Journal of Statistical Physics, Vol. 34, Nos. 3/4, 1984

Diffusion on Random Systems above, below, and at Their Percolation Threshold in Two and Three Dimensions

R. B. Pandey,[1,2] D. Stauffer,[1] A. Margolina,[3] and J. G. Zabolitzky[1]

Received September 22, 1983

A detailed Monte Carlo study is presented for classical diffusion (random walks) on random $L * L$ triangular and $L * L * L$ simple cubic lattices, with L up to 4096 and 256, respectively. The speed of a Cyber 205 vector computer is found to be about one order of magnitude larger than that of a usual CDC Cyber 76 computer. To reach the asymptotic scaling regime, walks with up to 10 million steps were simulated, with about 10^{11} steps in total for $L = 256$ at the percolation threshold. We review and extend the dynamical scaling description for the distance traveled as function of time, the diffusivity above the threshold, and the cluster radius below. Earlier discrepancies between scaling theory and computer experiment are shown to be due to insufficient Monte Carlo data. The conductivity exponent μ is found to be 2.0 ± 0.2 in three and 1.28 ± 0.02 in two dimensions. Our data in three dimensions follow well the finite-size scaling theory. Below the threshold, the approach of the distance traveled to its asymptotic value is consistent with theoretical speculations and an exponent $2/5$ independent of dimensionality. The correction-to-scaling exponent at p_c seems to be larger in two than in three dimensions.

KEY WORDS: Random walk; percolation; conductivity; scaling; fractals; Monte Carlo; FORTRAN programs; supercomputer.

1. INTRODUCTION

The "ant in the labyrinth" of de Gennes[1] has become a conventional term to discuss the problem of classical diffusion in random percolating systems.[2–8] Here one studies the random walk motion of particles on the

[1] Institute of Theoretical Physics, Cologne University, 5000 Köln 41, West Germany.

[2] Now at the University of Cambridge, Department of Physics, Cavendish Laboratory, Madingley Road, Cambridge, CB3 0HE, U.K.

[3] Now at Department of Chemical Engineering, Princeton University, Princeton, New Jersey 08544, USA.

random clusters formed by connecting the neighboring open sites (in sit percolation) or allowed bonds (in bond percolation), which are distribute randomly, with probability p, on a lattice. While some workers restrict th motion of an ant only on the "infinite" percolating clusters for p above p_c others allow it to execute random walks on any cluster chosen randomly Here, we follow the latter "liberal" approach. The range of motion i limited below p_c where all clusters are finite; the ants may travel to infinit only on the infinite cluster which exists for $p \geqslant p_c$ along with finite clusters

More quantitatively, if the ant starts for $p < p_c$ from an arbitrar origin, the mean square displacement $R^2 = \langle r^2(t) \rangle$ after a long time approaches

$$R^2 = R_\infty^2 - A \exp\left[-(t/T)^w\right] + \cdots \quad (1$$

where A is some p-dependent constant, R_∞ is the saturation value for I and is some average cluster radius diverging at the percolation threshol with a critical exponent m. Also the characteristic time T diverges at p_c. Fo $p > p_c$, R grows with time asymptotically according to a diffusion law:

$$dR^2/dt = \text{const} + B \exp\left[-(t/T')^{w'}\right] + \cdots \quad (2$$

Right at $p = p_c$, the asymptotic behavior of $R(t)$ is presumed to follow a anomalous diffusion law(2)

$$R \propto t^k + \cdots \quad (3$$

Although the direct study of diffusion by random walk motion on regular lattice is a well-known problem, its connection to the conductivit of a random percolating system, particularly at p_c, has been made onl recently. The scaling description of fractals(3) has added greatly to th interest in studying diffusion on random networks.(4,5) Gefen *et al.*(4 generally derived scaling relations among the exponents describing conduc tivity and percolative quantities. This type of scaling approach is th percolation analog of dynamical scaling, as developed for example for spi diffusion in magnets near the Curie points.(6) Mitescu and Roussenq hav recently studied the diffusion on two- and three-dimensional lattices(2) they raise serious objections regarding these relations among exponents. I their Monte Carlo simulations, where they average over all cluster sizes, th critical exponent m for R was found in three dimensions to be about 1.65 i contradiction to the scaling prediction $2\nu - \beta = 1.3$. Second, they foun the diffusivity D above p_c to vanish with an exponent 1.7 whereas it i supposed to vanish as the conductivity with an exponent near 2.(7,8) Third their best estimate for the critical exponent k at $p = p_c$ was about 0.25 a opposed to the theoretical prediction $k = 0.2$. Some of these discrepancie also appeared in the different type of Monte Carlo simulation by Be

Avraham and Havlin[9] (see also Kutner and Kehr[4]). Thus there was the danger that the whole scaling theory of percolation clusters[10] had to be reexamined in view of these independent contradictions with Monte Carlo experiment.

To shed more light on these discrepancies between theory and computer experiment, we have performed a detailed simulation for diffusion on the simple cubic lattice, and a less detailed one on the triangular lattice. We used much longer times t (10 million steps) than before, lattices much larger (17 million sites) than those employed by Mitescu and Roussenq, and a better computer (CDC Cyber 76 scalar computer at Cologne and CDC Cyber 205 vector computer at Bochum). Also, our FORTRAN program worked more efficiently, taking an execution time of about 2 μs per step in case of the scalar machine and about 0.3 μs per step in case of the vector machine. The computational aspects in particular of vector computers were one reason why this problem was chosen by us. We will see later that this effort paid off by resolving some of the discrepancies and confirming much better the scaling theory of percolative diffusion.

In the following section we describe briefly the scaling description to facilitate the reader's understanding. Section 3 describes the computer programs used, and Section 4 our results, which are summarized in Section 5. A short note using some of our results at the three-dimensional percolation threshold was published before[11]; its main conclusion was confirmed by Havlin and Ben Avraham.[12]

2. SCALING THEORY

In this section we relate the various power laws and exponents mentioned in the previous section by the scaling hypothesis; mostly we review known results.[2,4] The random walks on randomly grown clusters involve two independent random processes: first, the fluctuation in displacement R with respect to time, and second the spatial fluctuations of the occupied sites. Thus we have to average over many walks on the same lattice, and then also over many lattices. This second average is not needed in diffusion on deterministic fractals and similar structures.[13] For our case an ant (unbiased random walker) starts its random motion from a randomly selected occupied site (local orgin). From then on its motion is restricted to the cluster to which the local origin belongs; if it happened to be an isolated site the ant never moves. Many ants walk on each lattice. If a local origin belongs to a cluster containing s sites, then the rms average distance from the origin will approach the radius of gyration R of that cluster (apart from a constant factor) since all sites are visited, for long times, with equal probability.[14] The probability that the local origin belongs to such an s

cluster is $n_s s$, where n_s is the average number, per lattice site, of s cluster: Thus the average displacement R involves a sum over all cluster sizes:

$$R^2 = \sum R_s^2 n_s s \qquad (t \to \infty, p < p_c) \qquad (4$$

The usual scaling assumption for the cluster numbers is[10]

$$n_s = s^{-\tau} f[(p-p_c)s^\sigma] + \cdots \qquad (5$$

where $\tau = 2 + 1/\delta$, $\sigma = 1/\beta\delta$, and β and δ are the usual critical exponent for percolation. Analogously one may postulate for the rms average radiu of gyration, R_s, of an s cluster

$$R_s = \xi g[(p-p_c)s^\sigma] \qquad (6$$

where $\xi \propto |p - p_c|^{-\nu}$ is the correlation or connectivity length.

Replacing the sum in Eq. (4) by an integral from zero to infinity an using $\beta = (\tau - 2)/\sigma$ one finds

$$R^2 \propto (p_c - p)^{-m}, \qquad m = 2\nu - \beta \qquad (7$$

for the average displacement below p_c after very long times. Numericall Mitescu and Roussenq[2] found $m = 2\nu - \beta/3$ to be a better approxima tion. This deviation from theory thus puts into jeopardy not only dynamica scaling but also the above static scaling assumptions, Eqs. (5), (6), an requires, if confirmed, a reconsideration of the whole scaling theory c percolation clusters.[10]

Equation (7) is only a special case. More generally one can make th dynamical scaling assumption[4,9]

$$R = t^k H(t/\xi^z) \qquad (8a$$

or equivalently,

$$R = t^k \tilde{H}[(p-p_c)t^{1/\nu z}] \qquad (8b$$

For large negative arguments x, this scaling function $\tilde{H}(x)$ varies a $(-x)^{-k\nu z}$ in order to give the time-independent result of Eq. (7); thu $zk = m/2\nu = 1 - \beta/2\nu$. For large positive arguments, on the other hanc the diffusion law, Eq. (2) has to be recovered, requiring $\tilde{H}(x)$ to vary fo large positive arguments as $x^{(1/2-k)\nu z}$. Thus for long times above p_c w have

$$R^2 \propto \xi^{z(2k-1)} t \propto (p-p_c)^{\nu z(1-2k)} t \qquad (9$$

On the other hand, Einstein's relation between mobility and diffusivit requires the latter to vary as $(p-p)^\mu$, where μ (often also called t) is th critical exponent for the conductivity of random resistor networks.[10] Thu $\mu = \nu z(1-2k) = z\nu - 2\nu + \beta$, using the above $zk = 1 - \beta/2\nu$. We hav therefore rederived[4]

$$z = 2 + (\mu - \beta)/\nu \tag{10}$$

and

$$2k = (2\nu - \beta)/(2\nu + \mu - \beta) \tag{11}$$

Right at $p = p_c$, a simple power law, Eq. (3), relates R and t, and therefore the scaling function $\tilde{H}(x)$ approaches a finite value for small arguments. Since generally for walks on a lattice a fractal dimension can be defined through[3]

$$\text{radius} \propto (\text{time})^{D'} \tag{12}$$

we may call $1/k$ the fractal dimension D' of this random walk on a random lattice at its percolation threshold. Above p_c, this fractal dimension is 2, as for random walks on a periodic lattice, and below p_c it is infinite.[18] Note that the fractal dimension D' of the walks at $p = p_c$ differs from the fractal dimension D_f of the clusters on which the walk occurs at $p = p_c$[3]:

$$R_s \propto s^{1/D_f}, \qquad D_f = d - \beta/\nu = d/(1 + 1/\delta) \tag{13}$$

The fractal dimensionality D is related to the fracton or spectral dimensionality $2D_f/(2 + \theta) = 2D_f/z$ which in turn can be evaluated from the number of visited different sites of the random walk.[15–18]

Of course, one can also define the above scaling assumption with the help of the diffusivity dR^2/dt instead of with R. Then[4] dR^2/dt varies as $\xi^{-\theta}$ above p_c, and as $R^{-\theta}$ at p_c, where $\theta = (\mu - \beta)/\nu$ from Eq. (10), and $z = 2 + \theta$.

For diffusion within an average cluster of s sites, the scaling assumption contains $(p - p_c)s^\sigma$ as an additional variable.[4,19] We denote the rms distance for one fixed cluster size s by r, instead of R for the average over all clusters. Equation (8a) then has a counterpart in

$$r = t^{1/z} h\left[(p - p_c)t^{1/\nu z}, (p - p_c)s^\sigma\right] \tag{14a}$$

or

$$r^2 = t\xi^{-\theta}\bar{h}\left[(p - p_c)t^{1/\nu z}, (p - p_c)s^\sigma\right] \tag{14b}$$

or

$$r = R_s F(t/s^{z/D_f}, R_s/\xi) \tag{14c}$$

In Eqs. (14a), (14b) the prefactor for the scaling function no longer is t^k but $t^{1/z}$ as required[4] to be consistent with Eqs. (7) and (10). At $p = p_c$, the two scaling functions in Eqs. (14a) and (14b) have a finite value at zero arguments.

Alexander and Orbach[5] noted that the ratio z/D_f is about 3/2 fo various dimensions. Assuming that relation, for which some arguments ca be given,[19,5] to be exact

$$z = 3D_f/2 \tag{15}$$

we not only can relate[5] the conductivity exponent μ to the other expo nents through

$$\mu/\nu = 3d/2 - 2 - \beta/2\nu \tag{16a}$$

but also have the simple relation

$$\beta k = (2 - \beta/\nu)/(d - \beta/\nu) \tag{16b}$$

which gives $k = 1/3$ exactly in two dimensions.

We now make two additional simplifying assumptions, similar t Wilke *et al.*[19]: First, for clusters much larger than the correlation length below p_c, the relation between r, R, t, and s is independent of th correlation length, i.e., of the distance from p_c. Second, for very long time below p_c, the distance r approaches its asymptotic limit R_s with a simpl exponential. The first assumption,

$$r/R_s = \bar{F}(t/s^{3/2}) \tag{17}$$

is a (questionable) "strong dynamic similarity" assumption for all "ani mals," i.e., for all clusters below p_c which are much larger than th correlation length. The second assumption

$$\log(R_s - r) \propto -t \qquad (t \to \infty) \tag{18}$$

is an alternative to the assumption of Mitescu and Roussenq,[2] who use

$$\log(R_{t=\infty} - R) \propto -t \qquad (t \to \infty) \tag{19}$$

for the average over all cluster sizes. From our assumption and the relatio $\log(n_s) \propto -s$ for very large clusters below p_c,[10] a straightforward integra tion in the limit of long times gives for the average over all clusters

$$\log(R_s - R) \propto -t^{2/5} \qquad (t \to \infty) \tag{20}$$

independent of dimensionality d. Assumption (19) gave difficulties[2] i describing the data. Wilke *et al.*[19] could not clearly confirm Eq. (18) fro Monte Carlo simulations on animals, whereas Fassnacht,[20] using prelimi nary results of our simulations, clearly preferred Eq. (20) over Eq. (19). W will also present better data on that question.

Needless to say, all these scaling theories are not valid far away fro the percolative phase transition. They are supposed to hold for large times large distances, large clusters, infinite systems, and for dimensionalities d larger than one and smaller than six.

3. METHOD

3.1. General Techniques

The basic idea behind the computer simulation of classical diffusion presented here is very simple. First, we prepare a random sample (called a lattice relaization) by randomly occupying the sites, with probability p, in a simple cubic or triangular lattice. As in simple percolation studies this is done with the numbers zero and one, or the logical variables .TRUE. and .FALSE. In this way clusters of various sizes are generated automatically, in contrast to the cluster growth algorithm employed, e.g., in Refs. 9, 12, and 21. Periodic boundary conditions usually reduce the finite-size effects. (The cluster growth method in principle corresponds to infinite lattices.)

Now one occupied site is selected randomly as local origin; from here the ant starts its random motion. One of the six nearest-neighbor sites is selected randomly, and the ant is moved to this site if it is occupied; otherwise the ant stays at its previous place. In both cases the time is increased by a unit step, whether the attempt to move was successful or not.(2,14) The process of randomly choosing a neighbor of the current ant position and of attempting to move to it is repeated again and again for a preset number of steps, the maximum time. From the calculated rms distance R as function of time t at various concentrations p we calculate the radii, diffusivities, and their critical exponents.

The main aim of our work was to check the reliability of earlier Monte Carlo simulations of percolative diffusion by making more Monte Carlo steps in larger lattices. Therefore we put particular emphasis in making the algorithm efficient, and we describe now how we did that on the CDC Cyber 76 scalar and the CDC Cyber 205 vector computers. We start with techniques common to both. All our programs were written in FORTRAN; at least those for the scalar computer should be applicable rather generally.

If we take ten million steps for each ant there is no need to know all ten million intermediate distances. We were content in calculating and printing out at most 1000 distances for 1000 different times. So in following the particle through ten million steps, we first make 10000 steps, then calculate and store its position, then make the next 10000 steps, after which a new position is calculated, etc. Therefore the innermost loop, over 10000 steps in this example, need not calculate many details which are needed in the final analysis only. We call this innermost loop

```
DO 50 II = 1, MX
```

and describe only the structure of this loop. If MX is small, like 10, the efficiency is reduced, of course. For the future we recommend(22) increasing MX, the lengths of the innermost loop, during one run by factors of 1(until it reaches one tenth of the maximum time.

The whole lattice was stored on a one-dimensional array. If on a plan(the ant sits on the rightmost site of line 10 it moves to the leftmost site o line 11, not of line 10 ("helical boundary conditions") if it is supposed t(move to the right. To calculate the distance from the local origin we trea the ant as if it had moved to the right; the boundary conditions merel: repeat the lattice and do not hinder the motion. We work with $L * L$ an($L * L * L$ sites in two and three dimensions. For example, with $L = 100$ o the triangular lattice, the left and right neighbors of site 550 are sites 54(and 551, whereas the top neighbors are 450 and 451 and the botton neighbors are 650 and 649. This index we call k. Then, instead of changin; d coordinates in d dimensions, we only have to change one index k if th(particle moves.

However, if the particle sits, for example, on site 1 and is supposed t(move up or to the left, then k tries to become negative; and no compute memory corresponds to negative index k. One could have checked at ever: step if the new k is smaller than 1 or larger than L^d; but that would hav(cost time. Thus instead we stored in addition about 20 planes (or lines $L+1, L+2, \ldots,$ which are identical to the first 20 planes $1, 2, \ldots, 20$ Similarly, as planes $0, -1, \ldots, -19$ we stored planes identical to plane $L, L-1, \ldots, L-19$ of the real lattice. Finally two planes -20 an($L+21$ are used which are all empty. Now we check for periodic boundar: conditions outside the innermost loop; e.g., if k was negative it wa increased by L. In the rare cases where during the MX steps of th(innermost loop the particle diffused to the region close to negative k, th(empty plane prevented it from diffusing across the prohibited boundary Then the top and bottom planes were treated as free boundaries whereas i general they were treated through helical boundary conditions. Therefore i the formulation of the innermost loop we no longer have to deal with th(boundary conditions and save execution time.

3.2. Scalar Computer Cyber 76

On the "normal" CDC computer Cyber 76, and similarly on othe CDC 7000 series machines with FORTRAN IV compiler, we can store th(occupation status in a logical array IS where .TRUE. means empty an(.FALSE. means occupied. The six shifts of the index k, i.e., 1, -1, L, $-L$ $L-1$, and $1-L$ for the triangular lattice, are stored in the array NBR o six elements. The array IDIST, also with an index from 1 to 6, stores ho: often the particle moves to the left, to the right, etc. Outside the innermos loop this array then gives the squared distance through (IDIST(1) -

IDIST(2)) * * 2 + (IDIST(3) − IDIST(4)) * * 2 + (IDIST(5) − IDIST(6)) * * 2 in the simple cubic lattice, and through [(2 * (IDIST(1) − IDIST(2)) + IDIST(3) − IDIST(4) − IDIST(5) + IDIST(6)) * * 2 + 3 * (IDIST(3) − IDIST(4) + IDIST(5) − IDIST(6)) * * 2]/4 in the triangular lattice. Our innermost loop is now quite trivial and took on the average about 1.2 μs:

```
      DO 50 II = 1, MX
      IN = 1 + IFIX(6. * RANF(0))
      KNEW = K + NBR(IN)
      IF(IS(KNEW)) GO TO 50
      IDIST(IN) = IDIST(IN) + 1
      K = KNEW
50    CONTINUE
```

Here RANF is the standard random number generator, giving real numbers homogeneously distributed between zero and unity.

However, the above program, while fast is very inefficient in its memory use since a full 60-bit computer word is used to store the one-bit information on whether or not the site is occupied. Computing time is nearly doubled to 2.2 μs per step, but memory saved by a factor 60, if we store a different site in each of the 60 bits of the word. Thus the above program was used for small lattices only whereas for L up to 180 in three dimensions each word IS(J) stored 60 sites. (We also employed then the slower auxiliary memory Level 2.) The innermost loop now reads as above, only with line 4 replaced by

```
J = KNEW/60
INTS = SHIFT(IS(J), KNEW − 60 * J)
IF(INTL) GO TO 50
```

Here SHIFT(word, n) is the usual function (available also on many IBM FORTRAN H Extended compilers through option XL) which shifts circularly the word by n bits to the left. INTL is a logical variable (true if the leftmost bit is one and false if it is zero) which at the beginning through EQUIVALENCE (INTS, INTL) is put onto the same storage location as INTS.

In both versions we checked at the beginning of each walk if the local origin is an isolated occupied site. Then no simulation was needed since that walk contributes zero to the average distance.

Following Havlin and co-workers, we also treated intermediate positions of the ant as starting points of a new, shorter walk. For that purpose we stored outside the innermost loop after every MX steps not only the total squared distance traveled so far, but also the d coordinates of the ant.

From, say, 1000 such positions we then at the end calculated 100 differen distances traveled in $10*\mathrm{MX}$ steps, 50 distances traveled in $20*\mathrm{MX}$ steps etc. We found these "Havlin averages" to be smoother than the singl results where all distances are measured from the true origin of the walk

3.3. Vector Computer Cyber 205

A "supercomputer" like the CDC Cyber 205 solves problems in a assembly-line fashion: A thousand identical operations executed on thousand different sets of data are done more efficiently on the vecto computer than on a normal or "scalar" machine. It is necessary, however that data needed consecutively are stored in consecutive locations of th memory and are all treated in the same way independently of each other In our problem we usually treated $N = 512$ ants in this assembly-lin fashion, since for each ant the same operations are executed: Selec randomly a neighbor, check if neighbor is occupied, move ant if yes Unfortunately, the number of statements now is much larger.

CDC does not provide an efficient vector random number generator Their VRANF(RANFD, N) which puts N random numbers into any arra RANFD, is in fact slower than successive calls to the usual scalar RANF Since a large fraction of computer time is spent generating random num bers we devised a specialized random number generator which produce random arrays of constant length N at a rate of about 20 ns per number.[23 Furthermore, the sequence of random numbers generated is exactly th same as with the usual RANF. For this purpose we initialize the resultin random vector RANFD by the standard scalar RANF code, and simulta neously we compute the Nth power of the multiplier, modulo 2^{47}, b lower-word multiplications:

```
      DATA XSEED/X'000054F4A38933BD'/, MPOWER/1/,
    1 IXMULT/84000335758957/, IEXPON/65489/
      DO 3 I = 1, N
      CALL Q8MPYL(XSEED, IXMULT, XSEED)
      CALL Q8MPYL(MPOWER, IXMULT, MPOWER)
  3   SAVE(I) = XSEED
```

After this initialization one can produce many vectors of N random rea numbers between zero and unity by calling two vector instructions for eac such vector:

```
CALL Q8MPYLV(X'08',,SAVED,,MPOWER,,SAVED)
CALL Q8PACKV(X'10',,IEXPON,,SAVED,,RANFD)
```

The first line generates random integers SAVED between 1 and 2^{47}; the second line transforms them into real numbers RANFD between zero and unity by inserting the properly normalizing exponent field and is not needed if only integers or random bits are wanted. (Here and later, variables ending with *D* are abbreviations, called descriptors, for vectors of length *N* like SAVE(1; N) and cause the statement to be executed for all *N* array elements.) Omission of the pack-statement leads to twice the speed (10 ns per number). All the above calls do not actually call subroutines but are translated directly into machine instructions by the compiler. This random number generator can be used only to generate sequences of *N* random numbers where *N* is fixed throughout the program. If not many different values of *N* occur, a number of independent such generators may be used to avoid these difficulties.

In order to vectorize KNEW = K + NBR(IN) in the above scalar program, we first have to put the *N* numbers NBR selected randomly for the *N* ants into consecutive memory locations, which is achieved by a Q8VGATHR operation. Then we have to find out if the new possible positions KNEW are occupied or not. KNEW is a vector of indices for the bit string IS containing our lattice. What we really would like to do is to gather bits with this index vector KNEW as we did for the word index vector KNEW from NBR by Q8VGATHR. Unfortunately, this bit operation is not implemented on the Cyber 205 which we regard as the most serious deficiency of its instruction set (bit gather would also be useful in the case of sparse matrix algebra).

Thus we now implement this bit gathering by separate instructions, similar to the above scalar program: Two shift instructions and the subtraction generate from the bit index KNEW a word index JNEW = KNEW/64, and a bit number JSHFT within the word IS(JNEW). We then gather these words, move by a shift operation the relevant bit to the rightmost bit position, and mask all preceding bits out by a logical AND. (A Q8LINKV statement declares this SHIFT and AND as a linked triadic operation which will execute at twice the normal speed for separate "unlinked" operations.) A comparison with unity compresses the one-bit-per-word bit string into a dense bit string OCCUP which may subsequently be used as a control vector to reject or accept a move. The displacement made by one move is gathered into IXYZN, which is added to the current position IXYZ if OCCUP is true. Both IXYZ and IXYZN store *x*, *y*, and *z* coordinates as 15-bit fields in one word to save time. (Thus the maximum allowed displacement within MX steps is $2^{14} - 1$.) The full innermost loop now looks as follows:

```
      BIT OCCUPD
      DATA IONE/1/,ISIXR/-6/,ISIXL/6/
       :
       :
      DO 50 II = 1,MX
      CALL Q8MPYLV(X'08',,SAVED,,MPOWER,,SAVED)
      CALL Q8PACKV(X'10',,IEXPON,,SAVED,,RANFD)
      IND = 1 + VIFIX(6.*RANFD; IND)
      KNEWD = KD + Q8VGATHR(NBRD,IND; KNEWD)
C
C     The next 8 lines simulate the nonexisting command:
C     OCCUPD = Q8VBITGATHR(ISD,KNEWD; OCCUPD)
      CALL Q8SHIFTV(X'08',,KNEWD,,ISIXR,,JNEWD)
      CALL Q8SHIFTV(X'08',,JNEWD,,ISIXL,,KMODD)
      JSHFD = KNEWD - KMODD
      ISGAD = Q8VGATHR(ISD,JNEWD; ISGAD)
      CALL Q8LINKV(X'10')
      CALL Q8SHIFTV(X'00',,ISGAD,,JSHFD,,ISGAD)
      CALL Q8ANDV(X'09',,ISGAD,,IONE,,ISGAD)
      OCCUPD = ISGAD.EQ.1
C
      IXYZND = Q8VGATHR(NXD,IND; IXYZND)
      WHERE (OCCUPD)
      IXYZD = IXYZD + IXYZND
      KD = KNEWD
      END WHERE
50    CONTINUE
```

Outside this innermost loop we calculate the distances as before; to average over all ants we sum up the distance for each ant with the Q8SSUM function.

In this way we achieved a speed of about 276 ns per step. The various gather operations needed slow the vector machine down to a speed of only 8 times that of the scalar Cyber 76, below the full vector potential.

To build up the lattice of occupied and empty sites takes an appreciable fraction of execution time for short and medium times. In a scalar version it required per site about 0.7 μs on the Cyber 76 and about 1.1 μs on the Cyber 205. A vectorized version, however, reduced that latter time to 0.06 μs. We achieved that speed by incorporating the sign bit of the difference between the random number and the probability p into a special occupation vector. A vectorized logical OR put this bit into the IS words, which are then shifted by one bit in a vectorized form. After 64 such vectorized calls for random numbers, extraction of sign bits, and shifts, we have stored 64 sites into each of the IS words. For the triangular lattice at $p = p_c = 1/2$, an even shorter time of 0.037 μs was reached by simply taking the leading nontrivial bit of the random integer as the occupation bit. The innermost loop then was

```
        DATA MASK/X'0000400000000000'/   at the beginning
        ICID = 0
        DO 21 II = 1,64
        CALL Q8MPYLV(X'08',,SAVED,,MPOWER,,SAVED)
        CALL Q8ANDV(X'09',,SAVED,,MASK,,IOCCD)
        CALL Q8LINKV(X'10')
        CALL Q8SHIFTV(X'08',,ICID,,IONE,,ICID)
21      CALL Q8ORV(X'08',,IOCCD,,ICID,,ICID)
        IS(1;N) = ICID
```

Here the first call gives random integers stored in SAVED, the second call extracts the leading bit of this integer, the last two calls put this bit into ICID. ICID, SAVED, and IOCCD describe arrays of length N, where N is also the number of ants in the system. This increase in speed by a factor of 20 shows clearly the advantages of vectorization in cases where no complicated gathering operations are needed.

From this description it is quite clear that this vector computer is quite clumsy in its use if one is interested in other than floating-point computations. It would be better, if the bit-by-bit logical and shift operations would, in a vectorized form, be denoted by simple functions analogous to the Cyber 76 FORTRAN language, instead of the assembler-type statements Q8 . . . used above. Also, gather operations should be made automatically by the compiler, and the random number generator should become available in a simple efficiently vectorized form. Then programs like the above would not run faster but would at least be easier to write down, as is appropriate for higher programming languages like FORTRAN. Implementation of the bit gather operation, which here takes about half of the execution time, in microcode as a machine instruction would presumably lead to a significant increase in speed for the present program. Of course, if the gather operation did not exist at all, as is the case with other supercomputers, the above altorithm would not be anywhere as fast.

4. RESULTS

4.1. General

Here we present the results of our computer simulations in $L*L*L$ simple cubic and $L*L$ triangular lattices. We take $p = 0.3117$ and 0.5 in these two cases, and $\beta = 5/36$, $\nu = 4/3$ in two dimensions, while $\beta/\nu = 0.48$ and $\nu = 0.88$ in three dimensions.[24] On the scalar computer we went up to $L = 180$ in the cubic lattice and made only test runs in the triangular case. Detailed statistics at $p = 0.3117$ were given in Ref. 11; and a similar effort was made for p above and below p_c. On the vector computer at $p = 0.3117$ we went up to $L = 256$ where we let 500 to 1000 ants run on each lattice, and simulated 4 lattices up to 10 million steps, 60 up to one million, 100 up to 100000, and 3000 lattices up to 10000 steps. This series of runs consisted of a total of nearly 10^{11} steps, each requiring one random number, check if neighbor is occupied, and possibly a move to this neighbor. As a result we get the rms average displacement R as a function of time t, where R is measured in units of the lattice spacing, and t in jump attempts. For the triangular lattice at $p = 0.5$, we used L up to $L = 4096$, with 512 ants on each lattice, and used 9 lattices up to ten million steps, 50 up to one million, and about 1000 up to 10000, and also many runs for $L = 1024$ and $L = 2816$, all on the vector computer. (Most of the data away from p_c were produced on the scalar machine.)

4.2. At Three-Dimensional Threshold

To study the diffusion at the percolation threshold in the simple cubic lattice, we worked on lattices of sizes $L = 30, 60, \ldots, 180$ on the scalar computer and used $L = 256$ on the vector machine. Figure 1 shows the distance R traveled by the ant as a function of time t, for $L = 30$, 180, and 256. From these data we determined the effective exponent $k = d(\log R)/d(\log t)$ as a function of time; the results from the scalar computer were already given in Ref. 11; with $k = 0.20 + 0.01$ estimated there. Figure 2a shows k as a function of $1/R$ for $L = 256$. The upturn for the longest times was observed already at $L = 60$ for shorter times[11] and presumably is a finite-size effect. Ignoring it we extrapolate visually to infinite times and distances to get k near 0.20. More accurately, Fig. 2b gives more reliable data at intermediate times, from which we estimate again

$$k = 0.20 \pm 0.01 \tag{21a}$$

The increase of the effective exponent k for very long times is a size effect which is easy to explain qualitatively. In a finite lattice at p_c, a

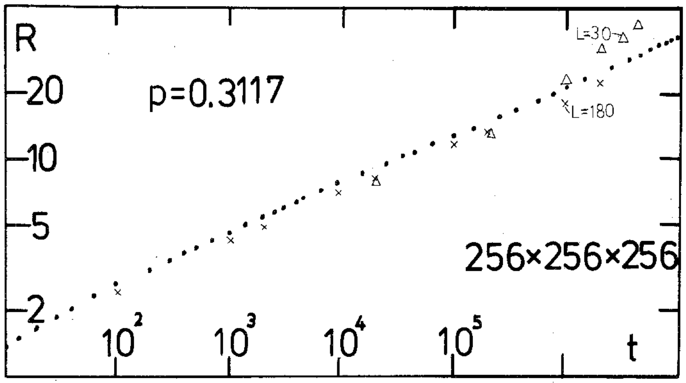
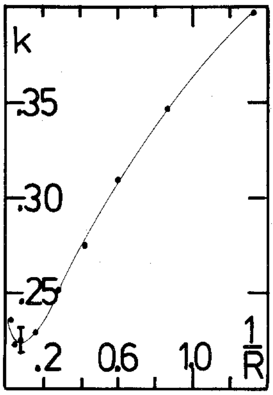


Fig. 1. Average rms displacement R versus time steps t for $L*L*L$ lattices with $L = 30$ ($\triangle$), 180 ($\times$), 256 ($\bullet$) at the percolation threshold $p = 0.3117$.



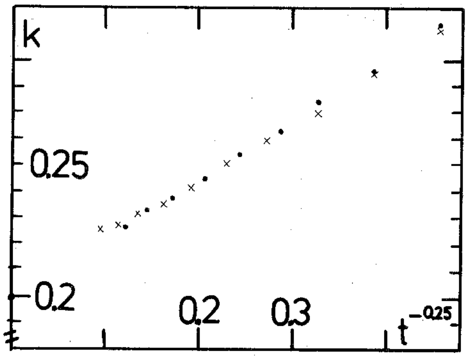


Fig. 2. The exponent k versus (a) $1/R$ and (b) $t^{-0.25}$ from the data for the sample 256^3 presented in Fig. 1 at $p = 0.3117$, to extrapolate the result to the asymptotic limit $R \to \infty$ and to estimate the correction terms to the asymptotic power law ($R \propto t^k$), respectively.

fraction of lattices is percolating, and the rest is not. Thus for a fraction of our lattice realizations, the ant can travel to infinity, whereas for the other realizations it is confined to finite clusters. Once the ant has moved over distances much larger than L, the random lattice, with helical boundary conditions, can be regarded as a periodic structure on which ordinary diffusion takes place. Thus for these percolating lattices, the exponent k must approach $1/2$ for sufficiently long times. In our rms average for the distance, at these very long times the large distances in the percolating samples dominate over the finite displacements of the confined ants. Thus also our average displacement R increases asymptotically with an exponent $k = 1/2$ as for normal diffusion.

The prediction from the Alexander–Orbach relation, Eq. (16b), is $k = 0.201 + 0.001$, in good agreement with our simulation. If we do not assume Eq. (16b) to be valid then we can derive $\mu/\nu = 2.26 + 0.2$ from Eqs. (10) and (21a). This result is in accord with and nearly as accurate as the best direct estimate(8) known to us (2.2 + 0.1), which was based on a Monte Carlo transfer matrix calculation involving lattice sizes up to $24 * 24 * L$. Our Fig. 2b then suggests a leading correction of the form

$$R = t^{0.2}(1.4 - 0.9/t^{0.20 \pm 0.02}) \tag{21b}$$

It seems quite possible that the leading corrrection term varies as $1/R$, without an additional exponent (Fig. 2a).

Earlier simulations(2,4,9) gave k near 0.25. We argue that the discrepancy between the scaling theory and these earlier Monte Carlo data is due to the fact that there the exponent k was calculated from moderately short times of the order of a thousand steps. As seen clearly in Fig. 2, these times are far from the asymptotic limit and indeed also in our work give an effective exponent near 0.25. Reference 12 gives an independent confirmation of this conclusion.

The times needed to get into "equilibrium" are unusually large because the correction exponent is so small. Therefore, to have the R increase by one order of magnitude, we have to increase the time t by five orders. Viewed from Fig. 2a, where we use R instead of t as the variable, our walks are not particularly large and on average cover at most about 35 lattice constants. So we have seen another example that a strong increase in Monte Carlo effort does not strongly reduce the error bars; it merely shows that earlier error bars were overly optimistic since they neglected systematic deviations from corrections to scaling.

4.3. Above the Three-Dimensional Threshold

For $p > p_c$ we always have an infinite network along with finite clusters. The larger $p - p_c$ is, the larger is the fraction of sites belonging to this infinite cluster. The average asymptotic random walk behavior is governed by Eq. (2), i.e., $R^2 \propto t$. The factor of proportionality is essentially the diffusivity and varies as $(p - p_c)^\mu$, since our local origins are chosen randomly anywhere in the system on an occupied site. Figure 3 shows how R^2 varies with t for various p. The asymptotic slope of these curves gives the diffusivity, as collected in Table I. Fitting these data to a simple power law in $p - p_c$ we find a critical exponent μ increasing slightly with system size L: $\mu = 1.71$, 1.80, and 1.85 for $L = 30$, 60, and 180. The statistics for $L = 180$ are as good as for the same size at p_c. However, one should not be optimistic on the errors in determining the slopes of R^2 versus t, particularly close to p. Also, we have no reliable data very close to p since there the

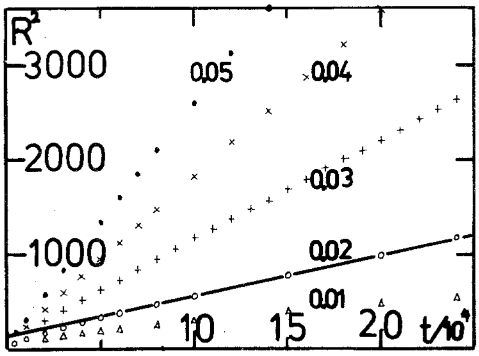


Fig. 3. Average mean square displacement R^2 versus t. The numbers on the data sets give $p - p_c$. Each sample had 20 lattice realizations with 200 ants each.

Table I. The Values of Diffusivity $(0.5dR^2/dt$ **for** $t \to \infty)$ **Calculated from the Data for** R **for** $p > p_c$ **for** $L * L * L$ **Samples with** $L = 30, 60$ **and** 180.[a]

	Diffusion constant dR^2/dt		
Δp	30^3	60^3	180^3
0.01		0.0015	0.0013
0.02	0.0056	0.0050	0.0041
0.03	0.0120		0.0105
0.04	0.0200	0.0160	0.0173
0.05	0.0270	0.0250	0.0253
0.06	0.0390	0.0370	0.0355
0.07	0.0480	0.0520	0.0456
Exponent:	1.71	1.80	1.85

[a]The exponent μ in (diffusivity) $\propto (p - p_c)^\mu$, was then evaluated from these data of diffusivity in each of the samples.

normal diffusion behavior sets in too late to be observable. The exponent just quoted is simply taken from the slope of a log–log plot, which neglects corrections to scaling. Therefore it is not surprising that this estimate is about 10% too low compared with our more accurate determination, $\mu = 2.0 + 0.2$, in the preceding section. (The data for $L = 30$ and 60 are rather poor but accurate enough to provide an idea of the trend with system size.)

4.4. Below the Three-Dimensional Threshold

For $p < p_c$ all clusters are finite. Thus if we allow the ants to execute random walk motions for sufficiently long times, then the displacement R will saturate to a value connected with an average cluster radius, Eqs. (1) and (7). A typical growth curve is shown in Fig. 4. Fassnacht[20] recently has studied the approach of R to its saturation value and estimated the exponent w in Eq. (1) as about 0.4:

$$\log\left[R_\infty - R(t)\right] \propto -t^{0.4 \pm 0.1} \qquad (22)$$

Using somewhat better statistics, we show in Fig. 5 that indeed $w = 0.4$ fits the data better than $w = 1$. Mitescu and Roussenq[2] determined the shape of the approach to equilibrium by a formula with several free parameters but with a fixed $w = 1$ which we regard questionable. Moreover, $w = 0.4$ is in nice agreement with the speculation mentioned in Section 2, which gives $w = 2/5$ for all dimensions.

The leading term in Eq. (1), viz.

$$R_\infty^2 \propto (p_c - p)^{-m}$$

can be analyzed more reliably. Table II gives our estimates for size $L = 30$, 60, and 180. A simple power law fit gives $m = 1.2$, 1.1, and 1.2 for these three sizes. Again the statistics on the small lattices are worse than for $L = 180$. Our estimate $m \cong 1.2$ is consistent with the theoretical prediction

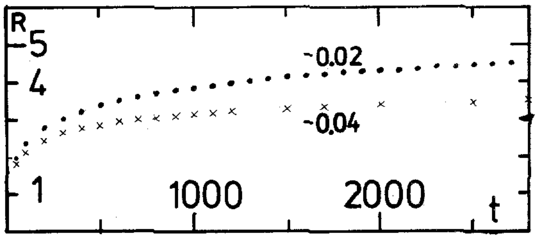


Fig. 4. Average rms displacement R versus t for the 180^3 samples, 20 realizations with 500 ants each, for the values of $p - p_c$ indicated in the figure.

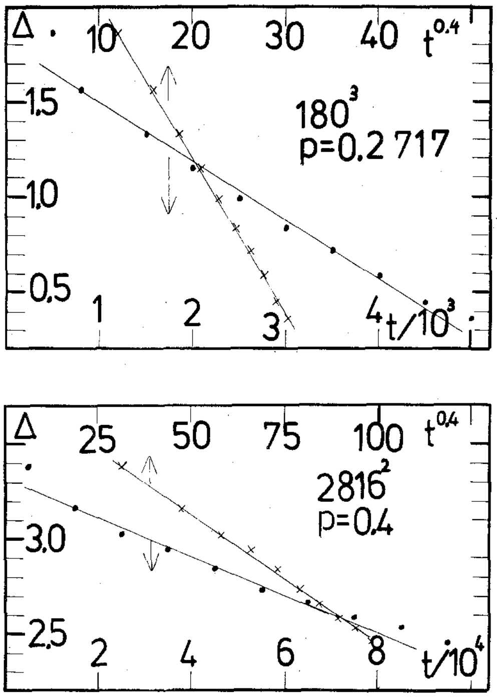


Fig. 5. Plots of $\triangle = \ln(R_\infty^2 - R^2)$ versus $t^{0.4}$ (+) and t (•): (a) Sample size 180^3, $p = 0.2717$ (50 lattices with 500 ants each), and (b) sample size 2816^2, $p = 0.4$ (6 lattices with 500 ants each).

Table II. The Saturation Values of the rms Displacement R (the Average Cluster Radii) at Various Concentrations p below Percolation Threshold for $L * L * L$ Samples with L = 30, 60, and 180.[a]

	R		
Δp	30^3	60^3	180^3
0.01	8.60	8.00	8.50
0.02	6.50	6.00	6.20
0.03	4.80	4.75	4.68
0.04	4.00	4.00	3.87
0.05	3.30	3.00	3.22
Exponent:	1.20	1.16	1.39

[a]The exponent m in $R_\infty^2 \propto (p_c - p)^{-m}$ was evaluated from these data of R in each sample.

$m = 1.34 + 0.03$ from Eq. (7), taking into account the unknown systematic errors in our analysis due to our neglect of corrections to scaling.

4.5. Finite-Size Scaling

Let us now see how our data on finite lattices fit the finite-size scaling laws widely discussed in the literature.[25] In a finite system the correlation length, which in infinite systems diverges as $(p - p_c)^{-\nu}$, has to remain finite and will approach a maximum near p_c of the order of L, the linear dimension of the system. Similarly, any other quantity diverging or vanishing with some power of $|p - p_c|$, will also stay finite at p_c in a finite system. However, if we express this other quantity not through powers of $p - p_c$ but of ξ, then this relation will still remain valid, apart from a constant factor, at p_c even in a finite system.

In our case, the diffusivity $D(p)$, which in an infinite system varies as $(p - p_c)^{\mu} \propto \xi^{-\mu/\nu}$, therefore is expected to be about $L^{-\mu/\nu}$ at p_c in a finite lattice. More generally, finite-size scaling expects

$$D(p,L) = L^{-\mu/\nu}\Phi\left[(p - p_c)L^{1/\nu}\right] \tag{23a}$$

and analogously,

$$R_{\infty}^2 = L^{m/\nu}\Psi\left[(p - p_c)L^{1/\nu}\right] \tag{23b}$$

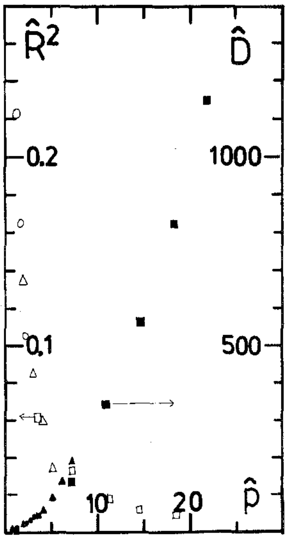


Fig. 6. Plots of $\hat{D} = (\text{diffusivity})L^{\mu/\nu}$ versus $\hat{p} = |p - p_c|L^{1/\nu}$ (full symbols) and $\hat{R}^2 = R_{\infty}^2 L^{-m/\nu}$ versus $\hat{p}$ (empty symbols) for samples $L * L * L$ with $L = 30$ (dots), 60 (triangles), 180 (squares). We used $\mu/\nu = 2.0$ and $m/\nu = 1.53$.

Figure 6 shows for $L = 30$, 60, and 180 the diffusivity and squared distance, normalized by $L^{-\mu/\nu}$ and $L^{m/\nu}$, respectively, versus the argument $(p - p)L^{1/\nu}$ of the scaling functions in Eq. (23). The agreement with finite-size scaling is reasonable; all the points for the different sample sizes lie almost on the same curves.

4.6. At the Two-Dimensional Threshold

Nearly all simulations in two dimensions were made on the vector computer with 512 ants on each lattice. We concentrated on $p = p_c$. Since $p_c = 1/2$ and also the expected $k = 1/3$ are known or hoped to be exact, it is more practical to work with the scaled distance $R/t^{1/3}$. This scaled distance should approach a constant for infinite time in infinite lattices. Figure 7 shows our data for $L = 2816$ and $L = 4096$ for the scaled distance as a function of $\log(t)$. Up to $t = 100000$ they seem to confirm this relaxation towards a plateau. However, beyond that time a decrease of the scaled distance was observed for $L = 2816$, somewhat larger than the statistical error. The same downturn occurred an order of magnitude later for $L = 4096$. In principle for very large times the displacement in finite systems should be larger than in infinite lattices due to our helical boundary conditions (see Section 4.2). The larger the lattice is, the later this upward deviation should occur. Therefore an increase of the scaled distance with time could easily be discarded as a finite-size effect. We observe the opposite trend, a decrease. Very small systems show clearly that the distance for small times is smaller, and not larger, than in large systems. It is therefore possible that the finite size of this triangular lattice decreases R for intermediate times and increases R for very long times. (For $L = 1024$

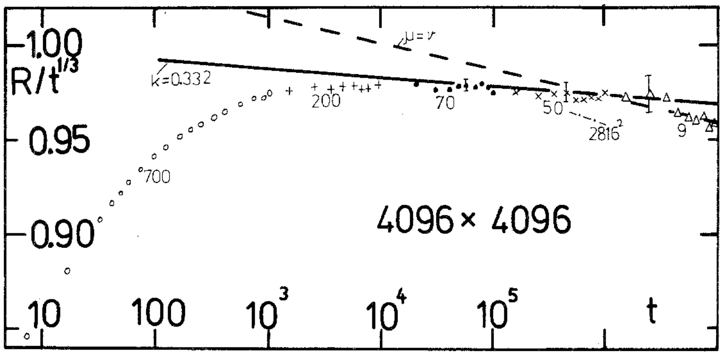


Fig. 7. $R/t^{1/3}$ versus t plots for $L * L$ lattices with $L = 4096$ and 512 ants on each lattice realization. The numbers on each data set in this semilogarithmic plot give the number of lattices. The solid line is our best fit, the dashed line corresponds to $\mu = \nu$. The dashed-dotted line indicates the finite-size downturn for $L = 2816$.

and $t > 10^5$, we did find such an increase.) A quantitative analysis was not possible since the finite-size effects in the region of interest are mixed too strongly with the finite-time effects (corrections to scaling). Thus we only conclude that the downward trend at the end of the curves should not be relied upon. (Some indications of a nonmonotonic size effect were also seen in Fig. 1 for three dimensions.)

If we took the final downturn seriously, we would find our data compatible with the hypothesis $\mu = \nu$, which corresponds to $k = 0.328$. This exponent corresponds to the strong negative slope shown in Fig. 7. But since we prefer to ignore this size-dependent downturn, we regard the line with the smaller negative slope in that figure to be a much better estimate. It corresponds to

$$k = 0.332 \pm 0.002 \tag{24}$$

and does not exclude the possibility $k = 1/3$, which is the Alexander–Orbach prediction. This estimate is exactly the same as that of Derrida and Vannimenus: $\mu = 1.28 \pm 0.02$, obtained by a transfer matrix simulation.[26] We encourage more computational effort with that method to determine μ better. (See our note added in proof.)

The approach to the plateau in Fig. 7 again gives a rough estimate of the correction exponent:

$$R = t^{1/3}(0.99 - 0.4/t^{0.45 \pm 0.05} + \cdots) \tag{25}$$

The coefficient 0.99 of the leading term in Eq. (25), as determined from $30 < t < 10^4$, seems slightly higher than the average 0.98 found from the less accurate data at much longer times. The exponent 0.45 of the correction term is about twice as large as in three dimensions. Therefore the corrections are for large times much smaller in two than in three dimensions, which makes it now easier to determine the leading term but more difficult to determine the corrections.

The cluster numbers, n_s, at $p = p_c$ in two dimensions, vary as[27] $s^{-\tau}$ $(1 - \text{const}/s^{\Omega})$ with possibly $\Omega = 1 - \sigma = 0.6$. Since times t scale as $s^{3/2}$ according to Eq. (17), this correction would correspond to a factor $(1 + \text{const}'/t^{0.4})$ in our case, compatible with Eq. (25).

Finally, the approach to equilibrium below p is again consistent with the speculation in Section 2, that $\log[R_\infty - R(t)]$ varies as $1/t^{2/5}$ independently of dimensionality. Our data were already included in Fig. 5.

5. CONCLUSION

We have studied in detail the problem of classical diffusion on random systems below, at, and above the percolation threshold by Monte Carlo studies of triangular and simple cubic lattices. Earlier discrepancies between scaling theory and computer experiment were shown to be due to too short times or too small lattices. Our results for the conductivity exponent μ, calculated by dynamical scaling from our data, are in good agreement with other recent results and with the hypothesis of Alexander and Orbach: $\mu = 2.0 \pm 0.2$ in three and $1.28 + 0.02$ in two dimensions. A correction-to-scaling exponent was estimated at $p = p_c$ to be appreciably larger in the triangular lattice than in the cubic lattice. The approach to the finite asymptotic value for the displacement below p seems to be consistent with a speculation that the corresponding exponent is $2/5$ independent of dimensionality. For future work in two dimensions it would be desirable to use larger lattices or to work with a different algorithm corresponding to infinite lattices. Work in four dimensions,(22) on pair diffusion in two and three dimensions,(28) and on biased diffusion(29) is in progress.

Part of the simulations were made on a Cyber 76 scalar computer and part on a Cyber 205 vector computer. We described the computer program in sufficient detail to introduce the reader to the difficulties of vector programming, which paid off in a gain of speed by about one order of magnitude.

We remark that it would have saved computer time had a much larger memory been available on the Cyber 205. For then we could have made all simulations on a much larger lattice, instead of trying without much success to analyze for finite-size effects in various smaller lattice sizes.

ACKNOWLEDGMENTS

We thank S. Selberherr, H. G. Baumgartel, S. Wansleben, S. Havlin, K. Kehr, and R. Rammal for helpful advice and information, and Sonderforschungsbereich 125 for financial support.

NOTE ADDED IN PROOF

The time for one diffusion step on the vector computer was reduced to 218 ns by using the "register gather" trick described by S. Wansleben and J. G. Zabolitzky, preprint; it was further reduced to 167 ns if each site is stored in one word, and not in one bit. For two dimensions, J. G. Zabolitzky (preprint) found $\mu/\nu \simeq 0.97$ with the method of Ref. 26, which leads to $k = 0.330$ for the exponent in Fig. 7, more accurate than our Eq. (24).

REFERENCES

1. P. G. de Gennes, *La Recherche* 7:919 (1976).
2. C. Mitescu and J. Roussenq, *Ann. Israel Phys. Soc.* 5:81 (1983).
3. B. B. Mandelbrot, *Ann. Israel Phys. Soc.* 5:59 (1983).

4. Y. Gefen, A. Aharony, and S. Alexander, *Phys. Rev. Lett.* **50**:77 (1983); K. W. Kehr, *J. Stat. Phys.* **30**:509 (1983); R. Kutner and K. W. Kehr, *Phil. Mag.* **A48**:199 (1983).
5. S. Alexander and R. Orbach, *J. Phys. (Paris) Lett.* **43**:L625 (1982).
6. P. C. Hohenberg and B. I. Halperin, *Rev. Mod. Phys.* **49**:435 (1977).
7. C. Mitescu and M. J. Musolf, *J. Phys. (Paris) Lett.* **44**:L679 (1983).
8. B. Derrida, D. Stauffer, H. J. Herrmann, and J. Vannimenus, *J. Phys. (Paris) Lett.* **44**:L701 (1983).
9. D. Ben-Avraham and S. Havlin, *J. Phys. A* **15**:L691 (1982).
10. D. Stauffer, *Phys. Rep.* **54**:3 (1979); J. W. Essam, *Rep. Progr. Phys.* **43**:843 (1980).
11. R. B. Pandey and D. Stauffer, *Phys. Rev. Lett.* **51**:527 (1983).
12. S. Havlin and D. Ben-Avraham, *J. Phys. A* **16**:L483 (1983).
13. P. Meakin and H. E. Stanley, *Phys. Rev. Letters* **51**:1457 (1983).
14. T. Vicsek, *J. Phys. A* **16**:1215 (1983).
15. R. Rammal and G. Toulouse, *J. Phys. (Paris) Lett.* **44**:L13 (1983).
16. S. Havlin and D. Ben-Avraham, preprint (National Institutes of Health, 1983).
17. R. B. Pandey and D. Stauffer, *J. Phys. A* **16**:L511 (1983).
18. P. Argyrakis and R. Kopelman, preprint (University of Michigan, 1983).
19. S. Wilke, Y. Gefen, V. Ilkovic, A. Aharony, and D. Stauffer, *J. Phys. A*, in press (1983).
20. C. J. Fassnacht, *J. Undergrad. Res. in Phys.* **2**:23 (1983).
21. J. C. Angles d'Auriac, A. Benoit, and R. Rammal, *J. Phys. A* **16**:4039 (1983).
22. D. Lukas, preprint (Cologne University, 1983).
23. M. H. Kalos, private communication.
24. D. W. Heermann and D. Stauffer, *Z. Phys.* **B44**:333 (1981); A. Margolina, H. J. Herrmann, and D. Stauffer, *Phys. Lett.* **69A**:73 (1982).
25. K. Binder (ed.), *Monte Carlo Methods in Statistical Physics* (Springer-Verlag, Heidelberg, 1979).
26. B. Derrida and J. Vannimenus, *J. Phys. A* **15**:L559 (1982).
27. A. Margolina, Z. V. Djordjevic, D. Stauffer, and H. E. Stanley, *Phys. Rev. B* **28**:1625 (1983).
28. O. Patzold, preprint (Cologne University, 1983).
29. R. B. Pandey, preprint (Cologne University, 1983).