Journal of Statistical Physics, Vol. 36, Nos. 1/2, 1984

A Transfer Matrix Program to Calculate the Conductivity of Random Resistor Networks

B. Derrida,[1] J. G. Zabolitzky,[2] J. Vannimenus,[3] and D. Stauffer[2]

Received November 29, 1983

We describe technical details on a new method of calculating the conductivity of random resistor networks which uses transfer matrix ideas. We give a program which calculates the conductivity of three-dimensional bars, and we provide a few comments on the advantages of this method and its performances.

KEY WORDS: Percolation; transfer matrix; random resistor network; FORTRAN program; vector computer.

1. INTRODUCTION

The problem of calculating the conductivity of random resistor networks by numerical means has motivated in the past a lot of efforts. Up to now the two main approaches which were used are the following: the resolution of Kirchoff's equations by relaxation methods[1-6] and techniques using the node elimination ideas.[6-10] The main advantage of the relaxation method is that the code is very simple. However it is time consuming and the convergence may be very slow.

There exist several variants of the node elimination method. They usually consist in eliminating the dangling bonds and the disconnected clusters and then in transforming the network in a systematic way which reduces the number of resistors without changing the conductivity. The conductivity of a given sample is calculated exactly. However the code is rather sophisticated and the calculations remain time consuming.

[1] Service de Physique Théorique, Institut de Recherche Fondamentale, Departement de Physique Theorique, Commissariat a L'Energie Atomique, CEN Saclay, 91191 Gif-sur-Yvette Cedex, France.

[2] Institüt für Theoretische Physik, Universität, D-5000 Köln, West Germany.

[3] Ecole Normale Supérieure, 24 rue Lhomond, F-75231 Paris Cedex 05, France.

0022-4715/84/0700-0031$03.50/0

The purpose of the present paper is to publish technical details on a new method based on transfer matrix ideas. This method was used already in[11,12] to find estimates of the conductivity exponent t of a diluted network of resistors in dimensions two and three. Since we think that the same method can be applied to several other situations (random mixture of conducting and superconducting bonds, diffusion on a random lattice, . . .) we want here to publish a version of our complete program and give some comments which make it understandable.

The main difference between our method and previous ones[1-10] is that the conductivity is calculated *exactly* at the same time as the lattice is constructed.

We think that the method presented here to calculate the conductivity of random resistor networks has the following advantages:

(a) The conductivity of a given sample is calculated exactly and the calculation never fails to converge.

(b) We have used our method for strip or bar geometries. However it works also for systems finite in all directions (squares, cubes) and for higher dimensions.

(c) The program does not require any library subroutine except a random number generator.

(d) It can be used for any distribution of random resistors. The fact that some resistors are infinite, i.e., some bonds are missing can be used to speed up the calculations. However it is not at all an essential condition. In particular, we plan to use the same method to study the case of a random mixture of conducting and superconducting bonds.

(e) We do not need to look for the geometrical properties of the conducting clusters. We do not need to worry about dangling bonds or isolated clusters as with other techniques.

(f) We can calculate the conductivity of very long strips. Therefore we do not need to average the conductivity over several samples and so we avoid the usual problem of deciding how one should average the conductivity.

(g) We construct the lattice element by element and at each time that we add a new bond or site, we calculate immediately its contribution to the conductivity. Therefore we never need to store the whole lattice in the memory. In this sense our method is analogous to that of Hoshen and Kopelman to count clusters.[13]

2. THE ALGORITHM

Let us now describe the algorithm that we use to calculate the conductivity. We shall describe here our algorithm in the two-dimensional

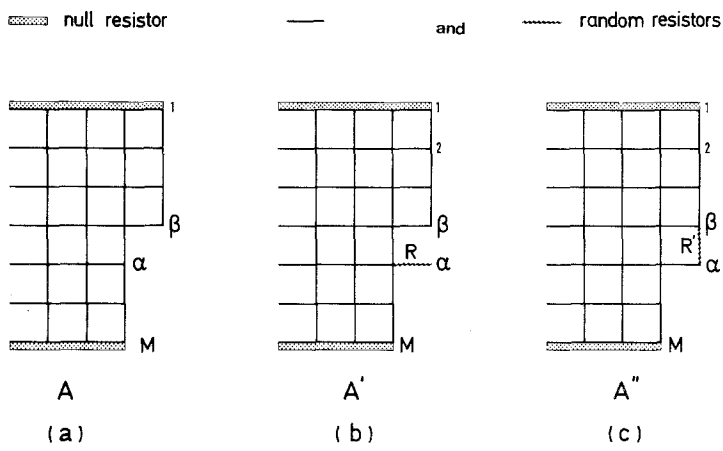

Fig. 1. The strip is constructed bond by bond. At each time that we add a new resistor, the matrix A is modified.

case. However the generalization to the three-dimensional case is very easy. We imagine our strip to be oriented horizontally, with a unit voltage applied vertically, and describe here the case of bond percolation.

At each stage of the construction of our strip, we describe the left part of the strip by a matrix A which is a $M \times M$ matrix (if M is the number of sites in a section, including the site of the first row and the site of the last row). The matrix A can be defined in the following way.[11] If we want to measure the electrical properties of the left part of a strip, we can attach to each site of the section (sites $1, 2, \ldots, M$) a wire which imposes a voltage V_α on site α (see Fig. 1a of the present paper and also Fig. 1 of Ref. 11). Then since the V_α are arbitrary, a current I_β will flow in the wire attached to site β. The problem is linear, therefore the currents I_β are linear functions of the voltages V_α:

$$I_\beta = \sum_\alpha A_{\beta\alpha} V_\alpha \tag{1}$$

Since the currents I_β depend only on the differences between the V_α, we can always choose $V_M = 0$. The only thing we need to do is to write how the matrix A is transformed when one adds one new resistor. In two dimensions, there are two kinds of resistor one can add.

If one adds an horizontal resistor R, on site α (see Fig. 1b), then the matrix A becomes a new matrix A'. One can show that the matrix elements of the new matrix A' are related to those of A by

$$A'_{ij} = A_{ij} - \frac{A_{i\alpha} A_{\alpha j} R}{1 + A_{\alpha\alpha} R} \tag{2}$$

Similarly, if one adds a vertical resistor R' between the sites α and β of (Fig. 1b), the matrix A' becomes a new matrix A'' given by

$$A''_{ij} = A'_{ij} + \frac{(\delta_{\alpha,j} - \delta_{\beta,j})(\delta_{\alpha,i} - \delta_{\beta,i})}{R'} \tag{3}$$

where δ_{ij} is Kronecker's symbol ($\delta_{i,j} = 1$ if $i = j$ and $\delta_{i,j} = 0$ otherwise).

We can, at this stage, make the following remarks.

(a) The first and the Mth horizontal rows are infinitely conducting, thus for $\alpha = 1$ or $\alpha = M$, $R = 0$. Therefore, for $\alpha = 1$ and M, equation (2) becomes very simple: $A'_{ij} = A_{ij}$, i.e., the matrix A remains unchanged.

(b) The matrix A is symmetric. This can be used to make the calculations shorter. However, the program given below does not use this property because it makes the vectorization of the program on the Cray more difficult.

(c) If for some α the element $A_{\alpha\alpha} = 0$, this means that the site α is not connected to any other site of the section by the left part of the strip. Therefore all the elements $A_{\alpha i} = A_{i\alpha} = 0$ for any i. So if $A_{\alpha\alpha} = 0$, equation (2) becomes $A'_{ij} = A_{ij}$ independent of the new bond R.

(d) All the bonds (which are not on the first or the Mth horizontal rows) have a resistance which is either 1 or ∞. If the resistance $R = 1$, we have to use formulas (2) and (3). It is clear that if the resistance R is infinite formula (3) becomes just $A''_{ij} = A'_{ij}$ and therefore the matrix A remains unchanged. In the case of equation (2), it becomes for $R = \infty$:

$$A'_{ij} = A_{ij} - \frac{A_{i\alpha} A_{\alpha j}}{A_{\alpha\alpha}} \tag{4}$$

[see remark (c) if $A_{\alpha\alpha} = 0$].

(e) We calculate the conductivity of a strip of length L for $L \gg 1$. The conductivity σ per unit length is given by

$$\sigma = \lim_{L \to \infty} \frac{A(1,1)}{L} \tag{5}$$

This formula can easily be understood: to measure the conductivity of the strip, we want to calculate the current which flows when one applies a voltage V on the first row and a voltage 0 on the Mth row. Equation (1) defines this current as $A(1,1)$ if a unit voltage is applied to the topmost row $V_1 = 1$ and zero voltages everywhere else on the column L; i.e., $V_i = 0$ for $2 \leqslant i \leqslant M$. This gives the conductivity σ in the limit $L \gg 1$ since the fact that we put the sites $2 \leqslant i \leqslant M - 1$ of column L at voltage 0 gives just a finite contribution to current I_1, whereas the strip of length L gives a contribution to I_1 which increases linearly with L for $L \gg 1$. One could choose other boundary conditions at column L like imposing the currents I_i

to be zero for the sites $2 \leqslant i \leqslant M - 1$. This would make the program slightly longer without changing the results for large enough L.

3. THE PROGRAM

In this section, we give the program that we have used to study the conductivity of random resistor networks in three dimensions. The program published here can be used as well on a CDC Cyber 76 as on a Cray 1 or a Cyber 205.

```
      PROGRAM  MAIN(INPUT, OUTPUT, TAPE5 = INPUT, TAPE6
     $ = OUTPUT)
      DIMENSION A(198, 198), V(198), NOC(198)
C     PARAMETERS
      PRO = .3117
      N = 14
      L = 200
      LINIT = L/10
      LPRINT = LINIT/5
      NY = N
      NZ = N
          WRITE(6, 1000)NY, NZ, PRO, L, LINIT
1000      FORMAT(//2I5, F12.5, 2I12/)
C     INITIALISATION
      NZP1 = NZ + 1
      NS = NY * NZ
      NSP1 = NS + 1
      NSP2 = NS + 2
      NBZP1 = NY * NZP1 + 1
      RINIT = 0.
      DO 10 K1 = 1, NSP2
      NOC(K1) = 0
      DO 10 K2 = 1, NSP2
10    A(K2, K1) = 0.
      NOC(1) = 1
      NOC(NSP2) = 1
      CALL RANSET(1)
C     START OF THE CALCULATION
      DO 20 KL = 1, 2
      NX = (KL - 1) * L + (2 - KL) * LINIT
      JTER = 0
        DO 30 ITER = 1, NX
        JTER = JTER + 1
```

```
C  ADDITION OF BONDS IN THE DIRECTION X
          DO 100 K3 = 2, NSP1
          NQ = NOC(K3)
          NOC(K3) = 0
          IF(RANF(K3).LT.PRO)NOC(K3) = 1
          D = A(K3, K3)
          IF(D.LT.1.E - 8)GO TO 100
            NQ = NQ * NOC(K3)
            D = 1./(D + NQ)
            DO 110 K2 = 1, NSP1
110         V(K2) = A(K2, K3)
            DO 120 K1 = 1, NSP1
            D1 = A(K3, K1)
            IF(D1 * D1.LT.1.E - 16)GO TO 120
            D1 = D1 * D
              DO 130 K2 = 1, NSP1
130           A(K2, K1) = A(K2, K1) - V(K2) * D1
120         CONTINUE
100       CONTINUE
C  ADDITION OF BONDS IN THE DIRECTION Y
          DO 200 J1 = 1, NY
          DO 210 J2 = 1, NZ
          K1 = J1 + NY * (J2 - 1) + 1
          IF(NOC(K1).EQ.0)GO TO 210
          K2 = K1 + 1
          IF(J1.EQ.NY)K2 = K2 - NY
          IF(NOC(K2).EQ.0)GO TO 210
          A(K1, K1) = A(K1, K1) + 1.
          A(K2, K1) = A(K2, K1) - 1.
          A(K2, K2) = A(K2, K2) + 1.
          A(K1, K2) = A(K1, K2) - 1.
210       CONTINUE
200       CONTINUE
C  ADDITION OF BONDS IN THE DIRECTION Z
          DO 300 J1 = 2, NBZP1
          K2 = J1
          K1 = J1 - NY
          IF(K1.LT.1)K1 = 1
          IF(NOC(K1).EQ.0)GO TO 300
          IF(K2.GT.NSP2)K2 = NSP2
          IF(NOC(K2).EQ.0)GO TO 300
          A(K1, K1) = A(K1, K1) + 1.
```

```
            A(K2, K1) = A(K2, K1) − 1.
            A(K2, K2) = A(K2, K2) + 1.
            A(K1, K2) = A(K1, K2) − 1.
300         CONTINUE
          IF(JTER.LT.LPRINT)GO TO 30
          JTER = 0
          IF(KL.EQ.1)GO TO 30
          SIGMA = (A(1, 1)-RINIT)/ITER
          WRITE(6, 2000)NY, NZ, ITER, A(1, 1), SIGMA
2000      FORMAT(2I5, I10, F17.7, F16.10)
30        CONTINUE
        RINIT = A(1, 1)
20      CONTINUE
        STOP
        END
```

This version of the program calculates the conductivity of a bar of size $N \times N \times L$ (see Fig. 2), in the case of site percolation. The algorithm and the formulas are exactly the same as in the two-dimensional case which was described in Section 2. In particular all the remarks made in Section 2 remain valid.

The only differences are as follows:

(a) The number M of the sites in the section is $M = N^2 + 2$. They are numbered as in Fig. 2.

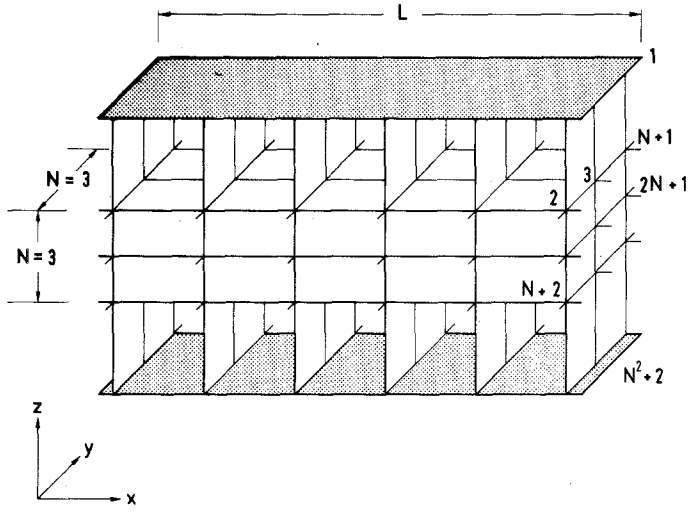

Fig. 2. A bar of size $N \times N \times L$. The sites of the section as numbered as in the program.

(b) We have to use formula (2) for all the bonds in the x direction and formula (3) for all the bonds in the y and z directions.

The parameters of this version are (i) the concentration PRO of occupied sites (here PRO = 0.3117), (ii) the width N of the bar (here $N = 14$), and (iii) the length L of the bar (here $L = 200$).

The size of the matrix A and the vectors V and NOC must be larger or equal to $N^2 + 2$ (here 198) where N is the bar width.

To eliminate the boundary effect due to the beginning of the bar, we start our calculation after a length LINIT = $L/10$. We print 50 intermediate results for the conductivity corresponding to bar lengths equal to $L/50, 2L/50, 3L/50, \ldots, 50L/50$.

The random number generator is RANF, which gives random numbers equally distributed between 0 and 1.

On the vector computers Cray 1 or Cyber 205, the innermost loop 130 is vectorized by the automatic vectorizer included in the FORTRAN compiler, leading for both machines to a typical increase in speed over a CDC Cyber 76 by a factor of three at the percolation threshold.

4. OUTER-LOOP VECTORIZATION FOR THE CDC CYBER 205

Definitely a factor of 3 increase in speed obtained by use of the automatic vectorizer is not satisfactory. The reason for this small increase is obvious: within loop 120, too frequently loop 130 is jumped over, i.e., the vectorized inner loop is not dominant in execution time over the nonvectorized outer loop 120. In the case of the CDC Cyber 205 vector computer it is possible to use the advanced features of this machine to vectorize (by hand, these features will not be employed by the automatic vectorizer) the outer loop. To our knowledge this vectorization is much more difficult for the Cray 1, because of the absence of bit vectors and the compress instruction.

The vectorized version of the loop 120 given below first collects row K3 of matrix A which is stored in nonconsecutive memory locations into a contiguous vector, $D1$, by means of the GATHER—vectorinstruction. The expression $X(I;N)$ denotes a vector of N elements, the first element being $X(I)$. In the next step the elements of this vector $D1$ are tested for their magnitude. It is important that this be done by means of forming the square and not using the absolute value function. The multiply operation is linked (chained) with the compare vector instruction by the compiler. This is not possible for the absolute value instruction in connection with the compare instruction. Therefore, this statement will be executed at an asymptotic rate for long vectors of 200 MFLOPS (Million Floating Point Operations per Second) on our 2-pipe machine. The result of the compare

is a bit vector, i.e., a vector of single bits being one where the magnitude of the corresponding $D1$ element is sufficiently large to be operated with. In the next line all elements of the vector $D1$ are multiplied with the constant D. In spite of involving more arithmetic operations than the scalar version this is much faster because of the vectorization. In the next statement the number of 1 bits in the bit vector is counted, another vector operation on the CDC Cyber 205 operating at an asymptotic rate of 800 MBIT/sec. This is the number of times loop 120 actually should be executed and in general is much smaller than the maximum, NSP1. In the next statement the vector of integers INTS holding the numbers $1, 2, \ldots,$ NSP1 is compressed on the bit vector IFNONZ. INTS is constructed once per program execution. The compress operation forms a vector INDEX holding the indices of the columns for which actually computations should take place, i.e., holding those values of $K1$ for which $A(K3, K1)$ is sufficiently large in magnitude. The remaining loop 120 now is improved above the original loop 120 in two respects: (1) it runs only over the nontrivial cases, (2) it contains only the absolute minimum scalar code required to set up the trivially vectorized 130 loop.

One may use the very same techniques to vectorize the essential parts of 100 and 300 loops; the latter may in fact be vectorized completely. Also the random number generation may be vectorized using the method published previously by two of the authors.[14]

However, doing this results only in a speed increase of 10% which does not warrant a lengthy discussion here. The sparse vector instructions of the CDC Cyber 205 are not useful for the present problem since their purpose is to save memory space and not execution time. Memory space is not the crucial problem in the present study.

```
      COMMON D1(198), INTS(198), INDEX(198)
      BIT IFNONZ(198)

      DO 1 I = 1, 198
1     INTS(I) = I

      D1(1; NSP1) = Q8VGATHP(A(K3, 1; 1), 198, NSP1; D1(1; NSP1))
      IFNONZ = D1(1; NSP1) * D1(1; NSP1) .GT. 1.0E - 16
      D1(1; NSP1) = D1(1; NSP1) * D
      NLOOP = Q8SCNT(IFNONZ(1; NSP1))
      INDEXD = Q8VCMPRS(INTS(1; NSP1), IFNONZ(1; NSP1);
     $ INDEX(1; NSP1))
      DO 120 KK1 = 1, NLOOP
      K1 = INDEX(KK1)
120   A(1, K1; NSP1) = A(1, K1; NSP1) - V(1; NSP1) * D1(K1)
```

5. PERFORMANCES

The two following curves give an idea of the program performances.

The first curve (Fig. 3) gives the execution time as a function of the bar width N, at the percolation threshold ($p = 0.3117$). One sees on this log–log plot that the time increases like N^4. One sees that there is only a factor 3 between the speed of the program on a Cray 1 and on a CDC Cyber 76, if vectorized by the automatic vectorizer. Using the advanced features of the Cyber 205 an additional factor 4 is achieved, i.e., one order of magnitude faster than the Cyber 76. For larger system sizes the speed advantage would be even larger.

The second curve (Fig. 4) gives a comparison of the execution times for a bar of width 10 as a function of the concentration p of the occupied sites. The difference of speed increases with p. One can reach a factor near 9 for p close to 1 with automatic vectorization and a factor 15 with the hand-made vectorization of Section 4 for the small system studied here. The fact that the times and the speed ratio depend on p can easily be understood. For small p, most of the bonds are missing and the simplification discussed in remark (d) of Section 2 occurs very often. On the

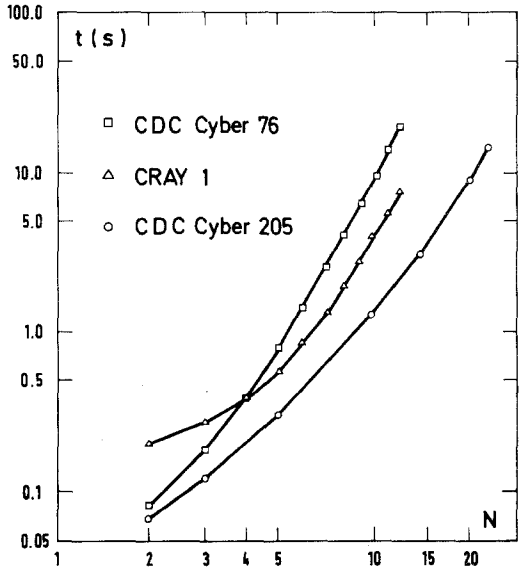

Fig. 3. The execution time as a function of the width N at the percolation threshold $p = 0.3117$ (log–log plot). The length $L = 1000$. The time increases roughly as N^4 on the CDC Cyber 76 as well as on the Cray 1 and Cyber 205. For the Cyber 205 the hand-optimized performance is given. Otherwise the performance is similar to that of the Cray 1.

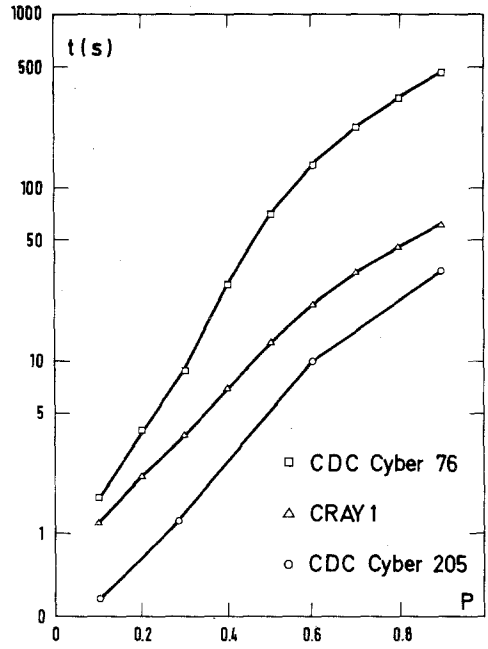

Fig. 4. The execution time as a function of the concentration of occupied sites for a bar of size $10 \times 10 \times 1000$. The ratio of speeds increases with p.

contrary, for p close to 1, most of the time is spent in using formula (2) which is vectorized on the Cray 1 or Cyber 205 already automatically. In the hand-optimized Cyber 205 version, the speed-up factor is largely independent of p and limited only by the small vector length. This again is obvious since the formerly scalar 120 loop, the execution of which hampers the small-p performance is now also vectorized.

Let us just conclude by saying that we have used this program up to a bar width $N = 24$ and a length $L = 130000$ at $p_c = 0.3117$, confirming the trends observed in Ref. 12.

ACKNOWLEDGMENTS

It is a pleasure to thank H. J. Herrmann for discussions and for a critical reading of the manuscript.

REFERENCES

1. S. Kirkpatrick, *Rev. Mod. Phys.* **45**:570 (1973).
2. I. Webman, J. Jortner, and M. H. Cohen, *Phys. Rev. B* **11**:2885 (1975).

3. J. P. Straley, *Phys. Rev. B* **15**:5733 (1977).
4. R. Blanc, C. D. Mitescu, and G. Thevenot, *J. Phys. (Paris)* **41**:387 (1980).
5. C. D. Mitescu, M. Allain, E. Guyon, and J. Clerc, *J. Phys. A* **15**:2523 (1982).
6. P. S. Li and W. Strieder, *J. Phys. C* **15**:6591 (1982); *J. Phys. C* **15**:L1235 (1982).
7. C. J. Lobb and D. J. Frank, *J. Phys. C* **12**:L827 (1979).
8. R. Fogelholm, *J. Phys. C* **13**:L571 (1980).
9. A. K. Sarychev and A. P. Vinogradoff, *J. Phys. C* **14**:L487 (1981).
10. M. Sahimi, B. Hughes, L. E. Scriven, and H. T. Davis, *J. Phys. C* **16**:L521 (1983).
11. B. Derrida and J. Vannimenus, *J. Phys. A* **15**:L557 (1982).
12. B. Derrida, D. Stauffer, H. J. Herrmann, and J. Vannimenus, *J. Phys. Lett. (Paris)* **44**:L701 (1983).
13. J. Hoshen and R. Kopelman, *Phys. Rev. B* **14**:3428 (1976).
14. R. B. Pandey, D. Stauffer, A. Margolina, and J. G. Zabolitsky, *J. Stat. Phys.*, to be published.