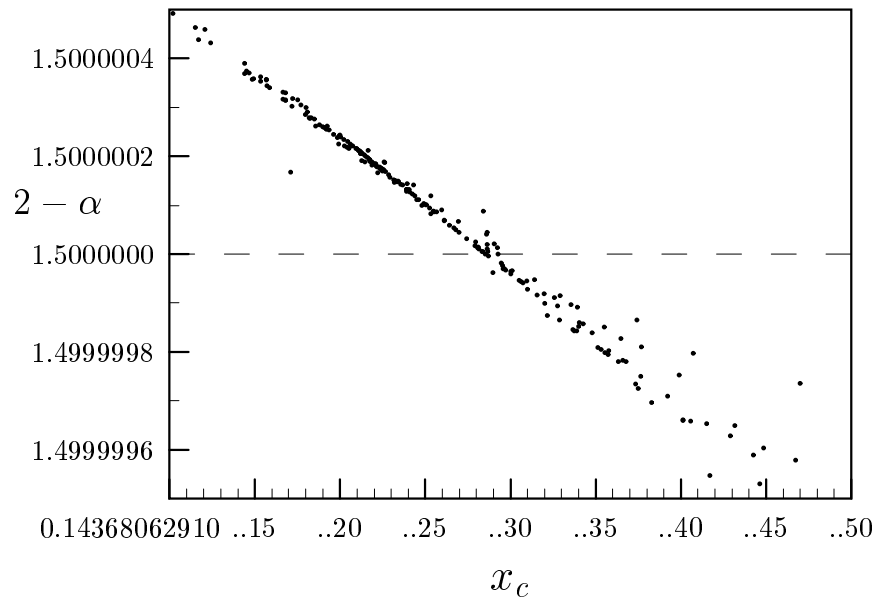


Figure 4: Estimates for the critical exponent $2 - \alpha$ vs. estimates for the critical point x_c^2 of the square lattice polygon generating function.

Table 1: The number, x_n , of embeddings of n -step polygons on the square lattice. Only non-zero terms are listed.

n	x_n	n	x_n
58	59270905595010696944	76	1158018792676190545425711414
60	379108737793289505364	78	7554259214694896127239818088
62	2431560774079622817356	80	49360379260931646965916677280
64	15636142410456687798584	82	323028185951187646733521902740
66	100792521026456246096640	84	2117118644744425875029583096670
68	651206027727607425003232	86	13895130612692826326409919713700
70	4216407618470423070733556	88	91319729650588816198004801698400
72	27355731801639756123505014	90	600931442757555468862970353941700
74	177822806050324126648352460		

Table 2: Estimates for the critical point x_c^2 and exponent $2 - \alpha$ obtained from first and second order inhomogeneous differential approximants to the series for square lattice polygon generating function. L is the order of the inhomogeneous polynomial.

L	First order DA		Second order DA	
	x_c^2	$2 - \alpha$	x_c^2	$2 - \alpha$
1	0.14368062897(17)	1.50000074(35)	0.14368062883(45)	1.50000092(92)
2	0.14368062902(14)	1.50000068(26)	0.14368062943(29)	1.49999957(80)
3	0.14368062878(35)	1.50000107(71)	0.14368062914(20)	1.50000034(51)
4	0.14368062910(29)	1.50000038(61)	0.14368062914(16)	1.50000038(44)
5	0.14368062890(43)	1.50000085(93)	0.14368062911(53)	1.5000002(12)
6	0.14368062863(49)	1.5000014(10)	0.14368062901(54)	1.5000005(12)
7	0.14368062886(39)	1.50000094(80)	0.14368062881(52)	1.5000009(10)
8	0.14368062885(64)	1.5000008(13)	0.143680629210(97)	1.50000021(24)

Table 3: A fit to the asymptotic form $p_2 n x_c^{2n} \sim n^{-\frac{5}{2}}[a_1 + a_2/n + a_3/n^2 + a_4/n^3 + \dots]$
 Estimates of the amplitudes a_1, a_2, a_3, a_4 .

n	a_1	a_2	a_3	a_4
20	0.09940085	-0.02745705	0.02476376	0.11822181
21	0.09940118	-0.02747548	0.02511347	0.11601107
22	0.09940140	-0.02748880	0.02537979	0.11423855
23	0.09940154	-0.02749767	0.02556592	0.11293766
24	0.09940164	-0.02750397	0.02570426	0.11192457
25	0.09940170	-0.02750829	0.02580364	0.11116357
26	0.09940174	-0.02751137	0.02587757	0.11057283
27	0.09940177	-0.02751355	0.02593211	0.11011880
28	0.09940179	-0.02751510	0.02597236	0.10977030
29	0.09940180	-0.02751619	0.02600168	0.10950667
30	0.09940181	-0.02751694	0.02602273	0.10931043
31	0.09940182	-0.02751745	0.02603734	0.10916929
32	0.09940182	-0.02751777	0.02604692	0.10907354
33	0.09940182	-0.02751795	0.02605254	0.10901552
34	0.09940182	-0.02751802	0.02605500	0.10898929
35	0.09940182	-0.02751802	0.02605494	0.10898993
36	0.09940182	-0.02751796	0.02605285	0.10901358
37	0.09940182	-0.02751785	0.02604913	0.10905699
38	0.09940182	-0.02751771	0.02604408	0.10911757
39	0.09940182	-0.02751755	0.02603796	0.10919302
40	0.09940182	-0.02751736	0.02603097	0.10928158
41	0.09940182	-0.02751717	0.02602327	0.10938160
42	0.09940181	-0.02751696	0.02601500	0.10949174
43	0.09940181	-0.02751675	0.02600629	0.10961079
44	0.09940181	-0.02751653	0.02599720	0.10973796
45	0.09940181	-0.02751631	0.02598785	0.10987195