Journal of Statistical Physics, Vol. 60, Nos. 1/2, 1990

A Numerical Method to Compute Exactly the Partition Function with Application to $Z(n)$ Theories in Two Dimensions

Gyan Bhanot[1,2]

Received November 15, 1989; revision received January 24, 1990

I present a new method to exactly compute the partition function of a class of discrete models in arbitrary dimensions. The time for the computation for an n-state model on an L^d lattice scales like $n^{L^{d-1}}nL^d$. I show examples of the use of this method by computing the partition function of the 2D Ising and 3-state Potts models for maximum lattice sizes 10×10 and 8×8, respectively. The critical exponents ν and α and the critical temperature one obtains from these are very near the exactly known values. The distribution of zeros of the partition function of the Potts model leads to the conjecture that the ratio of the amplitudes of the specific heat below and above the critical temperature is unity.

KEY WORDS: Potts and Ising models; exact partition function; zeros; exponents; scaling.

In this paper, I will describe a numerical method to compute *exactly* the partition function of a certain class of discrete models and give examples of its use in two dimensions. The inspiration for this paper came from an old work by Binder.[1]

I will illustrate the method by using the simplest possible example, the two-dimensional Ising model on a 2×2 lattice. For the moment, consider open boundary conditions. One starts by enumerating all states of the two spins in the bottom row. For the 2×2 lattice, there are four such states and they are shown in Fig. 1. It is convenient to label the states by the binary digits 00, 01, 10, 11 corresponding to the values of the spins in an obvious notation that is very useful in the Ising case. I will choose the energy func-

[1] Theory Division, CERN, CH-1211 Geneva 23, Switzerland, and Supercomputer Computations Research Institute, Florida State University, Tallahassee, Florida 32306.
[2] Current address: Thinking Machines Corporation, Cambridge, Massachusetts 02142-1214.

0022-4715/90/0700-0055$06.00/0

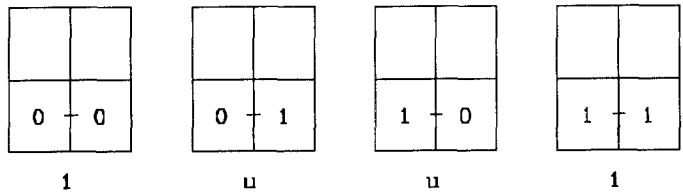

Fig. 1. Configurations of two spins in the bottom row of a 2×2 Ising model. The Boltzmann weights corresponding to each configuration are written below the configuration. Blank squares indicate unfilled lattice sites, and bars connecting squares indicate bonds for which the Boltzmann weight is accounted for.

tion so that the energy of a pair 00 is 0 and that of a pair 01 is 1. The states then have Boltzmann weight 1, $u = e^{-\beta}$, u, and 1, respectively, as shown in Fig. 1. Define two arrays $Z^o(S)$ and $Z^n(S)$ with four storage locations each (i.e., as many storage locations as states of the lowest row of spins). Initialize Z^o as follows:

$$Z^o(00) = 1, \qquad Z^o(01) = u, \qquad Z^o(10) = u, \qquad Z^o(11) = 1 \tag{1}$$

Here the argument of Z^n and Z^o refers to the spins that are uppermost in each column. Now imagine adding a spin on the top left-hand corner site with value 0. The configurations will be multiplied by the Boltzmann weights of the vertical bonds generated by the addition of the new spin. These are shown in Fig. 2. The information about these may be stored in Z^n by performing the following operation:

$$Z^n(00) = Z^o(00) + uZ^o(10) \tag{2a}$$

$$Z^n(01) = Z^o(01) + uZ^o(11) \tag{2b}$$

This clearly accounts for the four states in Fig. 2. For the case when the added spin has value 1 one has to perform two more operations:

$$Z^n(10) = Z^o(10) + uZ^o(00) \tag{3a}$$

$$Z^n(11) = Z^o(11) + uZ^o(01) \tag{3b}$$

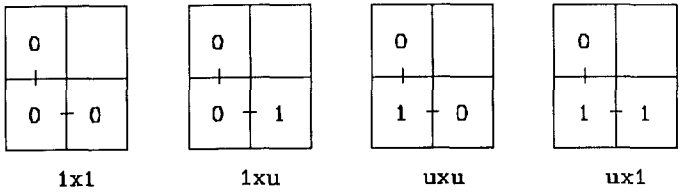

Fig. 2. The same as Fig. 1, but now with one more site filled up.

If one looks at Eqs. (2) and (3) and one imagines what happens (as far as the vertical bonds are concerned) when another spin is added in the empty upper right-hand corner, one is led to the following general algorithm for an $L \times L$ system:

For a spin added at site location i, perform the set of operations

$$Z^n(S) = Z^o(S) + uZ^o(S') \tag{4}$$

on all the states S. In Eq. (4), S is a string of bits,

$$S = b_1 b_2 \cdots b_i \cdots b_L \tag{5a}$$

and

$$S' = b_1 b_2 \cdots \bar{b}_i \cdots b_L \tag{5b}$$

where $\bar{b}_i$ changes 0 to 1 and vice versa. By repeating this algorithm for each spin in a row, all the Boltzmann weights for the vertical bonds generated are accounted for. After the row is completed, one can then account for the horizontal bonds. This is because, given a row of spins in a state $S = b_1 b_2 \cdots b_L$, one can immediately compute the Boltzmann weight of the bonds connecting these spins. It is given by u^k, $k = \sum_{i=1}^{L-1} b_i \oplus b_{i+1}$ (for open boundary conditions), where $\oplus$ denotes exclusive OR. Thus, the Boltzmann weight for the horizontal bonds for $Z(S)$ is determined from the binary representation of S. The lattice is thus built up row by row starting from the first row. After all the rows are added, the partition function is calculated as

$$Z(u) = \sum_S Z(S) \tag{6}$$

The procedure described above obviously generalizes to d dimensions, where one builds up the lattice starting from a $(d-1)$-dimensional partition function. The procedure so far may also be generalized to any model, discrete or continuous.

Let us see now why Binder's algorithm is potentially more efficient than the straightforward algorithm of enumerating all the states. The reason is that if there is enough storage at one's disposal, the number of arithmetic operations performed in the method described is much smaller than enumerating all states. To see this, suppose one had a theory where each spin takes on n values. In d dimensions, one would then start by enumerating all the states of the $(d-1)$-dimensional theory. Thus, Z^o and Z^n each need $n^{L^{d-1}}$ storage locations. Therefore, the storage requirement scales as

$$\text{Storage} \sim 2n^{L^{d-1}} \tag{7}$$

Now let us count how many operations are necessary to generate the partition function. For each added spin, one has to update all the $Z(S)$ values and the algorithm analogous to Eq. (4) has n terms in it. Thus, to add L' layers of spins takes

$$\text{Operations} \sim n^{L^{d-1}} n L^{d-1} L' \tag{8}$$

steps. On the other hand, the number of states generated is $n^{L'L^{d-1}}$ and this is much greater than Eq. (8) for large L, L'.

Up to this point, the method is just what was in Binder's paper.[1] The problem with the method as described so far is that after a lot of computations, one ends up with the numerical value for Z at some temperature $1/\beta$, and to calculate Z for any other temperature, one has to repeat the procedure again. I will show that for a class of models where the energy of a spin pair takes on integer values only, the method can be extended to find the partition function for any temperature from a single sequence of the operations outlined above. For the rest of this paper, I will therefore restrict myself to such systems where the energy of each spin pair is an integer.

As noted above, the procedure described so far generates the *value* of the partition function at some value of u. To find Z at an arbitrary temperature, one would like instead to find the spectral density function $N(k)$ defined by

$$Z(u) = \sum_{k=0}^{k_{\max}} N(k)\, u^k \tag{9}$$

where $N(k)$ is the number of states of the system at energy k and $k_{\max}$ is the maximum possible energy (which is obviously finite, since we are on a finite lattice). One can imagine three alternative ways of computing the $N(k)$.

1. Since $Z(u)$ is a polynomial of degree $k_{\max}$, one way to find $\{N(k)\}$ is to calculate $Z(u)$ for $k_{\max}+1$ distinct values of u and then solve the linear system (9). Unfortunately, this requires that $Z(u)$ be calculated to incredible precision, as the coefficient matrix of this system (a Vandermonde matrix[2]) is extremely ill conditioned. One possibility would be to use integer values of u and then carry out the computations in exact rational arithmetic. This avoids roundoff problems, but even so, the number of bits needed to write Z completely will be of order $V + V\log(V)$, where $V = L^d$. This is certainly a possible procedure, but it will not be considered further here.

2. A second method is to implement the algorithm in symbolic arithmetic: that is, all quantities, instead of being real numbers, are polyno-

mials $p(u)$ (with integer coefficients) in the symbolic variable u. What one actually stores is, of course, the array of coefficients $\{a_k\}_{k=0}^{k_{\max}}$ corresponding to the polynomial $p(u)=\sum_{k=0}^{k_{\max}} a_k u^k$. The point is that the two fundamental operations in the Binder algorithm—multiplication by u and addition—can easily be implemented on polynomials: on the coefficient array $\{a_k\}$, they correspond simply to shift and vector addition, respectively.

Since operations on polynomials of degree $\leqslant k_{\max}$ are roughly $k_{\max}+1 \sim dL^d$ times more costly in storage and CPU time than operations on real numbers, the computational complexity of this algorithm is given by

$$\text{Storage} \sim 2n^{L^{d-1}} dL^d \tag{10a}$$

$$\text{Operations} \sim n^{L^{d-1}} ndL^{2d-1}L' \tag{10b}$$

3. The third method, which is the one used here, is a generalization of the second method and in fact includes both the first and the second methods as extreme special cases. It has the advantage that it reduces storage requirements, at the expense of extra computations.

Suppose that we carry out the algorithm in symbolic arithmetic, but now the dummy variable u is assumed to satisfy the additional relation

$$u^m = c \tag{11}$$

where m and c are integers. Then, all computations can be carried out with polynomials of *degree* $m-1$ *only*: any higher powers of u are "recycled" into the range $0 \leqslant k \leqslant m-1$ by inserting the appropriate factors of c. For example, if $p(u)$ is the polynomial

$$p(u) = \sum_{k=0}^{m-1} a_k u^k \tag{12}$$

then $up(u)$ is the polynomial

$$up(u) = \sum_{k=0}^{m-1} a'_k u^k \tag{13}$$

where

$$a'_k = \begin{cases} ca_{m-1} & \text{if} \quad k=0 \\ a_{k-1} & \text{if} \quad 1 \leqslant k \leqslant m-1 \end{cases} \tag{14}$$

Thus, multiplication *by* u, which is the crucial step [Eq. (4)] in the algorithm, amounts merely to a *circular shift and multiply* operation on the

$\{a_k\}$. Of course, this "folding" operation loses information: if at any stage of the algorithm, the true polynomial is

$$p(u) = \sum_{k=0}^{k_{\max}} a_k u^k \tag{15}$$

then this method will represent it by the "folded" polynomial

$$\hat{p}(u) = \sum_{k=0}^{m-1} \hat{a}_k u^k \tag{16}$$

where

$$\hat{a}_k = \sum_{j=0}^{\mathrm{Int}[(k_{\max}-k)/m]} a_{k+jm} c^j \tag{17}$$

In particular, the output of the algorithm will be the "folded" polynomial

$$\hat{Z}(u) = \sum_{k=0}^{m-1} I_m(k)\, u^k \tag{18}$$

whose coefficients are linear combinations of the N's multiplied by powers of c, namely,

$$I_m(k) = \sum_{j=0}^{\mathrm{Int}[(k_{\max}-k)/m]} N(k+jm)\, c^j \tag{19}$$

It is therefore necessary to perform runs at *several* values of the pair (m, c) and then combine the results (by solving a suitable linear system of equations) so as to reconstruct the $\{N(k)\}$. There are many possibilities on what values of c and m to choose. One extreme is to set $m = 1$ and run at $k_{\max} + 1$ distinct values of c; this is nothing but method 1 discussed above. Another extreme is to set $m = k_{\max} + 1$ (in which case c is irrelevant). This is just method 2. In general, the best choice is to fix m at the largest value that is allowed by the available computer memory and then run at several values of c. The values $c = 0$ and $c = 1$ are particularly convenient.

For Ising-like systems, $N(k)$ is symmetric about $k = k_{\max}/2$. One can therefore get all the $N(k)$ by taking $m = 1 + k_{\max}/2$ and $c = 0$. If there is not enough storage for this, then one could use $m = 1 + k_{\max}/4$ and do two runs, one for $c = 0$ and another for $c = 1$. If there are still problems of lack of adequate storage, then one can use smaller values of m and other integer values of $c > 1$. However, in this case, one must be careful about increasing the accuracy with which the I's are calculated, because this goes up as the

number of bits in c increases [see Eq. (19)]. For all the examples I have studied in this paper, I have used $c=0$ or $c=1$ and the maximum possible value of m so that I could generate all the N's using one or two values of c at the most.

We can now evaluate the computational complexity of this method: Suppose first that we take $m=1+[k_{\max}/2] \sim dL^d/2$ and $c=0$. Then, the requirements are

$$\text{Storage} \sim n^{L^{d-1}} dL^d \tag{20a}$$

$$\text{Operations} \quad \tfrac{1}{2} n^{L^{d-1}} ndL^{2d-1} L' \tag{20b}$$

If, on the other hand, we take $m=1+[k_{\max}/4] \sim dL^d/4$ and $c=0, 1$, then the storage is decreased by a factor of two from the above. However, notice that the total number of operations remains the same. This is because, although one has to do two runs at the different values of c, each run has only half as many operations. In general, if we take $m=1+[k_{\max}/2p]$, then one has to use $c=0, 1, 2, \ldots, p-1$. The storage requirements are decreased by a factor $[\log_2(p)]/p$, while the total number of operations remains the same.

Thus, the basic idea of the method is the opposite of solving the Vandermonde matrix. Instead of using several values of c and a fixed, small value of m, which would require one to solve an ill-conditioned system of linear equations, the idea is to use as large an m value as possible with some simple small values of c so that there are only very simple equations to solve to get the N's from the I's.

A few important points should be noted before considering examples of this technique. First, note that the I's will be large integers. One useful way of working with such large integers is to express them as a combination of two integers with an understood exponent e. Thus,

$$I=(j_1, j_2, e)=j_1+j_2 10^e \tag{21}$$

This allows one to extend the lattice sizes one can analyze, assuming of course that there is enough memory and computing power.

Further, the method as described can be used for systems with open or fixed boundary conditions or any combinations of these. One can also study systems where $d-1$ edges have periodic boundary conditions and the remaining one has either open or fixed boundaries. However, the method cannot deal with completely periodic boundary conditions without an enormous increase in storage [approximately the square of Eq. (7)].

I will now present results for the partition function of the two-dimensional Ising model on lattices with open boundary conditions. Table I gives

Table I. Partition Function of 2D Ising Model on 10×10 and 9×9 Lattices with Open Boundary Conditions[a]

k	$N(k)$, $L=10$	$N(k)$, $L=9$
0	2	2
1	0	0
2	8	8
3	80	72
4	228	190
5	480	432
6	2904	2372
7	10160	7776
8	28512	21634
9	94560	70544
10	334188	234492
11	1001600	684336
12	3024428	2025932
13	9390320	6080880
14	28416640	17769272
15	82962176	50670720
16	243286762	144436672
17	706898800	406911200
18	2023979520	1130610188
19	5729054800	3107180888
20	16122614142	8461908280
21	44911259552	22781518288
22	123963179176	60668035684
23	339237864112	159839880584
24	921006159792	416592162178
25	2478596355488	1073633622112
26	6615343595116	2735962636896
27	17508949583072	6892649260024
28	45956908921096	17162325923036
29	119601063089488	42224935085024
30	308624008346184	102625819853840
31	789531362631936	246330597987560
32	2002236996085046	583745350990298
33	5032702605600208	1365335930718616
34	12536459497288912	3150846295183988
35	30943134757697456	7172032090649952
36	75666085125011546	16096607390244494
37	183277438049400480	35608145031851608
38	439653074154090240	77611845404974452
39	1044290798751834512	166612094830696616
40	2455601601917768420	352141101485111164

[a] The k's are the possible energies and $N(k)$ is the exact number of states at that energy. For the Ising model, $N(k)=N(k_{\max}-k)$ with $k_{\max}=2L(L-1)$. Hence, only $N(k)$ for $k \leqslant k_{\max}/2$ are listed.

Table I. *(Continued)*

41	5715191389040815680	732463626079434728
42	13162814059194332284	1498779830132023700
43	2999284338439276256	3015720617514065464
44	67599071186609390360	5964319529107745650
45	150667233135443195280	11589328787828817456
46	33200928462077092530 4	22115178532979996228
47	72315438404544375520 0	41424843026785397752
48	155652080746729033078 0	76132386821095360478
49	330988948676824255310 4	137218510027713547336
50	695178061506756392751 2	242428684682846167112
51	144174964490484550861 44	419637104618984345232
52	295175295544460206999 14	711328701410891951820
53	596416242457456972380 192	1180209924661787853816
54	118899403477259922266 688	1915688893225844215304
55	233802595837168308698 272	304054048074093267627 2
56	453352171299545237693 512	471649828352941924719 8
57	866590747690830006960 960	7146857891458317367016
58	163251732830652022411 0480	10573581547941149882100
59	302997199321928496527 3792	1526609784706369752896
60	553893739277220586453 1480	21499288224546805917152
61	996987431829577805700 9504	29519329716772396767416
62	176642477776931751607 65184	39498272853693385575900
63	307969930309904332090 84768	51481355461780817578352
64	528193923128856785454 36692	65334580863611097487494
65	890868356751834030182 00384	80703121534458107878000
66	147717156491450787191 527200	96992501635739079414728
67	240717216720664582618 430368	113383198675792819701304
68	385392505420278103285 005328	128884747013326018242742
69	606010560730521174349 765088	142427993043658173714192
70	935618924404472613615 285232	152985286453100484078072
71	141782044521988029486 8836160	159700376205906735578624
72	210818803698121434989 0853640	162005092561274395844676
73	307487233263778671853 8528128	
74	439784611534828934826 6362648	
75	616617791137274185464 6839616	
76	847278009492205619259 6717424	
77	114062783051393677416 82022656	
78	150400257416619189623 36070720	
79	194188303368953319011 59821792	
80	245446177883342809859 35169068	
81	303629066145527566887 51417312	
82	367524648351340692784 62022192	
83	435206768584207429555 89180512	
84	504068278357795896648 89986348	
85	570946465381959698513 85893856	
86	632340971125784331001 49657552	
87	684707764399371105493 10951936	
88	724796759410225413433 44445160	
89	749988741490507403509 04095104	
90	758582643620087053889 32311560	

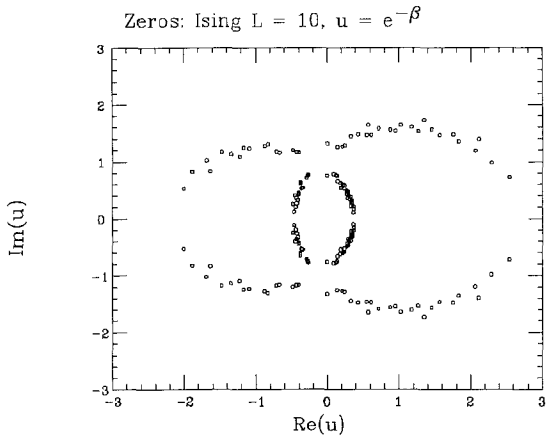

Fig. 3. The zeros of the Ising model on a 10×10 lattice with open boundary conditions in the $u = e^{-\beta}$ complex plane.

$N(k)$ versus k for lattice sizes 10×10 and 9×9. Since $N(k)$ is symmetric for the Ising model [$N(k) = N(k_{\max} - k)$, $k_{\max} = 2L(L-1)$], I have listed only half the $N(k)$ values in Table I. Note that the number of states on a 10×10 lattice is 2^{100}, which is a rather large number.[3] The amount of computer time necessary to generate Table I was about 1 CPU min on a 2-processor Cray XMP. Whereas there is no intrinsic interest in studying the 2D Ising model numerically, one can use these partition functions to see if any useful predictions for the exponents and critical temperature can be made from

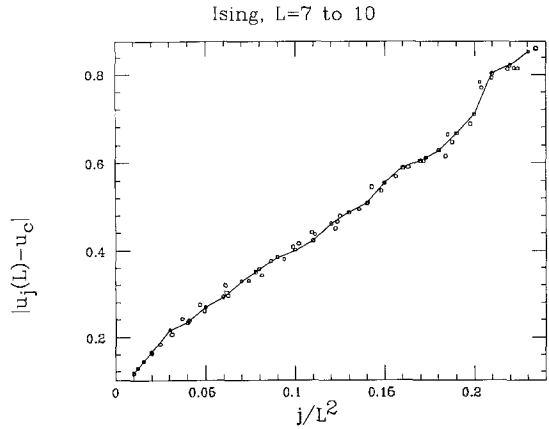

Fig. 4. Test of scaling law [Eq. (23)] for the 2D Ising model. All the zeros in the first quadrant inside the unit circle for $L = 7$ to 10 have been used in the plot. The line connects data points corresponding to $L = 10$.

exact results on such small systems. In general, I found that if the problem can be made to fit in the computer memory, then the method was so efficient that the CPU time required was only a few seconds.

Since Z is a polynomial in u, its analytic structure is completely determined by its zeros.[4),3] Figure 3 shows all the zeros of the 10×10 Ising system in the complex u plane. Itzykson *et al.*[6] showed that the distance from the zero closest to the $\mathrm{Re}(u)$ axis scales with lattice size L as $L^{-1/\nu}$. Thus, if u_c is the infinite-volume critical point and $u_1(L)$ is the zero closest to the $\mathrm{Re}(u)$ axis, then,

$$\mathrm{Im}[u_1(L)] \sim L^{-1/\nu}[1 + O(L^{-\omega})] \tag{22a}$$

$$|u_1(L) - u_c| \sim L^{-1/\nu}[1 + O(L^{-\omega})] \tag{22b}$$

and

$$\mathrm{Re}[u_1(L)] - u_c \sim L^{-1/\nu}[1 + O(L^{-\omega})] \tag{22c}$$

In ref. 6 a stronger version of this scaling law is also derived:

$$|u_j(L) - u_c| \sim \left(\frac{j}{L^d}\right)^{1/d\nu} \tag{23}$$

where $j = 1, 2, 3,...$ label the zeros in order of increasing distance from u_c. Figure 4 shows the test of the scaling law of Eq. (23) for the 2D Ising model.

For this model,

$$u_c = \sqrt{2} - 1 = 0.414213... \qquad \text{and} \qquad \nu = 1 \tag{24}$$

One possible procedure to extract ν and u_c from the zeros is to estimate ν from successive L values using Eq. (22a) and then use some finite-size extrapolation procedure to find the infinite-volume value of ν. After that, one could use Eq. (22c) to find u_c. Table II lists the zeros $u_1(L)$ for $L = 2, 3,..., 10$ and estimators $\nu(L)$ for ν defined by

$$\nu(L) = -\frac{\log[L/(L+1)]}{\log\{\mathrm{Im}[u_1(L)]/\mathrm{Im}[u_1(L+1)]\}} \tag{25a}$$

Two alternate estimators for ν are

$$\nu(L) = -\frac{\log[L/(L+1)]}{\log[|u_1(L) - u_c|/|u_1(L+1) - u_c|]} \tag{25b}$$

[3] For some recent numerical studies of these zeros and for references see ref. 5.

Table II. The Zero Closest to the Re(u) Axis for Various L for the Ising Model and the Estimate $v(L)$ of v from Eq. (25a)[a]

L	$\mathrm{Re}(u_1(L))$	$\mathrm{Im}(u_1(L))$	$v(L)$
2	0.00000000000	0.41421356237	1.327491
3	0.20124863451	0.30519306466	1.163236
4	0.26982689144	0.23832440110	1.132630
5	0.30440253447	0.19570710834	1.116694
6	0.32534008878	0.16622629444	1.105160
7	0.33943333762	0.14458498201	1.095899
8	0.34959422031	0.12799880987	1.088160
9	0.35728151256	0.11486763746	1.081566
10	0.36330845658	0.10420557321	

[a] The extrapolated average value obtained by the BST procedure is $v = 1.0033(21)$.

and

$$v(L) = -\frac{\log[L/(L+1)]}{\log\{\mathrm{Re}[u_1(L)-u_c]/\mathrm{Re}[u_1(L+1)-u_c]\}} \tag{25c}$$

these estimators require, however, an *a priori* knowledge of u_c. Also, it is obvious from Eq. (22) that for any of these estimators,

$$v(L) = v[1 + O(L^{-\omega})] \tag{26}$$

The extrapolation procedure that I have found to work best is that of Bulirsch and Stoer[7] (BST), which I will now briefly describe. For a general discussion about this method, see ref. 8. The idea of the extrapolation procedure is that if $T(h)$ is a function with an expansion

$$T(h) = T_0 + a_1 h^{-\omega} + a_2 h^{-2\omega} + \cdots \tag{27}$$

and h_N ($N = 0, 1, 2, \ldots$) is a sequence converging to zero as $N \to \infty$, then, given the values $T(h_N)$ for a sequence of values of h_N, the desired limit T_0 is obtained from a sequence of extrapolants,

$$\begin{matrix} T_0^{(0)} & & \\ T_0^{(1)} & T_1^{(0)} & \\ T_0^{(2)} & T_1^{(1)} & T_2^{(0)} \end{matrix} \tag{28}$$

etc., where the $T_m^{(N)}$ are defined as follows:

$$T_{-1}^{(N)} = 0, \qquad T_0^{(N)} = T(h_N) \tag{29a}$$

$$T_m^{(N)} = T_{m-1}^{(N+1)} + (T_{m-1}^{(N+1)} - T_{m-1}^{(N)}) \times \left[\left(\frac{h_N}{h_{N+m}} \right)^{\omega} \left(1 - \frac{T_{m-1}^{(N+1)} - T_{m-1}^{(N)}}{T_{m-1}^{(N+1)} - T_{m-2}^{(N+1)}} \right) - 1 \right]^{-1} \tag{29b}$$

Using this algorithm on the $v(L)$ estimates of Table II, one gets the asymptotic estimate

$$v = 1.0009,\ 1.0061,\ 1.0028 \tag{30}$$

from using the data for $L = 2$–10, $L = 3$–10, and $L = 4$–10, respectively. The extrapolation is not sensitive to the choice of ω and the results quoted are for $\omega = 1.0$. Using $v = 1$, one can now compute two estimates for u_c by using either Eq. (22c) or $|u_1(L)| \sim u_c + AL^{-1/v}$ via the BST procedure just described. One finds

$$u_c = 0.414200(11) \tag{31}$$

in excellent agreement with the exact result [Eq. (24)].

I will now generalize the discussion to models where the site variables take on n values. For concreteness and to keep the notation simple, I will discuss $Z(n)$ models in two dimensions. The generalization to arbitrary dimension and models should be obvious after the discussion that follows.

A $Z(n)$ configuration in one dimension is a set S of L numbers $\{n_1, n_2, ..., n_L\}$, $n_i \in [0, n-1]$. The n_i label the $Z(n)$ angles $\theta_i = 2\pi n_i/n$. The one-dimensional system has n^L states, which may conveniently be labeled by an integer index,

$$I_S = n_1 + nn_2 + n^2 n_3 + \cdots + n^{L-1} n_L \in [0, n^L - 1] \tag{32}$$

Define two arrays $Z^o(S)$ and $Z^n(S)$ with as many storage locations as states of the 1D system (n^L). The analogue of the iteration step of Eq. (4) that adds one spin n_i at location i is

$$Z^n(S) = \sum_{n_j=0}^{n-1} e^{-\beta f(n_i, n_j)} Z^o(S') \tag{33}$$

where, if

$$S = \{n_1, n_2, ..., n_i, ..., n_L\} \tag{34}$$

$$S' = \{n_1, n_2, ..., n_j, ..., n_L\} \tag{35}$$

Table III. Partition Function of 8×8 and 7×7 Potts Models[a]

k	$N(k)$, $L=8$	$N(k)$, $L=7$
0	3	3
1	0	0
2	24	24
3	192	168
4	552	462
5	2088	1824
6	11304	8928
7	41664	31392
8	148386	111684
9	587688	423216
10	2193300	1510572
11	7720728	5209944
12	27537474	18065964
13	97132296	61574472
14	334271004	206061780
15	1137947376	682080864
16	3843421530	2231687358
17	12818773296	7204443672
18	42298227456	22974767664
19	138314323008	72364170912
20	448089240318	225026403222
21	1438146998664	690610737672
22	4574963788656	2091155349516
23	14425679275296	6243838334808
24	45084393148710	18373853627526
25	139648935821064	53255451879384
26	428680498713156	151934469419988
27	1303884760918824	426346021158288
28	3928871392273452	1175846544042816
29	11725097717341728	3184673584497912
30	34647163889868624	8463104247723672
31	101341900685306232	22046901347978448
32	293316951182710806	56247151958954040
33	839762821791103608	140394144730039800
34	2377287574813856616	342477392173729128
35	6651755879018772480	815570022852671736
36	18388012681028246742	1893850948922891904
37	50197507759057494528	4282946591792482368
38	135261530391980304528	9421173493821644760
39	359582171737685788248	20130654579263172240
40	942609542987942012364	41725664029037517546
41	2435256118061086047552	83776129533272914272
42	6197222628225402131196	162692100725406071196

[a] The k's are the possible values of the energy and $N(k)$ is the number of states at that energy. The maximum possible energy is $k_{\max} = 2L(L-1)$.

Table III. *(Continued)*

43	15525331268690663280216	305125905414307569912
44	38266501591607687448702	551794823492840197644
45	92739457153227664217544	960654116676351255216
46	220852825261841467774536	1607467280678385203208
47	516475249506545288311200	2581029757581048927600
48	118525131679499442762 7662	3970194201945262150176
49	266736599253951880741 5072	584117360043988320 3384
50	588238461598864290047 6148	8206876906647453549192
51	127028397360614404630 70376	10994910494946859758312
52	268406689465897542596 76888	14025648539287777952328
53	554485458523491789854 02152	17013565092668829899472
54	111902904244440315648 204204	19601081474242806920 8468
55	220438862509400023281 234456	21423923022906475914096
56	42350975584491196613 1943386	22193482879885189229832
57	79285211439477291481 3063968	21771179708605905705072
58	14450820827085630032 00717988	20208583131061984494552
59	25619840403514842747 12667992	17737512051220595306352
60	44141907707156458253 33311376	14712540247365904408134
61	73844878231036092495 86152728	11525918044389093098712
62	11983575350216845651 701030252	8523526812396935480436
63	18847406776901530038 840755136	5946726105595443369456
64	28702569821938615406 283871536	391194360013679726757 0
65	42286674107265855479 396703384	242478064097136878505 6
66	60216659100912854864 184775656	141507375643515471631 2
67	82810752689432279070 373530696	776800161495415847832
68	10988857969543391167 6772905944	400660425263642790030
69	14059423918427147297 0682554376	193904875439535065760
70	17330124512935568666 1942833760	87907143578625315684
71	20565699466755136062 5028442104	37256344099024673568
72	23480160801252358318 6410329658	14724351321410949000
73	25775257021749879166 2403969224	5410091775761568144
74	27189330155845218228 9236489328	1841059284810225696
75	27545903792747917277 7454264728	577550284153252512
76	26789664144579760234 2148889448	166040211143267904
77	24999750675749995016 7839421744	43418679825191256
78	22376099259647609833 1786106996	10226640750772884
79	19202098202622158459 9113086888	2141358146070576
80	15793346939341846913 3577041844	391387088304576
81	12445530842659764339 4601582784	60773967560208
82	93933916098635721361 446124176	7674121598604
83	67882328800513023649 649676960	725187096504
84	46953403232080538221 573099950	41869995708
85	31073935957604141376 395970168	
86	19668537504816747494 192265600	
87	11901584764535848096 088822256	
88	68814636918760480427 49443808	
89	37997691653333953308 05121288	

Table III. *(Continued)*

90	2002421821386729970447589172
91	1006364364920951693202042480
92	481932011135808455486626278
93	219693973804473428011517736
94	952258357931207950372832 76
95	391938310622652634184091 60
96	152944627226356068191460 42
97	564832312506623086725964 8
98	196994949085781559734856 0
99	647233697269536071777568
100	199738188211753448316636
101	57694563236897928233520
102	15532995049768573552992
103	3877925335950604678368
104	892119601906523571432
105	18761654785820482 8816
106	35701326530842588188
107	6063375007720464072
108	901707099455080434
109	114132122957405952
110	11738897145225828
111	897937832100888
112	40724629633188

and $f(n_i, n_j)$ is the energy of a pair of nearest neighbor spins with $Z(n)$ angles specified by the integers n_i, n_j. One must perform the iteration step of Eq. (33) once over all the S's for each spin added. In fact, Eq. (33) is valid for any model and can even be generalized to continuous models by replacing the sum by an integral. However, the procedure is simplest for discrete models and when f is an integer-valued function, so that Z is a polynomial in $u = e^{-\beta}$. I will restrict myself to this case only.

Specifically, for the $Z(n)$ model, one can make f have the required property and still remain in the correct universality class. Define

$$f(n_i, n_j) = \min[|n_i - n_j|, n - |n_i - n_j|] \tag{36}$$

This function has the merit of restricting the energy to integer values. One can also write f as a finite Fourier series in terms of the $Z(n)$ angles $\theta_{ij} = 2\pi(n_i - n_j)/n$. For even n,

$$\begin{aligned} f(n_i, n_j) &= f(n_i - n_j) = f(\theta_{ij}) \\ &= \frac{n}{4} - \sum_{k=1}^{n-1} \frac{1-(-1)^k}{n} \frac{1}{1-\cos(2\pi k/n)} \cos(k\theta_{ij}) \end{aligned} \tag{37}$$

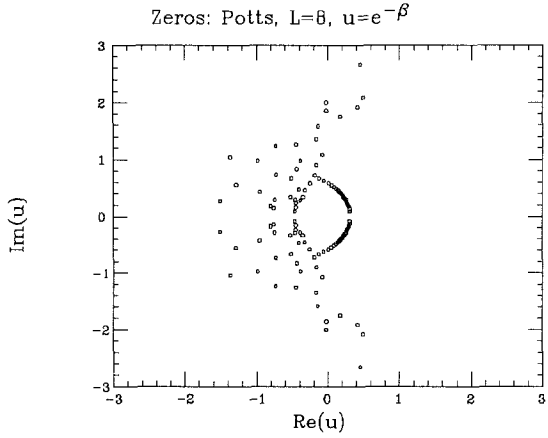

Fig. 5. Zeros of the 8×8 Potts model with open boundary conditions in the complex $u = e^{-\beta}$ plane.

It is obvious that the theory defined by the energy function of Eq. (36) has the same vacuum state as the usual $Z(n)$ theory because the ground state has all spins aligned.

As a nontrivial example, consider the $Z(3)$ model (also called the 3-state Potts model[4]). Here, $n_i = 0, 1, 2$ and

$$f(n_i, n_j) = 1 - \delta(n_i, n_j) \in [0, 1] \tag{38}$$

[4] See ref. 9 for a comprehensive view of the Potts model.

Table IV. The First Zero for the Potts Model for Various L Values and the Estimates $v(L)$ of v from Eq. (25)[a]

L	$\mathrm{Re}[u_1(L)]$	$\mathrm{Im}[u_1(L)]$	$v(L)$ Eq. (25a)	$v(L)$ Eq. (25b)	$v(L)$ Eq. (25c)
2	0.02906309100	0.28739427521	1.536578	0.9506645	0.6866508
3	0.17933073658	0.22073871857	1.177574	0.9823404	0.7780325
4	0.23703749401	0.17289408198	1.091044	0.9816150	0.8231159
5	0.26766654497	0.14091489903	1.045968	0.9731017	0.8469487
6	0.28671621031	0.11837377826	1.016353	0.9632042	0.8602000
7	0.29972800675	0.10171522195	0.994788	0.9537328	0.8677652
8	0.30918364569	0.08893857422			

[a] The exactly known value of $u_c = (\sqrt{3} - 1)/2$ is used in columns 5 and 6. Using the BST procedure, one obtains from columns 4, 5, and 6 that $v = 0.8386$, 0.8328, and 0.8479 from the $L = 2$–8 data and $v = 0.8342$, 0.8445, and 0.8399 from the $L = 3$–8 data.

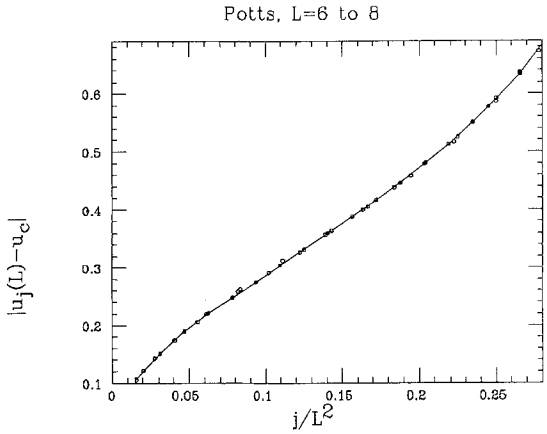

Fig. 6. Test of scaling law [Eq. (23)] for the Potts model on lattices of size $L = 6$–8. All zeros in the first quadrant are included. The straight line connects the $L = 8$ data.

Table III shows the partition function of this model computed by the method described above for 7×7 and 8×8 lattices. Note again that a very large number of states ($\sim 2^{100}$) has been generated. Figure 5 plots all the zeros for the 8×8 lattice and Table IV lists the first zero for various L values and estimates of $\nu(L)$ using Eq. (25). The critical coupling and the exponents of this model are also known: $u_c = (\sqrt{3} - 1)/2$ and $\nu = 5/6$. Using the BST procedure, one can extrapolate the $\nu(L)$ values of Table IV to $L = \infty$ and estimate ν. The results are given in Table IV and are surprisingly close to the correct result, given the small lattice sizes used.

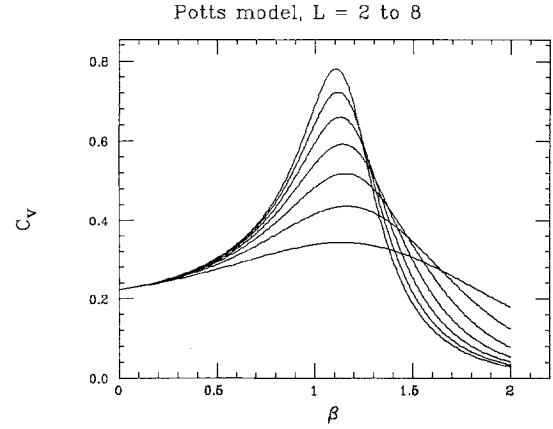

Fig. 7. The specific heat at constant volume C_v of the Potts model as a function of β for various lattice sizes ($L = 2$–8).

Figure 6 is a test of the scaling law of Eq. (23) and the scaling seems to work much better for the Potts model than for the Ising model. Figure 7 plots the specific heat of the model, defined by

$$C_v = \frac{\partial^2 \log Z}{\partial \beta^2} \tag{39}$$

The peak of the specific heat scales like $L^{\alpha/\nu}$ and Table V shows the β value at the peak, C_v at the peak, and the critical exponent estimates extracted from the C_v peak. These estimates of $\beta_c(L)$ and $\alpha(L)/\nu(L)$ can be extrapolated to $L = \infty$ to give estimates of β_c and α/ν, which are given in Table V and are again very close to the correct values.

Finally, ref. 6 also relates the critical exponent α to the angle ϕ that the line of zeros of Z makes with the $\mathrm{Re}(u)$ axis and the ratio A_-/A_+ of the specific heat amplitude below and above the critical temperature (which is a universal quantity). The result [ref. 6, Eq. (1)] is

$$\tan[(2-\alpha)\phi] = \frac{\cos(\pi\alpha) - A_-/A_+}{\sin(\pi\alpha)} \tag{40}$$

Figure 8 shows the angle ϕ versus $1/L$. The horizontal line is the infinite-volume extrapolation, which gives $\phi = 0.968(2)\,\pi/2$. The error comes from the extrapolation and is estimated by using different sets of L values to make the extrapolation. If one uses this result for ϕ in Eq. (40), one finds $A_-/A_+ = 1.10(07)$. Since this value is so close to unity and the angle ϕ is so close to $\pi/2$, I would conjecture that for the Potts model,

$$\phi = \pi/2 \qquad \text{and} \qquad A_-/A_+ = 1 \tag{41}$$

Table V. The Location of the Peak of the Specific Heat C_v and Its Maximum Value as a Function of L[a]

L	β(peak)	C_v(peak)	$\alpha/\nu(L)$
2	1.1345095	0.3422606	0.59470295
3	1.1639425	0.4355910	0.60281669
4	1.1548822	0.5180767	0.59824105
5	1.1410812	0.5920653	0.59139593
6	1.1279001	0.6594731	0.58428310
7	1.1163123	0.7216273	0.57747388
8	1.1063324	0.7794743	

[a] The last column gives estimates of the ratio α/ν from the position of the specific heat peak. The extrapolated result is $\alpha/\nu = 0.453(7)$, as compared to the exact value 0.4. The extrapolated value of β_c is 0.9987(18), to be compared with the exact result 1.00505... .

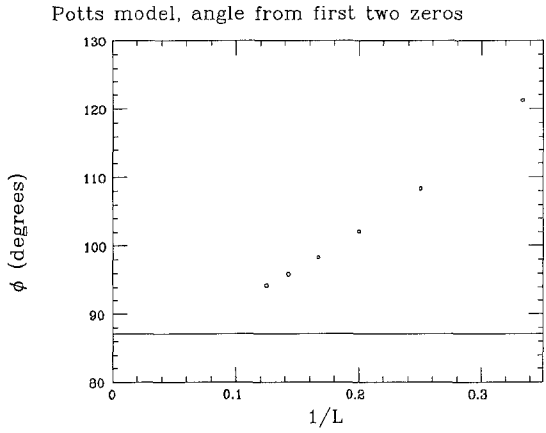

Fig. 8. The angle ϕ at which the extrapolated line of zeros intersects the Re(u) axis. The horizontal line is the extrapolated $L = \infty$ result using the BST formula [Eq. (29)].

In summary, I have described a method that can compute the partition function of certain classes of discrete models (those whose energy takes integer values). Although one is limited to small lattices, the lattice sizes seem sufficient to extract useful information about critical couplings and exponents. One might even hope that by studying the analytic structure of the theory via the zeros of the partition function on small volumes, one might be able to guess the complete analytic solution. In fact, the zeros in Fig. 5 near the Re(u) axis seem to behave very much like a conic section. This is also in agreement with the observation that the angle ϕ at which the line of zeros meets the Re(u) axis is almost $\pi/2$. If this is borne out by studies on larger lattices, one could write down the analytic form for the critical part of the free energy of the Potts model in 2 dimensions. This point is currently being studied.

Finally, the algorithm described in this paper has some similarity with the manipulation of very large integers using residue arithmetic modulo several different primes.[10]

ACKNOWLEDGMENTS

This work was supported in part by the U.S. Department of Energy under contract number DE-FC05-85ER250000. I would like to thank the TH Division at CERN, Ecole Normale Supérieure and CEN, Saclay for their hospitality. I had very useful discussions about the ideas presented here with Alain Billoire, Malte Henkel, Apoorva Patel, John Richardson, and Roman Salvador, which are gratefully acknowledged.

REFERENCES

1. K. Binder, *Physica* **62**:508 (1972); see also A. J. Chorin, *Commun. Math. Phys.* **99**:501 (1985); Lei Gong-yan, *J. Comput. Phys.*, to appear.
2. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computing* (Cambridge University Press, Cambridge, 1985).
3. R. B. Pearson, *Phys. Rev. B* **26**:6285 (1982).
4. C. N. Yang and T. D. Lee, *Phys. Rev.* **87**:404, 410 (1952); M. Fisher, in *Lectures in Theoretical Physics*, Vol. 12C (University of Colorado Press, Boulder, Colorado, 1965), p. 1.
5. P. Carter and G. Bhanot, *Nucl. Phys. B* **5A**(Proc. Suppl.):334 (1988).
6. C. Itzykson, R. B. Pearson, and J. B. Zuber, *Nucl. Phys. B* **220**:415 (1983).
7. R. Bulirsch and J. Stoer, *Numerical Math.* **6**:413 (1964).
8. M. Henkel, *J. Phys. A: Math. Gen.* **21**:2617 (1988).
9. F. Y. Wu, *Rev. Mod. Phys.* **54**:235 (1982).
10. D. E. Knuth, *The Art of Computer Programming* (Addison-Wesley, 1981), Vol. 2, Section 4.3.2.