A transfer matrix approach to the enumeration of plane meanders

A transfer matrix approach to the enumeration of plane meanders

Iwan Jensen†

Department of Mathematics and Statistics, University of Melbourne, Victoria 3010, Australia

E-mail: `I.Jensen@ms.unimelb.edu.au`

Abstract. A closed-plane meander of order n is a closed self-avoiding curve intersecting an infinite line $2n$ times. Meanders are considered distinct up to any smooth deformation leaving the line fixed. We have developed an improved algorithm, based on transfer matrix methods, for the enumeration of plane meanders. While the algorithm has exponential complexity, its rate of growth is much smaller than that of previous algorithms. The algorithm is easily modified to enumerate various systems of closed meanders, semi-meanders, open meanders and many other geometries.

1. Introduction

Meanders [1] form a set of combinatorial problems concerned with the enumeration of self-avoiding curves crossing a line through a given number of points. Meanders are considered distinct up to any smooth deformation. This problem seems to date back, at least, to work of Poincaré on differential geometry [2]. Since then it has, from time to time, been studied by mathematicians in various contexts such as the folding of a strip of stamps [3, 4] or the folding of maps [5]. More recently it has been related to enumerations of ovals in planar algebraic curves [6] and the classification of 3-manifolds [7]. During the last decade or so has it has received considerable attention in other areas of science. In computer science meanders are related to the sorting of Jordan sequences [8] and have been used for lower bound arguments [9]. In physics meanders are relevant to the study of compact foldings of polymers [10, 11], properties of the Temperley–Lieb algebra [12, 13], matrix models [14–16], and defects in liquid crystals and $(2 + 1)$-dimensional gravity [17].

The difficulty in the enumeration of the most interesting combinatorial problems is that, computationally, they are of exponential complexity. That is to say, the time it takes to calculate the first n terms in the generating function grows asymptotically as λ^n, where $\lambda > 1$ is the growth rate. Initial efforts at computer enumeration of meanders were based on direct counting. Independently, Koehler [4] and Lunnon [5], studied the number of ways of folding a strip of stamps (or a map) of length n, and published results up to $n = 16$ and 24, respectively. Lando and Zvonkin [1] studied closed meanders, open meanders and multi-component systems of closed meanders, and calculated the number of open meanders up to $n = 26$ and the number of closed meanders up to $n = 14$. The calculation of the number of closed meanders was

---

† The series for the various generating functions so far generated with this algorithm and studied in [21] can be obtained via e-mail by sending a request to I.Jensen@ms.unimelb.edu.au or via the world wide web on the URL http://www.ms.unimelb.edu.au/˜iwan/ by following the instructions.

subsequently extended up to $n = 16$ by Pratt [18]. Di Francesco *et al* studied semi-meanders, a problem equivalent to the stamp folding problem [10], and extended the calculation to $n = 29$ [11], in addition they studied numerous other problems including that of multi-component systems of semi-meanders.

Few exact and mathematically rigorous results have been obtained for any of the many meander problems. However, in a recent paper it was conjectured that some of the meander problems can be related to a gravitational version of a certain loop model [19]. From the conformal field theory of the model, conjectures were proposed for the exact critical exponent of closed and open meanders, $\alpha = (29 + \sqrt{145})/12 = 3.420\,1328\ldots$, as well as the exponent for semi-meanders, $\overline{\alpha} = 1 + \sqrt{11}(\sqrt{29} + \sqrt{5})/24 = 2.053\,1987\ldots$. This work has recently been extended to multi-component systems of closed and semi-meanders [20] and to various other geometries. Conjectures were given for the critical exponents as functions of the loop-fugacity q. These were checked numerically [20] and found to be correct within numerical error. In a recent paper [21] we analysed extended series for the meander generating functions. Using the numerical technique of differential approximants [22] we obtained accurate estimates for the exponents and found that the conjecture for the semi-meander exponent is unlikely to be correct, while the conjecture for closed meanders is just consistent with the results from the analysis.

The purpose of this paper is to give a detailed description of the new and improved algorithms used to derive the series studied in [21]. While the algorithm still has exponential complexity, the growth rate is much smaller than that experienced with direct counting, and consequently the calculation can be carried much further. The algorithm is easily modified to enumerate various multi-component systems of closed meanders, semi-meanders or open meanders. In particular, we have extended the calculation for closed meanders up to $n = 24$, for open meanders up to $n = 43$ and for semi-meanders up to $n = 45$.

In section 2 we shall briefly describe some meander problems and define the meandric numbers. Section 3 contains a detailed description of the transfer matrix algorithm for the enumeration of closed meanders and an outline of generalizations to other problems. Finally, we give our conclusions in section 4.

2. Definitions of meanders

A *closed meander* of order n is a closed self-avoiding curve crossing an infinite line $2n$ times (see figure 1). The meandric number M_n is simply the number of such meanders distinct up to smooth transformations. Note that each meander forms a single connected curve. The number of closed meanders is expected to grow exponentially, with a sub-dominant term given by a critical exponent, $M_n \sim C R^{2n}/n^\alpha$. The exponential growth constant R is often called the *connective constant*. The generating function is expected to behave as

$$\mathcal{M}(x) = \sum_{n=1}^{\infty} M_n x^n \sim A(x)(1 - R^2 x)^{\alpha - 1} \tag{1}$$

and hence have a singularity at $x_c = 1/R^2$ with exponent $\alpha - 1$.

We can extend the definition to *multi-component systems of closed meanders*, where we allow configurations with several disconnected closed meanders. The meandric numbers $M_n^{(k)}$ are the number of meanders with $2n$ crossings and k components, and we thus obtain the more general generating function:

$$\mathcal{M}(x, q) = \sum_{n=1}^{\infty} \sum_{k=1}^{n} M_n^{(k)} x^n q^k. \tag{2}$$
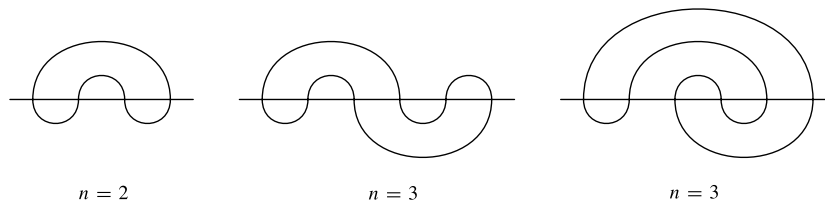
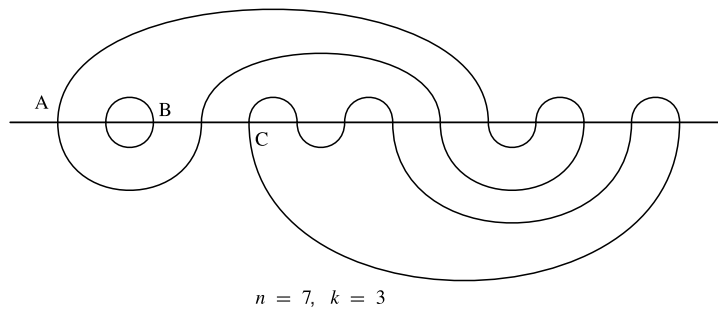**Figure 1.** Some examples of closed meanders of order 2 and 3, respectively.

$$n = 7, \quad k = 3$$

Figure 2. An example of a system of closed meanders of order 7 with 3 components.
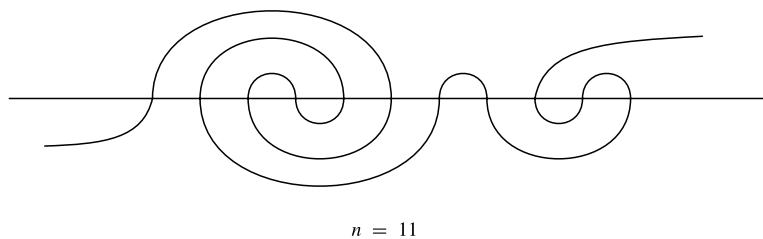

$$n = 11$$

Figure 3. An example of an open meander of order 11.

Obviously, $M_n = M_n^{(1)}$, and $\mathcal{M}(x) = \partial \mathcal{M}(x, q)/\partial q|_{q=0}$. An example of a multi-component meander system with several pieces is shown in figure 2. In addition one could also look at further restrictions on these systems. Two obvious examples would be to exclude configurations where meanders are nested within one another or where meanders can be separated from one another, i.e. all smaller meanders are completely enclosed within a larger meander.

An *open meander* of order n is a self-avoiding curve running from west to east while crossing an infinite line n times (see figure 3). The number of such curves is m_n and we can define a generating function for this problem in analogy with (1). It should be noted [1] that $M_n = m_{2n-1}$, and hence the critical exponent is identical to that of closed meanders and the connective constant is R.

Finally, instead of looking at crossings of an infinite line one could consider a semi-infinite line and allow the curve to wind around the end-point of the line [10]. A *semi-meander* of order n is a closed self-avoiding curve crossing the semi-infinite line n times. The number of semi-meanders of order n is denoted by $\overline{M}_n \sim C'\overline{R}/n^{\overline{\alpha}}$ and we define a generating function as in (1). In this case a further interesting generalization is to study the number of semi-meanders $\overline{M}_n(w)$ which wind around the end-point of the line exactly w times. Again we could also

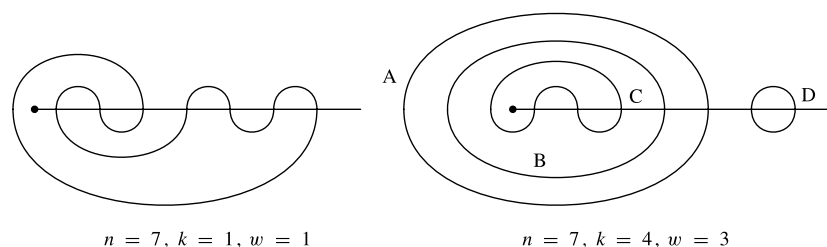$n = 7, k = 1, w = 1$ $n = 7, k = 4, w = 3$

Figure 4. Two examples of semi-meanders. The first of these has one component, crosses the line 7 times and winds around the origin once, while the second has 4 components crosses the line 7 times and winds around the origin 3 times.

study systems of multi-component semi-meanders according to the number of independent meanders. Two semi-meanders are shown in figure 4.

3. Enumeration of meanders

The method used to enumerate meanders is based on the transfer matrix algorithm devised by Enting [23] in his pioneering work on the enumeration of self-avoiding polygons. Derrida [24] used a similar algorithm to study self-avoiding walks. The transfer matrix technique involves drawing a boundary line perpendicular to the infinite line. The intersection between the boundary and a given meander results in a set of loop-ends. Each loop-end is connected (to the left of the boundary) to at most one other loop-end. In the case of closed meanders the matching is perfect and each loop-end is connected to exactly one other loop-end. In the case of open meanders there is in addition one and only one loop-end which is free and therefore not connected to any other loop-ends, as illustrated in figure 5. In addition to the information describing the configuration of loop-ends, and how they are connected, we need to know where the infinite line is situated within the loop-ends. This can be done simply by specifying how many loop-ends lie beneath the infinite line. For each such configuration we keep count of all the possible (partially completed) meanders which gives rise to that particular configuration
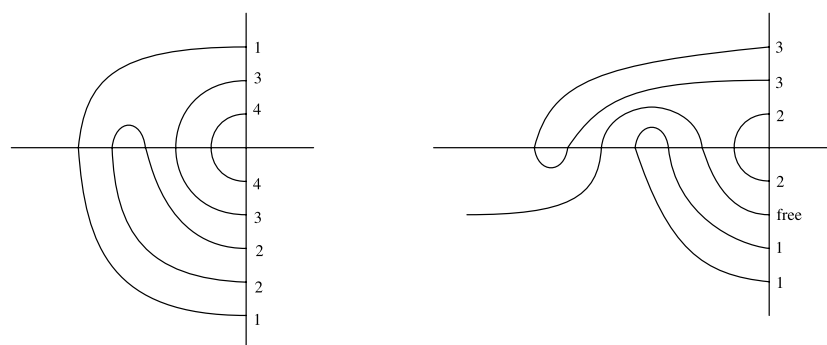

Figure 5. Two examples of loop-configurations along the boundary (vertical line) during the transfer matrix calculation for closed meanders (left panel) and open meanders (right panel). Numbers along the boundary give a possible encoding of the loop-ends in the partially completed meander.
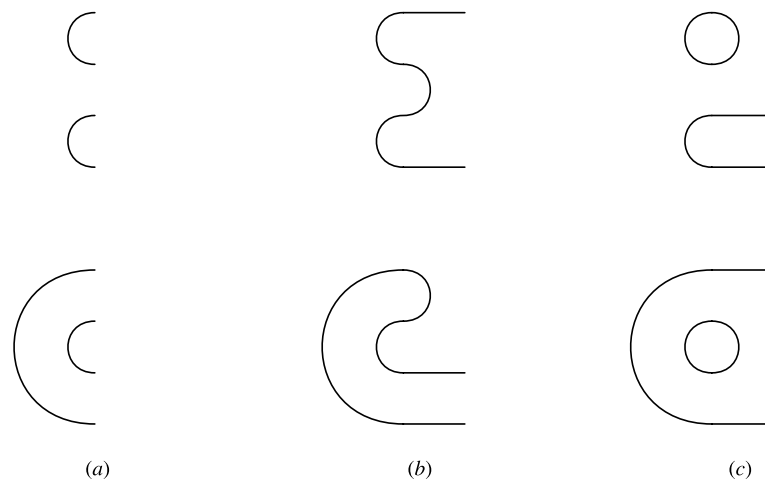
Figure 6. Illustration of (*a*) how two partial loops can be placed, (*b*) how they can be connected to from a single loop, and (*c*) connections leading to graphs with more than one component.

of loop-ends. Meanders can then be enumerated by successive moves of the boundary line, so that exactly one crossing is added with each move. An extra crossing is added *either* by putting in a new loop across the infinite line *or* by taking an existing loop-end immediately above/below the line and dragging it to the other side.

These general remarks hold for all the meander enumeration problems. In the following we give a detailed description of the algorithm used in the enumeration of closed connected meanders. Afterwards we describe how to generalize the method to other meander problems.

3.1. Enumeration of closed connected meanders

In the enumeration of closed connected meanders the two major constraints which must be observed are self-avoidance and the constraint that all meanders constructed during the calculation must consist of a single connected component. As we move the boundary line, the partially completed meanders will consist of a number of disjoint loop segments, which must be connected to each other if a valid closed meander is to be produced. A pair of loops can be placed relative to one another in two distinct ways, namely, side by side or nested, as shown in figure 6(*a*). In each case it is possible to connect the loop-ends so as to form a single loop (*b*) or so as to form graphs with two separate components (*c*). It is connections equivalent to these latter cases which we must avoid. So the constraint, which must be observed in order to avoid separate components, is that a loop can be closed on itself only if the boundary intersects no other loops.

To avoid loops closing on themselves we have to label the loop-ends so we can keep track of how they are connected to one another. One scheme would be to assign a unique label to each loop as indicated in figure 5. However, due to the two-dimensional nature of the problem and since the curve making up a meander is self-avoiding, there is a scheme better suited to a computer algorithm. Each configuration of loop-ends can be represented by an ordered set of states $\{\sigma_i\}$, where

$$\sigma_i = \begin{cases} 0 & \text{lower end of a loop} \\ 1 & \text{upper end of a loop.} \end{cases} \tag{3}$$

It is easy to see that this encoding uniquely describes which loop-ends are connected. In order to find the upper loop-end, matching a given lower end, we start at the lower end and work upwards in the configuration counting the number of '0's and '1's we pass (the '0' of the initial lower end is *not* included in the count). We stop when the number of '1's exceeds the number of '0's. This '1' marks the matching upper end of the loop. It is worth noting that there are some restrictions on the possible configurations. First, every lower loop-end must have a corresponding upper end, and it is therefore clear that the total number of '0's is equal to the total number of '1's. Secondly, as we look through the configuration starting from the bottom the number of '0's is never smaller than the number of '1's. Those familiar with algebraic languages will immediately recognize that each configuration of labelled loop-ends forms a Dyck word (see [25]).

3.1.1. The transfer matrix algorithm.   The total configuration of loop-ends and their placement relative to the infinite line can thus be described by a pair of integers (h, S), where h is the number of loop-ends below the infinite line and S is the integer whose binary representation corresponds to the configuration of loop-ends. We shall call such an (h, S)-pair a *signature*, and in practice we represent it by a 64-bit integer with the first 6 bits coding h and the remaining bits coding S. In the following we shall often explicitly write out the binary representation, $\{b_0 b_1 \ldots b_n\}$ of S, and use the notation $\{S_1 S_2\}$ to mean a configuration of loop-ends obtained by concatenating the binary strings S_1 and S_2.

The algorithm for enumerating closed connected meanders is:

(a) Set the maximum order n of the meandric numbers we wish to calculate. Start with the signature $(1, \{01\})$ with a count of 1, that is one loop crossing the infinite line. Set the number of crossings $n_c = 1$. Set the meandric number $M_1 = 1$.

(b) Next move the boundary one step ahead and add one more crossing, $n_c = n_c + 1$, to all existing signatures. A new crossing is added either by putting in an additional loop or by dragging an existing loop-end (immediately above or below) across the infinite line.

- *Adding.* A new loop is inserted simply by taking an existing *source* signature $(h, \{S_1 S_2\})$, where S_1 (S_2) is the configuration of loop-ends below (above) the infinite line, and creating a new *target* signature $(h + 1, \{S_1 01 S_2\})$. The count of the source is added to the count of the target. This first type of move is illustrated in figure 7 where in moving to position 2 we generate the target signature $(2, \{0011\})$ from the source $(1, \{01\})$. Additional loops are also inserted while moving to positions 4 and 7.

- *Dragging.* As we cross the infinite line with an existing loop-end we *either* leave it at that *or* we may be allowed to connect the loop-end to the loop-end on the other side of the infinite line.

  * *No connection.* From the source (h, S) we produce the two targets $(h - 1, S)$ and $(h + 1, S)$, by dragging a loop-end from below to above and vice versa. Both these updates are allowed provided $h - 1 \geqslant 0$ and $h + 1 \leqslant 2m_1$, where m_1 is the number of '1's in S. For $h = 0$ only the target $(1, S)$ is allowed and for $h = 2m_1$ only the target $(h - 1, S)$ is allowed. Examples of these moves are given in figure 7 when moving to positions 3, 5, 8 and 11.

  * *Connecting loop-ends.* There are four distinct cases depending on whether the loop-ends below and above are of type '0' or '1'.

    + *Case 00.* In this case we connect a lower loop-end from below the line to a lower loop-end above the line. From a source $(h, \{S_1 00 S_2\})$ we generate the target $(h-1, \{S_1 \widehat{S_2}\})$, where the symbol $\widehat{S_2}$ indicates that this string is changed
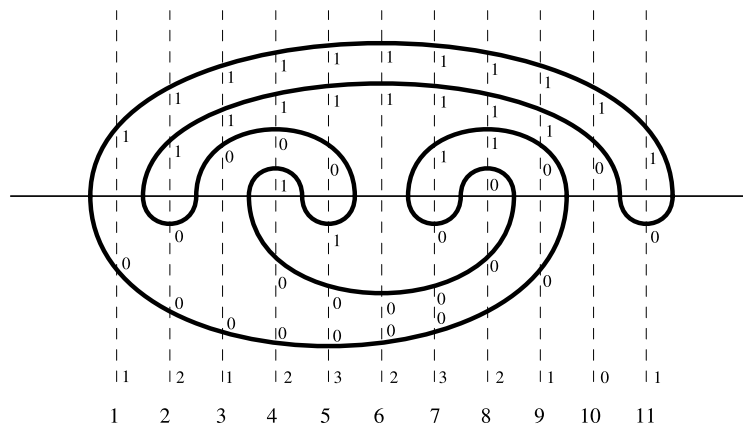
Figure 7. Positions of the boundaries (broken lines) during the transfer matrix calculation. Numbers along the boundaries give the encoding of the loop structure of the intersection with the partially completed meander to the left of the boundaries.

via further processing. This is so because by connecting the two lower loop-ends an upper loop-end elsewhere in the old configuration S_2 becomes a lower loop-end in the new configuration $\widehat{S_2}$. An example of this type of relabelling is shown in figure 7 where in the move to position 9 we see that the signature $(2, \{000111\})$ before the step becomes the configuration $(1, \{0011\})$ after the step. That is, the upper end of the third loop before the step becomes the lower end of the second loop after the step. In general, the nesting of loops could be more complicated and the general rule for the relabelling of the configuration is as follows. When connecting two '0's we work upwards in the configuration, counting the number of '0's and '1's we pass until the number of '1's exceeds the number of '0's. This '1' is the matching end of the inner loop and it should now be changed to a '0', thus becoming the lower end of the outer loop (drawing a few further figures should make this relabelling clearer).

+ *Case 10.* In this case we connect an upper loop-end from below to a lower loop-end above. So from a source $(h, \{S_1 10 S_2\})$ we generate the target $(h - 1, \{S_1 S_2\})$.

+ *Case 01.* This is never allowed since it would result in a closed loop and thus generate graphs with separate components. The only exception is when there are no other loop-ends in S, but this case is dealt with in (c).

+ *Case 11.* In this case we connect an upper loop-end from below the line to an upper loop-end above the line. From a source $(h, \{S_1 11 S_2\})$ we generate the target $(h - 1, \{\widehat{S_1} S_2\})$. The rule for the relabelling of $S_1 \; \widehat{S_1}$ is similar to the case '00', but we work downwards in the string S_1 until we find the unmatched lower loop-end, which is then changed to an upper loop-end.

Note that all of the above moves may be allowed. So from a given source we can generate up to four targets, by adding a new loop, by dragging a loop-end from below to above the infinite line, doing the reverse, or by connecting two loop-ends across the infinite line. As we move along and generate new target signatures their counts are calculated by adding up the count for the various source signatures which could generate that target.

For example the target $(2, \{0011\})$ is generated from the sources $(1, \{01\})$, $(1, \{0011\})$, $(3, \{0011\})$, and $(3, \{001011\})$, by, respectively, putting in an additional loop, moving a loop-end below the line, moving a loop-end above the line and connecting two loop-ends across the line.

(c) If n_c is odd then set $j = (n_c + 1)/2$ and extract the meandric number M_j as the count of the signature $(1, \{01\})$. This is the only case in which we are allowed to close a loop. Doing so obviously adds one more crossing.

(d) If $n_c < 2N - 1$ go to (b).

Not all the possible signatures that can be generated in a calculation to order n are actually required. The main restriction is that no meanders should have more than $2n$ crossings. Since each move adds one more crossing and reduces the number of loop-ends above/below the infinite line by at most one, it is clear that for a given signature we have to add at least $n_a = \max(h, 2m_1 - h)$ additional crossings in order to produce a closed meander. Thus if, for a given signature, $n_c + n_a > 2n$, we can discard the signature since it would contribute only to a meandric number exceeding the order to which we wish to carry out the calculation. A further saving of a factor of almost 2 is obtained by using the symmetry with respect to reflection in the infinite line. A further factor of approximately 2 is obtained as follows. Note that n_a is the minimum number of additional crossings and that for some signatures further crossings are needed. The most obvious case is when the loop-ends above and below the infinite line are connected to one another. In this case we cannot connect the two ends and first we have to move one of the loop-ends across the line. So when $h = m_1$, this results in at least two extra crossings. In the general case one can readily write an algorithm to count the actual number of additional crossings required, and as stated above, this results in a saving of close to 2 in the number of signatures one need retain.

3.2. Generalizations to other meander problems

3.2.1. Multi-component systems of closed meanders. As we noted above connecting a '0' below the line to a '1' above the line results in a closed loop. Failure to observe the restriction on this closure would result in graphs with disconnected components, either one closed meander over another or one closed meander within another. Obviously, these are just the types of graphs required in order to enumerate multi-component systems of closed meanders. So by noting that each such closure adds one more component it is straightforward to generalize the algorithm to enumerate systems of closed meanders. The only major change is that, rather than just storing the number of partially completed meanders, for each signature we have to store a generating function, that is a polynomial giving the number of partially completed meanders with k components, where $1 \leqslant k \leqslant n$.

3.2.2. Open meanders. Open meanders are a little more complicated. The first part of the necessary generalization consists in adding an extra piece of information to our signature. We have to keep track of a single free end by specifying its position within the configuration of connected loop-ends. One simple way of doing this is, in analogy with the infinite line, to specify the number, $h_f - 1$, of (connected) loop-ends below the free end, so that h_f is the position of the free end as counted from the bottom. So a configuration is now described by a signature (h, h_f, S). Naturally, we also have to generalize the algorithm described above. We now start with the signature $(1, 1, 0)$, that is a single free end below the infinite line, and no crossings. The updating rules for adding a new crossing are very similar to those described above for closed meanders. One difference is that when a new loop is added (two loop-ends
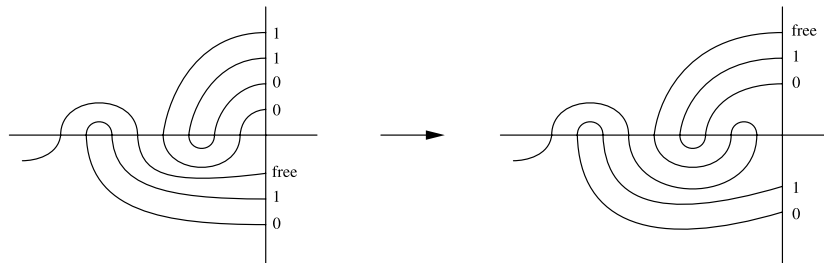
Figure 8. An example of the transformation of a signature as the free end is joined to a loop-end. In this case the source signature (3, 3, {010011}) gives rise to the target (2, 5, {0101}).

joined) below the free end, h_f is increased (decreased) by 2. We also need to consider what happens when joining the free end to a connected loop-end. In this case we have to change the matching end of the connected loop to the new free end in the target signature and we have to change h_f accordingly. An example illustrating this is shown in figure 8. The updating rule when the free end does not join with the loop-end on the other side is obviously just to increase (decrease) h by 1 as the free end is moved below (above) the line.

3.2.3. Semi-meanders.

Finally, in order to enumerate semi-meanders all we need do is change the starting configuration. We now start in a position just before the first crossing of the semi-infinite line with w loops nested with one another. By running the algorithm for each w from 0 to n we can count all semi-meanders with up to n crossings. The generalization to multi-component systems of semi-meanders is the same as for closed meanders.

3.3. Computational complexity

Using the new algorithm we have calculated M_n up to $n = 24$ as compared with the previous best of $n = 16$ obtained by Pratt [18]. To fully appreciate the advance it should be noted that the computational complexity grows exponentially, that is the time required to obtain the n term grows asymptotically as λ^n. For direct enumerations time is simply proportional to M_n and thus $\lambda = \lim_{n\to\infty} M_{n+1}/M_n \approx 12.26$. Thus extending the count of the meandric number from 16 to 24 by direct counting would have required approximately $12.28^8 \simeq 5 \times 10^8$ as much CPU time as the calculation of the first 16 terms. The transfer matrix method employed in this paper is far more efficient. In figure 9 we have plotted the maximum number of signatures required in order to calculate the number of closed meanders up to order n. As can be seen, the number of signatures grows exponentially with n, and the numerical evidence suggests that the computational complexity is such that $\lambda \approx 2.5$, which obviously is a very significant improvement on direct counting. The drawback of the transfer matrix method is that, since we need to store all the different signatures, the memory requirement of the algorithm also grows exponentially with growth rate λ, whereas direct counting algorithms typically have memory requirements which are linear in n. In fact, it is exactly the memory requirement which is the major limitation of the transfer matrix method. The calculations reported in this paper used up to 2 Gb of memory and typically took a few days of CPU time.

Di Francesco *et al* [20] carried out a more detailed analysis of the complexity of the algorithm as it applies to multi-component systems of closed meanders and obtained the estimate $N_{sig} \sim a^n = (2.582\,603\ldots)^n$, for the number of signatures, very close to the purely empirical estimate obtained here.
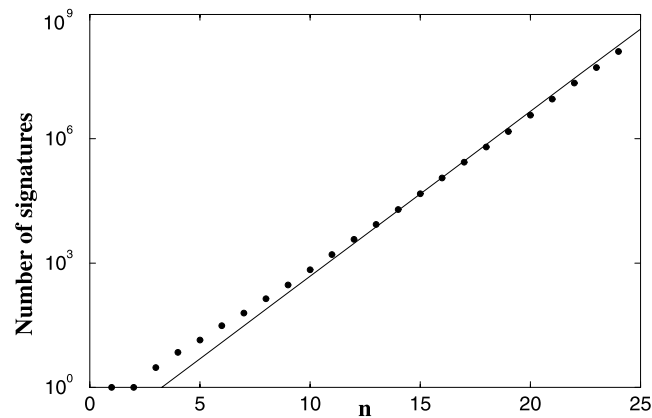
Figure 9. The number of distinct signatures required during the transfer matrix calculation of the number of closed meanders with $2n$ crossings. The full line, drawn as a guide to the eye, corresponds to a growth rate $\lambda = 2.5$.

Another way of gauging the improved efficiency is to note that the calculations for semi-meanders carried out in [11] were 'done on the parallel Cray-T3D (128 processors) of the CEA-Grenoble, with approximately 7500 h × processors'. Or in total about 100 years of CPU time. The equivalent calculations can be done with the transfer matrix algorithm in about 15 minutes on a single processor workstation!

3.4. Further particulars

Finally, a few remarks of a more technical nature. The number of contributing configurations becomes very sparse in the total set of possible states along the boundary line and as is standard in such cases one uses a hash-addressing scheme [26]. Since the integer coefficients occurring in the expansion become very large, the calculation was performed using modular arithmetic [27]. This involves performing the calculation modulo various prime numbers p_i and then reconstructing the full integer coefficients at the end. In calculations involving only single component meanders most of the memory is used storing the signatures so we used primes of the form $p_i = 2^{30} - r_i$. The Chinese remainder theorem ensures that any integer has a unique representation in terms of residues. If the largest integer occurring in the final expansion is M, then we have to use a number of primes m such that $p_1 p_2 \cdots p_m > M$. Up to 3 primes were needed to represent the coefficients correctly. In calculations involving systems of multi-component meanders the main storage requirement comes from the generating functions. In order to save memory we used primes of the form $p_i = 2^{15} - r_i$ so that the residues of the coefficients in the polynomials could be stored using 16 bit integers. In this case we used up to 6 primes.

4. Conclusion

We have presented an improved algorithm for the enumeration of plane meanders. The computational complexity of the algorithm for the problem of closed meanders is estimated to be 2.5^n, much better than direct counting algorithms which have complexity 12.26^n. Implementing this algorithm enabled us to obtain closed meanders up to order $n = 24$,

compared with the previous best of $n = 16$. We also counted the number of open meanders up to order $n = 43$ and semi-meanders up to order $n = 45$. From our extended series we obtained precise estimates for the connective constants and critical exponents [21]. This showed that a recent conjecture for the exact value of the semi-meander critical exponent is unlikely to be correct, while the conjectured exponent value for closed and open meanders is just consistent with the results from the analysis.

Acknowledgment

References

[1] Lando S K and Zvonkin A K 1993 *Theor. Comput. Sci.* **117** 227
[2] Poincaré H 1912 *Rend. Circ. Mat. Palermo* **33** 375
[3] Touchard J 1950 *Can. J. Math.* **2** 385
[4] Koehler J E 1968 *J. Combust. Theory* **5** 135
[5] Lunnon W 1968 *Math. Comput.* **22** 193
[6] Arnold V 1988 *Siberian Math. J.* **29** 717
[7] Ko K H and Smolinsky L 1991 *Pac. J. Math.* **149** 319
[8] Hoffmann K, Mehlhorn K, Rosenstiehl P and Tarjan R E 1988 *Inf. Control* **68** 170
[9] Alon N and Maass W 1988 *J. Comput. Syst. Sci.* **37** 118
[10] Di Francesco P, Golinelli O and Guitter E 1997 *Math. Comput. Modelling* **26** 97
[11] Di Francesco P, Golinelli O and Guitter E 1996 *Nucl. Phys.* B **482** 497
[12] Di Francesco P, Golinelli O and Guitter E 1997 *Commun. Math. Phys.* **186** 1
[13] Di Francesco P 1998 *Commun. Math. Phys.* **191** 543
     Di Francesco P 1997 *J. Math. Phys.* **38** 5905
[14] Makeenko Y 1996 *Nucl. Phys. Proc. Suppl.* **49** 226
[15] Semenoff G W and Szabo R J 1997 *Int. J. Mod. Phys.* A **12** 2135
[16] Di Francesco P 1999 *Preprint* math-ph/9911002 http://arxiv.org/abs/
[17] Kholodenko A L 1999 *J. Geom. Diff.* **33** 23
     Kholodenko A L 1999 *Preprint* hep-th/9901040 http://arxiv.org/abs/
[18] Pratt V R *Sloane's On-Line Encyclopedia of Integer Sequences* ed N Sloane
          http://www.research.att.com/~njas/sequences/
[19] Di Francesco P, Golinelli O and Guitter E 2000 *Nucl. Phys.* B **570** 699
[20] Di Francesco P, Guitter E and Jacobsen J L 2000 *Nucl. Phys.* B **580** 757
     Di Francesco P, Guitter E and Jacobsen J L 2000 *Preprint* cond-mat/0003008 http://arxiv.org/abs/
[21] Jensen I and Guttmann A J 2000 *J. Phys. A: Math. Gen.* **33** L187
[22] Guttmann A J 1989 *Phase Transitions and Critical Phenomena* vol 13 ed C Domb and J L Lebowitz (New York:
          Academic) pp 1–234
[23] Enting I G 1980 *J. Phys. A: Math. Gen.* **13** 3713
[24] Derrida B 1981 *J. Phys. A: Math. Gen.* **14** L5
[25] Delest M-P and Viennot G 1984 *Theor. Comput. Sci.* **34** 169
[26] Mehlhorn K 1984 *Data Structures and Algorithms I: Sorting and Searching (EATCS Monographs on Theoretical
          Computer Science)* (Berlin: Springer)
[27] Knuth D E 1969 *Seminumerical Algorithms (The Art of Computer Programming 2)* (Reading, MA: Addison-
          Wesley)