The Percolation Behavior of Current Through Square and Cubic Resistor Networks

Peter Gilbert

August 26, 1991

Abstract

This paper begins with a discussion of elementary percolation theory, especially the "critical phenomenon." This theory is used to study the behavior of current flowing across two and three dimensional disordered electrical networks composed of random resistors. Resistors are assigned resistance values according to a given probability, and the map taking probabilities to expected values of the effective conductance of the network is studied. Given a probability p , the forward problem is to estimate the value of this map. Given an object composed of a random mixture of two materials each having constant conductance, the inverse problem is to estimate the proportion p of materials from boundary measurements of the effective conductance of the object. Algorithms are presented which solve the forward and inverse problems and estimate the critical probability in two and three dimensions.

1 Introduction to Percolation Theory

Suppose that a large cubic chunk of turf is immersed in water. What is the probability that a grain of dirt near the center of the chunk of turf is wetted? Percolation theory is concerned with questions like this. In the standard "percolation model," let Z^d be the d dimensional lattice. Let p be a number satisfying $0 \leq p \leq 1$. Declare each edge of the lattice to be open with probability p and closed with probability $1-p$. Water can flow through an open edge and not through a closed edge; hence p is the proportion of passages which are broad enough to allow water to flow along them. Objects such as the chunk of turf have large networks of fissures which can be approximated by the infinite lattice Z^3 . Approximate the chunk of turf by Z^3 , and let x be a vertex of Z^3 near the center of the chunk of turf. Then the vertex x is wetted if and only if there is a path from x to a vertex on the boundary of the chunk of turf, using open edges only (see Figure 1). Percolation theory is concerned with the existence of such open paths.

1.1 The Critical Phenomenon

One of the most interesting aspects of percolation theory is the occurence of a "critical phenomenon." On the infinite lattice Z^d , define an open cluster to be a set of open edges in which for any two edges in the cluster, there exists a path of open edges connecting them. Define the percolation probability $\theta^d(p)$ to be the probability that the origin of the lattice Z^d belongs to an infinite open cluster. A fundamental theorem of percolation theory is that for dimension $d \geq 1$, there exists a critical probability p_c^d such that

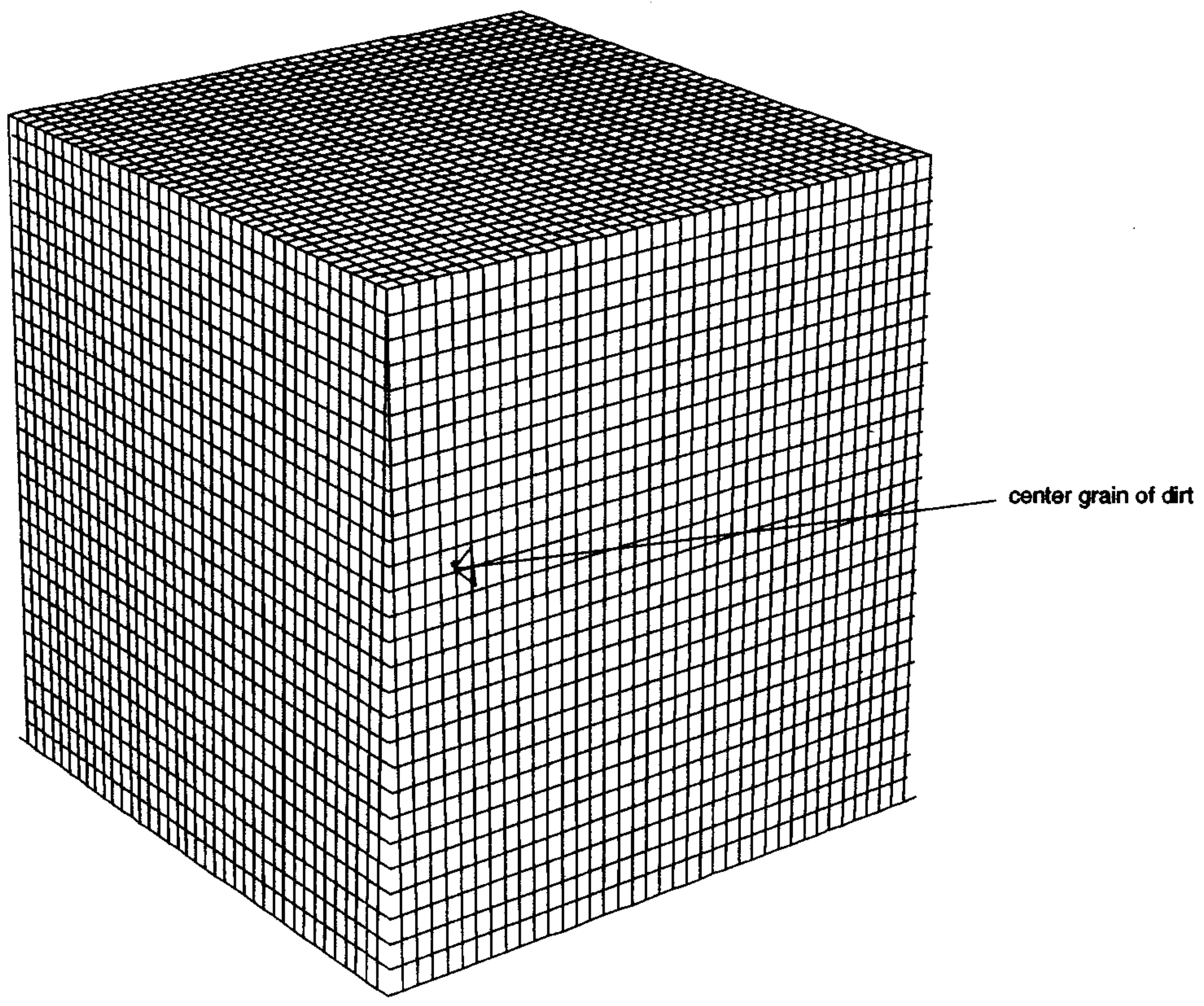
$$\begin{array}{rl} \theta^d(p) = & 0 \; if \; p < p_c \\ > & 0 \; if \; p > p_c \end{array}$$

Formally the critical probability p_c^d is defined by

$$p_c^d = sup\{p : \theta^d(p) = 0\}$$

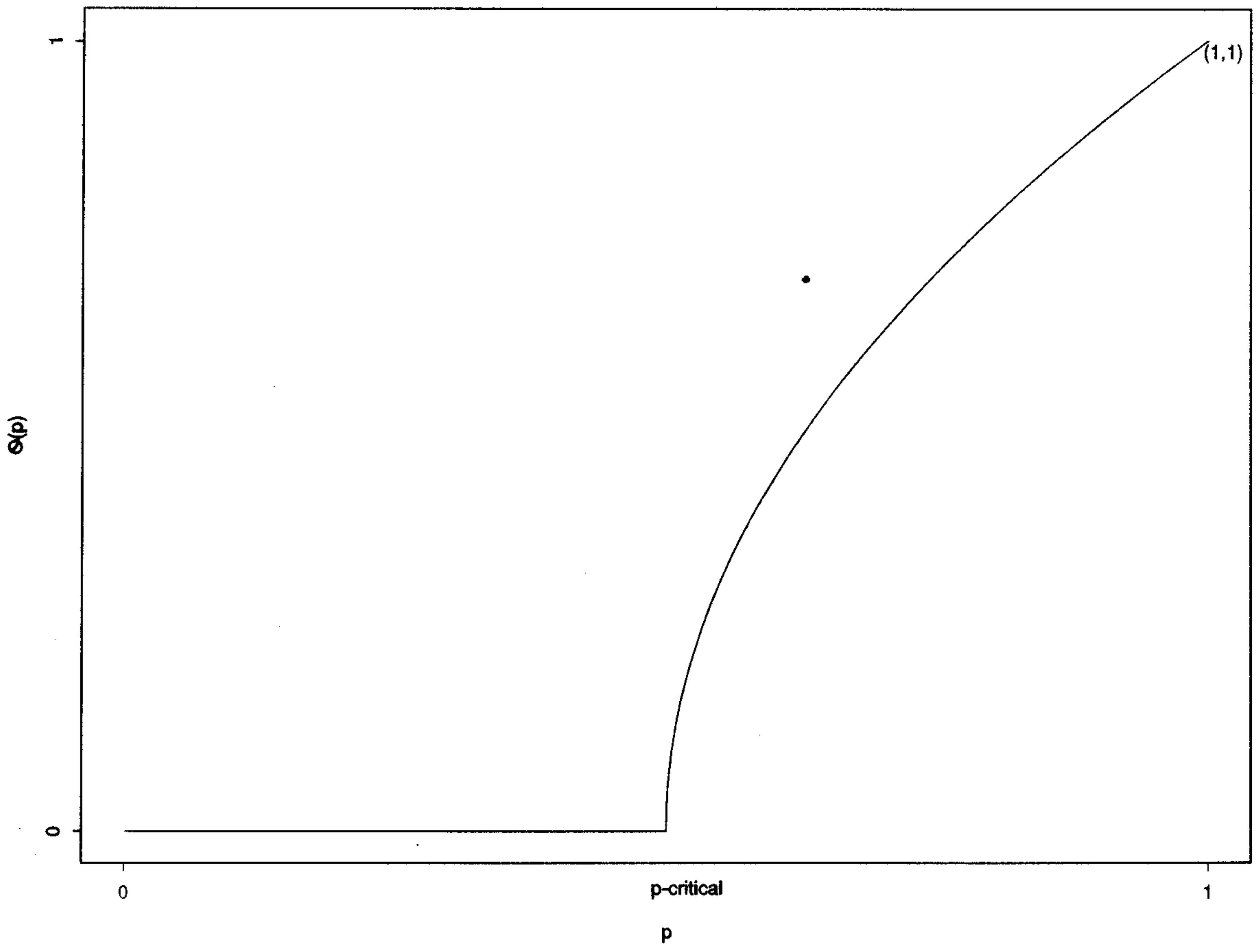
For a proof that the critical probability exists for $d \geq 1$, see Grimmett (p13). Grimmett asserts that the percolation probability $\theta(p)$ behaves roughly as in Figure 2.

Figure 1 Modelling the Network of Fissures in the Chunk of Turf



Is the Center Grain of Dirt Wetted when the Chunk of Turf is Immersed in Water?

Figure 2 Percolation Probability V Probability



The critical probability p_c^1 is clearly one, and it has been proved that $p_c^2 = 1/2$ (Grimmett, Kesten). For $d > 2$, the value of the critical probability is unknown. In this paper I develop a method to estimate the critical probability in two and three dimensions. The method involves studying the percolation of electrical current across square and cubic lattice resistor networks.

2 Setup of the Square and Cubic Resistor Networks

For $d = 2$, let $G_n^2 = \{1, 2, ..., n\} \times \{1, 2, ..., n\}$ be the interior nodes defining the square network, and let $G_{n0}^2 = \{$ (0,m): $1 \leq m \leq n\}$ and $G_{n1}^2 = \{$ (n+1,m): $1 \leq m \leq n\}$ be the left and right side nodes of G_n^2 , respectively. Assign a potential of zero to each of the nodes on G_{n0}^2 and a potential of one to each of the nodes on G_{n1}^2 . Insulate the top and bottom sides of the network. Place silver bars along the left and right sides of the network and connect a battery, creating a potential difference of one across the resistor network. The two-dimesional network G_5^2 connected to a battery is shown in Figure 3.

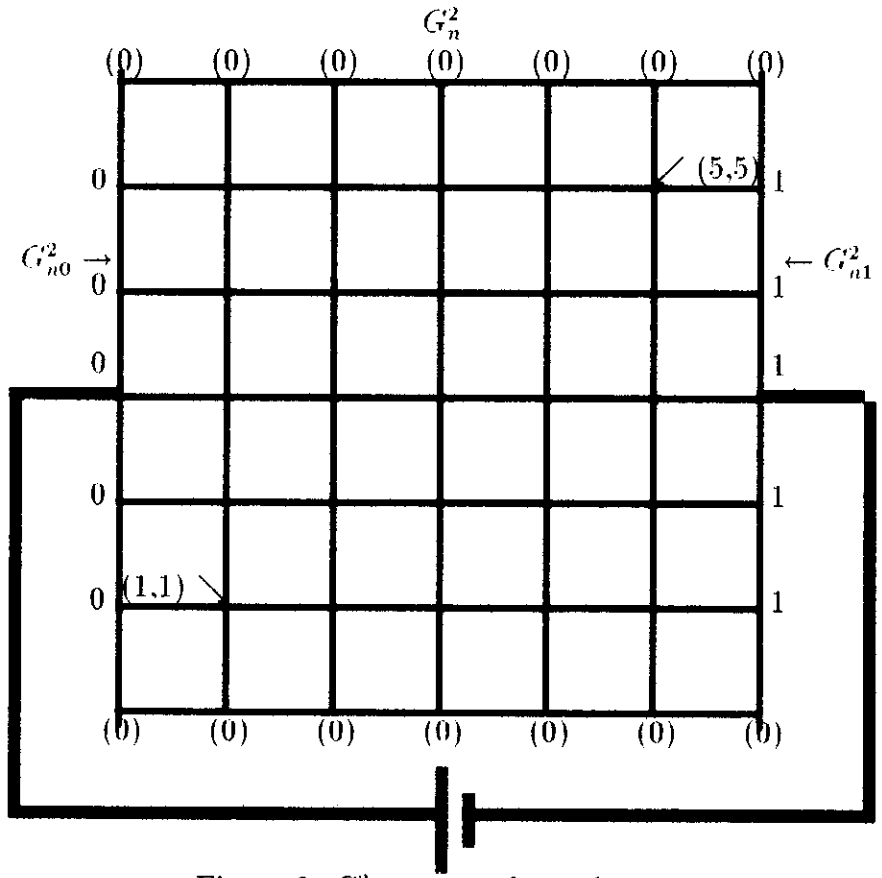


Figure 3: G_n^2 connected to a battery

For $d = 3$ the situation is similar. Here $G_n^3 = \{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\}$ are the interior nodes of the cubic network, and the nodes on the left and right faces are $G_{n0}^3 = \{ \text{(0,l,m)}: 1 \leq l, m \leq n\}$ and $G_{n1}^3 = \{ \text{(n,l,m)}: 1 \leq l, m \leq n\}$. Assign zero potentials to the nodes on the left face of the cubic network and one potentials to the nodes on the right face of the network. Insulate the top, bottom, front, and back faces of the network. Place silver plates along the left and right faces of the cubic network and connect a battery, creating a potential difference of one across the resistor network.

Given a square or cubic network G_n^d and a probability p satisfying $0 \leq p \leq 1$, assign each edge, independently of all other edges, a resistance of α with probability p and a resistance of β with probability $1 - p$. What is the effective resistance of the network? Given an object composed of a random mixture of two materials each having constant resistance, this problem is useful for determining the constitution of the mixture from measurements of effective resistance.

Suppose that $\alpha = 1$ and $\beta = \infty$. This characterizes edges as being either open or closed; open if current can flow along it (corresponding to finite resistance 1) and closed if current cannot flow along it (corresponding to infinite resistance). This specialized problem lies in the domain of percolation theory, and is the object of this paper. I study the behavior of $E_p(R_n^d)$, the expected value of the resistance of the network G_n^d , as a function of the probability p . Of primary interest is determining for which values of p the expected value of the resistance $E_p(R_n^d)$ is infinite. It is difficult to make accurate measurements of very large resistances, so the recipricol expected conductance values $E_p(C_n^d) = \frac{1}{E_p(R_n^d)}$ are studied.

3 The Map of Probabilities to Expected Conductance Values

Let $V = 1$ be the voltage drop across the resistor network G_n^d , and let I_n be the total current between the left and right sides or faces of the network G_n^d . By Ohm's Law, $C_n = I_n$, so that $C_n = 0$ is equivalent to no current passing from the right to the left side or face of the network. Let $h_n^d : P \longmapsto E_p(C_n^d)$ be the map of probabilities $p \in [0, 1]$ to expected values of conductance across the network. It is very difficult to explicitly describe the map h; I could only manage to express $h_n^d(p)$ as a linear combination of edge conductance values and expected values of Kirchoff matrices. One evident property of h is that it is monotonic increasing, since $p_1 < p_2$ implies that there is a greater chance that the network $G_n^d(p_1)$ has more internal resistance than the network $G_n^d(p_2)$, which implies that more current is expected to cross $G_n^d(p_2)$ than $G_n^d(p_1)$.

Another property of the function h_n^d is that it can be explicitly calculated for $p = 1$. When $p = 1$, of course each edge of the resistor network has conductance one. Since the nodes of G_{n0}^d have zero potential and the nodes of G_{n1}^d have one potential, there is a voltage drop of $\frac{1}{n+1}$ across each horizontal edge. Kirchoff's Law gives, for each interior node $x \in G_n^d \setminus G_{n0}^d \cup G_{n1}^d$,

$$\sum_{y \in G_n^d, y \ adjacent \ to \ x} \mid Potential(x) - Potential(y) \mid = 0$$

By this law, since there are $n+1$ horizontal edges between the left and right sides or faces of the network, the following is deduced: the nodes directly to the left of the nodes on G_{n1}^d have potential $\frac{n}{n+1}$. the nodes two horizontal edges away from G_{n1}^d have potential $\frac{n-1}{n+1}$, and finally the nodes adjacent to G_{n0}^d have potential $\frac{1}{n+1}$. Now notice that there are n^{d-1} nodes on each vertical cross-section (a vertical line if d=2, a vertical plane into the page for d=3) of the network. By Ohm's Law, the total current leaving the side or face G_{n0}^d is $\frac{n^{d-1}}{n+1}$, so that

$$E_1(C_n^d) = \frac{n^{d-1}}{n+1}.$$

4 The Forward Problem

Given a probability $p \in [0,1]$ and a resistor network G_n^2 or G_n^3 , assign to each edge of the network a conductance μ with probability p and conductance γ with probability $1-p$. In two dimensions connect a battery and set up the network as shown in Figure 3, and in three dimensions set up the network in the anologous way. The forward problem is to estimate the value of $h_n^d(p)$. In section 8 I describe an algorithm which calculates the sample mean conductance $\bar{C}_n^d(p)$ corresponding to the probability p . I wrote a simple function in Splus titled *forward*, which takes the sample mean, the sample size, and the sample standard deviation as parameters and returns a 95% confidence interval about the true mean conductance $h_n^d(p)$. For given p, the confidence interval is calculated as

$$(\bar{C}_n^2(p) - 1.96 * \frac{\sigma}{sqrt(n)}, (\bar{C}_n^2(p) + 1.96 * \frac{\sigma}{sqrt(n)}).$$

where σ is the sample standard deviation and 1.96 is the Z statistic corresponding to $\alpha = .05$ in the standard normal distribution. In Table 1 there is a list of confidence intervals for several probabilities calculated on the two dimensional network with 20 nodes on a side G_{20}^2 .

Probability	*Lower Limit for Confidence Bar*	*Upper Limit for Confidence Bar*
0.00	$9.52\ \epsilon^{-11}$	$9.52\ \epsilon^{-11}$
0.10	$1.19\ \epsilon^{-11}$	$1.21\ \epsilon^{-11}$
0.20	$1.60\ \epsilon^{-11}$	$1.64\ \epsilon^{-11}$
0.30	$2.56\ \epsilon^{-11}$	$2.74\ \epsilon^{-11}$
0.40	$5.66\ \epsilon^{-11}$	$6.89\ \epsilon^{-11}$
0.45	0.0023	0.0075
0.46	0.0058	0.013
0.47	0.0059	0.014
0.48	0.0098	0.019
0.49	0.015	0.026
0.50	0.022	0.035
0.51	0.027	0.044
0.52	0.043	0.058
0.53	0.049	0.068
0.54	0.060	0.078
0.55	0.066	0.085
0.60	0.162	0.182
0.70	0.349	0.369
0.80	0.548	0.564
0.90	0.752	0.764
1.00	0.952	0.952

Table 1: Confidence Intervals about values of $h^2_{20}(p)$

5 The Inverse Problem

Let O be an object which is known to be a random mixture of two materials having constant conductance μ and γ respectively. Approximate the object O by the appropriate square or cubic resistor network G_n^2 or G_n^3. Connect a battery and set up the network as described in section 2. Measure the effective conductance of the network m times. and compute the sample mean $C_n^d(p)$. The inverse problem is to recover the constitution of the object O by estimating the probability p, which is the proportion of materials composing the object O. I wrote an Splus function titled *inverse* which takes a range of probabilities, the corresponding sample means $\bar{C}_n^d$, the corresponding sample standard deviations, the sample size m and the dimension d as parameters and returns a 95% confidence interval about the true probability p. See the Appendix for information on how the function calculates the confidence interval. In Table 2 there is a list of confidence intervals about the true mean p corresponding to 100 effective conductance measurements taken on the two dimensional network with 20 nodes on a side G_{20}^2 .

Sample Mean	*Lower Limit for Confidence Bar*	*Upper Limit for Confidence Ba*
0.001	0.0090	0.4740
0.01	0.0090	0.4900
0.0289	0.4510	0.5100
0.05	0.4910	0.5430
0.10	0.5530	0.5790
0.20	0.6140	0.6210
0.30	0.6610	0.6720
0.40	0.7110	0.7210
0.50	0.7630	0.7740
0.60	0.8130	0.8260
0.70	0.8620	0.8760
0.80	0.9100	0.9250
0.90	0.9750	0.9780

Table 2 Confidence bars about true probabilities on G_{20}^2 with m=100

5.1 The Critical Phenomenon of the Function h

Intuitively, for large n the function h_n^d is similar to the percolation probability function θ . Given p , $\theta(p)$ is the probability that the origin of the infinite lattice Z^d is a member of an infinite open cluster. For a large finite lattice, the existence of an infinite open cluster is intuitively a similar condition to the existence of an open path from one side or face of the network to the other. Hence $\theta^d(p) = 0$ corresponds to no open path existing from one side or face of the network to the other. Since it is known that a critical probability p_c^d exists for the function $\theta^d(p)$, it is reasonable to guess that the probability of an open path existing from one side or face of the network to the other exhibits critical behavior. Moreover, I claim that the probability function describing for which probabilities there exists an open path from one side or face of the network to the other is equivalent to the probability function describing for which probabilities positive current flows. To see this, recall that by Ohm's Law $C_n = I_n$. The total current I_n is calculated by first solving for all of the interior potentials of the network. Then

$$C_n^d = I_n^d = \sum_{v \text{ a node adjacent to } G_{n0}^d} Voltage(v)Conductance(edge\ to\ left\ of\ v).$$

From this expression it is clear that $C_n = 0$ if and only if there is an open path from G_{n0}^d to G_{n1}^d . Hence studying the probability of an open path existing from one side or face of the network to the other is equivalent to studying the probability of positive effective conductance.

Suppose that there is a critical probability κ^d for which no current flows across the network for probabilities less than κ^d and current flows across the network for probabilities greater than κ^d . Then, if κ^d exists, for large n the function h_n^d is of the form:

$$\begin{aligned} h_n^d(p) &= 0 \ if\ p < \kappa^d \\ &> 0 \ if\ p > \kappa^d. \end{aligned}$$

For $d = 2$ I prove that κ^2 exists, and is equal to $p_c^2 = \frac{1}{2}$. For $d = 3$ I argue that κ^3 exists. For $d = 2$ and $d = 3$ I describe an algorithm which maps probabilities to sample mean conductances of the network G_n^d . With this algorithm I approximate h_n^2 , and use the approximation to empirically validate the theoretical result that $\kappa^2 = \frac{1}{2}$. I then approximate h_n^3 and use the approximation to estimate κ^3 . I conjecture that $\kappa^3 = p_c^3$, so that through this algorithm the unknown critical probability p_c^3 is estimated.

6 Proof That κ^2 Exists and Equals $\frac{1}{2}$

Theorem:

Let G_n^2 be a square resistor network, where each edge has conductance 1 with probability p and conductance 0 with probability

$1-p$. Let $E_p(C_n^2)$ be the expected value of the conductance between G_{n0}^2 and G_{n1}^2 . Then there exists a positive integer N such that $n \geq N \Rightarrow$

$$P\{E_p(C_n^2) = 0\} = 1 \ if \ \ p < \frac{1}{2}, \tag{1}$$

$$P\{E_p(C_n^2) > 0\} = 1 \ if \ \ p > \frac{1}{2}. \tag{2}$$

Equation (1) shows that for large enough square networks, no current will flow across the network if $p < \frac{1}{2}$, and equation (2) shows that positive current will flow if $p > \frac{1}{2}$. Hence κ^2 exists and equals $\frac{1}{2}$. I prove this Theorem, leaving out some in-depth percolation theory arguments which digress from the focus of this paper.

6.1 Proof of (1)

Let $p \in [0, \frac{1}{2})$. To begin the proof of (1), recall that in section 5.1 I showed that the effective conductance C_n^2 of the network is zero if and only if there is no conducting path from G_{n0}^2 to G_{n1}^2 . Hence

$$P\{E(C_n^2) = 0\} = P\{There\ does\ not\ exist\ an\ open\ path\ from\ G_{n0}\ to\ G_{n1}\}.$$

Select an interior node near the center of the resistor network G_n^2 to be the origin. Let W be the open cluster centered at the origin, and let $\#W$ denote the number of edges in W . Finally let $E_p(\#W)$ be the expected value of $\#W$ on the network G_n^2 .

Next, define a network H_n^2 by stacking three networks G_n^2 on top of each other. Then H_n^2 is described by its interior nodes $\{1, 2, ..., n\} \times \{1, 2, ..., 3n\}$ and its left and right side nodes $H_{n0}^2 = \{(0, m) : 1 \leq m \leq 3n\}$ and $H_{n1}^2 = \{(n+1, m) : 1 \leq m \leq 3n\}$ (see Figure 4).

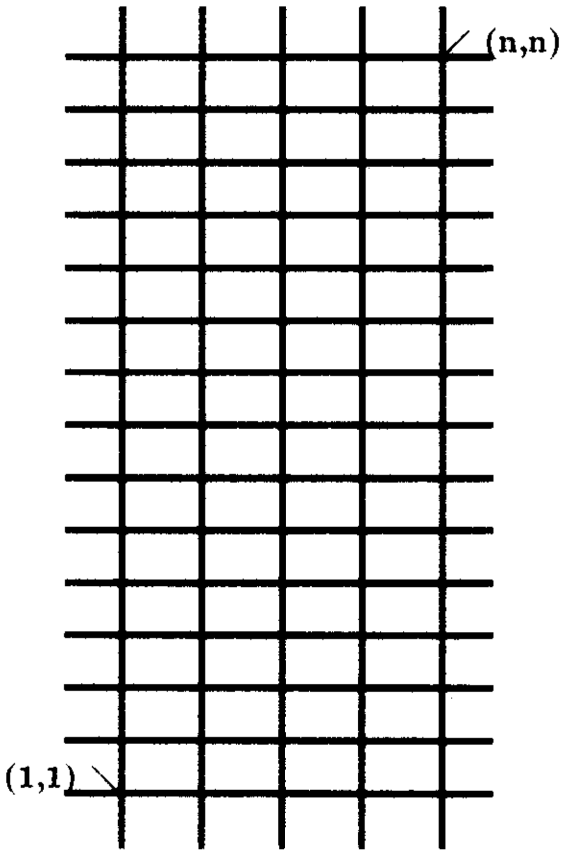


Figure 4: H_n^2

Let $\Psi(p)$ be the probability that there exists a conducting path from H_{n0}^2 to H_{n1}^2 . Harry Kesten, in Percolation Theory for Mathematicians (p 83), proved that if $p \in [0, \frac{1}{2})$, and $E_p(\#W) < \infty$, then $\Psi(p) \to 0$ *as* $\to \infty$. The proof is long and digresses from this papers focus, so it is not included. Call Kesten's result Lemma 1:

Lemma 1:

If $p \in [0, \frac{1}{2})$ and $E_p(\#W) < \infty$, then $\Psi(p) \to 0$ *as* $n \to \infty$.

An important result of percolation theory is that if the conclusion of Lemma 1 holds, then as n increases, the probability that $\#W$ is greater than or equal to n decays exponentially. Call this result Lemma 2 (see Kesten, p 83):

Lemma 2:

If $\Psi(p) \to 0$ as $n \to \infty$, then there exist constants $0 < c, k < \infty$ such that

$$P_p\{\#W \geq n\} \leq c\epsilon^{-kn}.$$

By the definition of the percolation probability function $\theta^2(p)$ given in section 1.1, evidently $E_p(\#W) = \theta^2(p)$. For $p \in [0, \frac{1}{2})$, $p < p_c^2$, so $E_p(\#W) = \theta^2(p) < \infty$. Hence the hypothesis of Lemma 1 holds, and its conclusion is that $\Psi(p) \to 0$ as $n \to \infty$. Then by Lemma 2, for constants $0 < c, k < \infty$,

$$P_p\{\#W \geq n\} \leq c\epsilon^{-kn}. \tag{3}$$

Since current has to travel through at least $n+2$ horizontal edges to flow from G_{n0}^2 to G_{n1}^2 , clearly

$$P_p\{E(C_n^2) = 0\} \geq P\{\#W < n\}. \tag{4}$$

Since $P_p\{\#W < n\} = 1 - P\{\#W \geq n\}$, (3) and (4) imply that

$$1 - P_p\{E(C_n^2) = 0\} \leq P\{\#W \geq n\}. \tag{5}$$

Then by (3), $1 - P_p\{E(C_n^2) = 0\} \leq c\epsilon^{-kn}$, which implies that

$$P_p\{E(C_n^2) = 0\} \geq 1 - c\epsilon^{-kn}. \tag{6}$$

Clearly $1 - c\epsilon^{-kn} \to 1$ as $n \to \infty$. Since (6) holds for $p < \frac{1}{2}$, (1) is proved.

6.2 Proof of (2)

Let $p > \frac{1}{2}$. Some definitions are required to prove (2). Define $\Phi(p, hor)$ to be the probability that there exists an open path from G_{n0}^2 to G_{n1}^2 , and $\Phi^*(p, hor)$ the probability that there exists a closed path (all edges have zero conductance) from G_{n0}^2 to G_{n1}^2 . In addition define $\Phi(p, ver)$ to be the probability that there exists an open path from the top to the bottom of the square network, and $\Phi^*(p, ver)$ to

be the probability that there exists a closed path from the top to the bottom of the square network. Notice that since there are the same number of horizontal edges as vertical edges in the network G_n^2 , $\Phi(p, hor) = \Phi(p, ver)$ and $\Phi^*(p, hor) = \Phi^*(p, ver)$. Kesten (p 172) proves that

$$\Phi(p, hor) + \Phi^*(p, ver) \geq 1. \tag{7}$$

The proof requires detailed manipulation of many results of percolation theory, so I will not explain it. Recall that in section 5.1 I showed that $E_p(C_n^2) > 0$ as soon as p is large enough so that the existence of an open path from G_{n0}^2 to G_{n1}^2 is assured. Hence

$$P_p\{E(C_n^2) > 0\} = \Phi(p, hor) \tag{8}$$

From (7),

$$\Phi(p, hor) \geq 1 - \Phi^*(p, ver) \tag{9}$$

Now, of course P_p { a given edge has conductance one} $= p$ and P_p { a given edge has conductance zero} $= 1 - p$, so evidently P_p { a given edge has conductance zero} $= P_{1-p}$ { a given edge has conductance one} . Extending this to sequences of edges,

$$\Phi^*(p, ver) = \Phi(1 - p, ver). \tag{10}$$

For $p > \frac{1}{2}$, obviously $1 - p < \frac{1}{2}$. Hence $E_{1-p}(\#W) < \infty$ and Lemma 1 holds, so that $\Psi(1-p) \to 0 \; as \; n \to \infty$. Then by Lemma 2, for constants $0 < c, k < \infty$,

$$P_{1-p}\{\#W \geq n\} \leq ce^{-kn}. \tag{11}$$

Now, since an open vertical path must consist of at least $n+1$ edges (since the network is $n+1$ edges high),

$$P_{1-p}\{E(C_n^2) = 0\} \geq P_{1-p}(\#W < n). \tag{12}$$

Since $P_{1-p}\{E(C_n^2) = 0\} = 1 - P_{1-p}\{E(C_n^2) > 0\} = 1 - \Phi(1 - p, hor) = 1 - \Phi(1 - p, ver)$,

$$1 - \Phi(1 - p, ver) \geq P_{1-p}\{\#W < n\}. \tag{13}$$

Evidently $P_{1-p}(\#W < n) = 1 - P_{1-p}(\#W \geq n)$, so that (11) becomes

$$1 - P_{1-p}(\#W < n) \leq ce^{-kn}, or \qquad (14)$$

$$P_{1-p}(\#W < n) \geq 1 - ce^{-kn}. \qquad (15)$$

Then by (13) and (15),

$$1 - \Phi(1 - p, ver) \geq 1 - ce^{-kn}. \qquad (16)$$

From (9) and (10),

$$\Phi(p, hor) \geq 1 - \Phi(1 - p, ver). \qquad (17)$$

Then by (16) and (17), $\Phi(p, hor) \geq 1 - ce^{-kn}$, and by the characterization (8),

$$P_p\{E(C_n^2) > 0\} \geq 1 - ce^{-kn}. \qquad (18)$$

Clearly $1 - ce^{-kn} \to 1 \; as \; n \to \infty$. Since (18) holds for $p > \frac{1}{2}$, (2) is proved.

7 Argument for the Existence of κ^3

At the end of section 5.1 I conjectured that there exists a critical probability κ^3 such that for $p < \kappa^3$ the expected conductance value of the network G_n^3 is zero and for $p > \kappa^3$ the expected conductance value is positive. Define H_n^3 to be the network resulting from stacking three cubic networks G_n^3 on top of each other; H_n^3 is the exact analog of H_n^2 . Define Γ_c^3 as:

$$\Gamma_c^3 = sup\{p : lim P_p\{\Upsilon(p) = 0\}\}, \qquad (19)$$

where $\Upsilon(p) = P_p$ { There exists an open path between the left and right faces of the network H_n^3 } .

Kesten (p333) proved that for $p < \Gamma_c^3$, $\Upsilon(p) = 0$, and for $p > \Gamma_c^3$, $\Upsilon(p)$ is uniformly bounded from zero for all n , at least for a subsequence of $n's$. Hence there is a critical probability describing the probability of an open path stretching across the network H_n^3 . Since the probability of nonzero expected current flow is equivalent to the probability of the existence of an open path from the left to the right face of the network, the existence of κ^3 is implied.

8 The Approximating Graphs of the Functions h_n^2

I wrote two Fortran programs titled SquareConductances.exe and CubicConductances.exe which create values approximating the functions h_n^2 and h_n^3 respectively. For each probability value $p = 0.00, 0.01, 0.02, 0.03, \ldots, 0.98, 0.99, 1.00$, SquareConductances.exe calculates a sample mean $\bar{C}_n^2(p)$, where m is the sample size. For given p , the following algorithm is performed m times to calculate a sample effective conductance: Using a random number generator, assign to each edge, independently of all other edges, a conductance of 1 with probability p and a conductance of zero with probability $1-p$. By Ohm's Law, since the potential difference between G_{n1}^2 and G_{n0}^2 is one, the effective conductance is equal to the total current flowing from G_{n1}^2 to G_{n0}^2 . The total current is calculated by first computing the potentials of the interior nodes of the network. These interior potentials are computed from the boundary information of zero potentials on left side nodes, one potentials on right side nodes, and zero current on top and bottom side nodes.

To solve for the interior potentials, Kirchoff's Law is used to set up a system of equations $Ax = b$, where x is the vector of interior potentials to be solved for, A is the n^2 by n^2 Kirchoff matrix containing information supplied by Kirchoff's Law, and b is a vector of length n^2 containing boundary information supplied by Kirchoff's Law. There are two techniques that I used to solve the linear system $Ax = b$. The first is a Linpackd routine which directly solves the linear system $Ax = b$. This method produces an accurate solution, but computation time is long and storage space must be allocated to declare the n^2 by n^2 Kirchoff matrix A . In the two dimensional case, these problems can be handled, but in the three dimensional case they are intolerable. The Kirchoff matrix in three dimensions is n^3 by n^3 , and for $n > 6$ it becomes impossible to declare the Kirchoff matrix. Hence in addition to writing programs that utilize Linpackd routines to explicitly solve the system $Ax = b$, I wrote programs using the Gauss-Seidel iteration scheme. This method only requires enough storage space to hold the solution vector x, and runs faster.

8.1 The Gauss-Seidel Iteration Scheme

Let L be the lower diagonal matrix consisting of the lower diagonal elements of A , and U the upper diagonal matrix consisting of the upper diagonal elements of A . Then clearly $A = L + U$, and the linear system $Ax = b$ becomes $(L + U)x = b$. To construct an iteration scheme, suppose that j indicates the jth iteration. The goal is to find a convergent sequence of vectors x^j to the solution vector x . The idea behind Gauss-Seidel iteration is to estimate x^{j+1} and x^j for each iteration and stop the iterations when x^{j+1} and x^j are sufficiently close to each other, i.e. $\| x^{j+1} - x^j \| \leq \epsilon$ for a prescribed $\epsilon > 0$. This of course does not guarantee convergence, but if the terms x^{j+1} and x^j are near enough, it is very likely that the final term x^{j+1} closely approximates the solution.

Now I motivate why Gauss-Seidel should work. If x^{j+1} and x^j are very near, then $(Lx^{j+1} + Ux^j) = b$ approximates the linear system $Ax = b$. Under the hypothesis that L is invertible, this approximation becomes

$$x^{j+1} = L^{-1}(b - Ux^j) = F(x^j) \quad (say). \tag{20}$$

So for each $j > 1$, x^{j+1} is expressed as a function of x^j , and x^{j+1} converges to a solution x if $F(x) = x$, for in this case $Lx = b - Ux$ by (20), so that $(L + U)x = b$ and hence $Ax = b$. Thus it is enough to show that F has a fixed point, which is true if F is a contraction mapping. Now, F is a contraction mapping if

$$\| x^{j+2} - x^{j+1} \| \leq C \| x^{j+1} - x^j \| \; for\ some\ C < 1\ and\ for\ all\ j \geq 1. \tag{21}$$

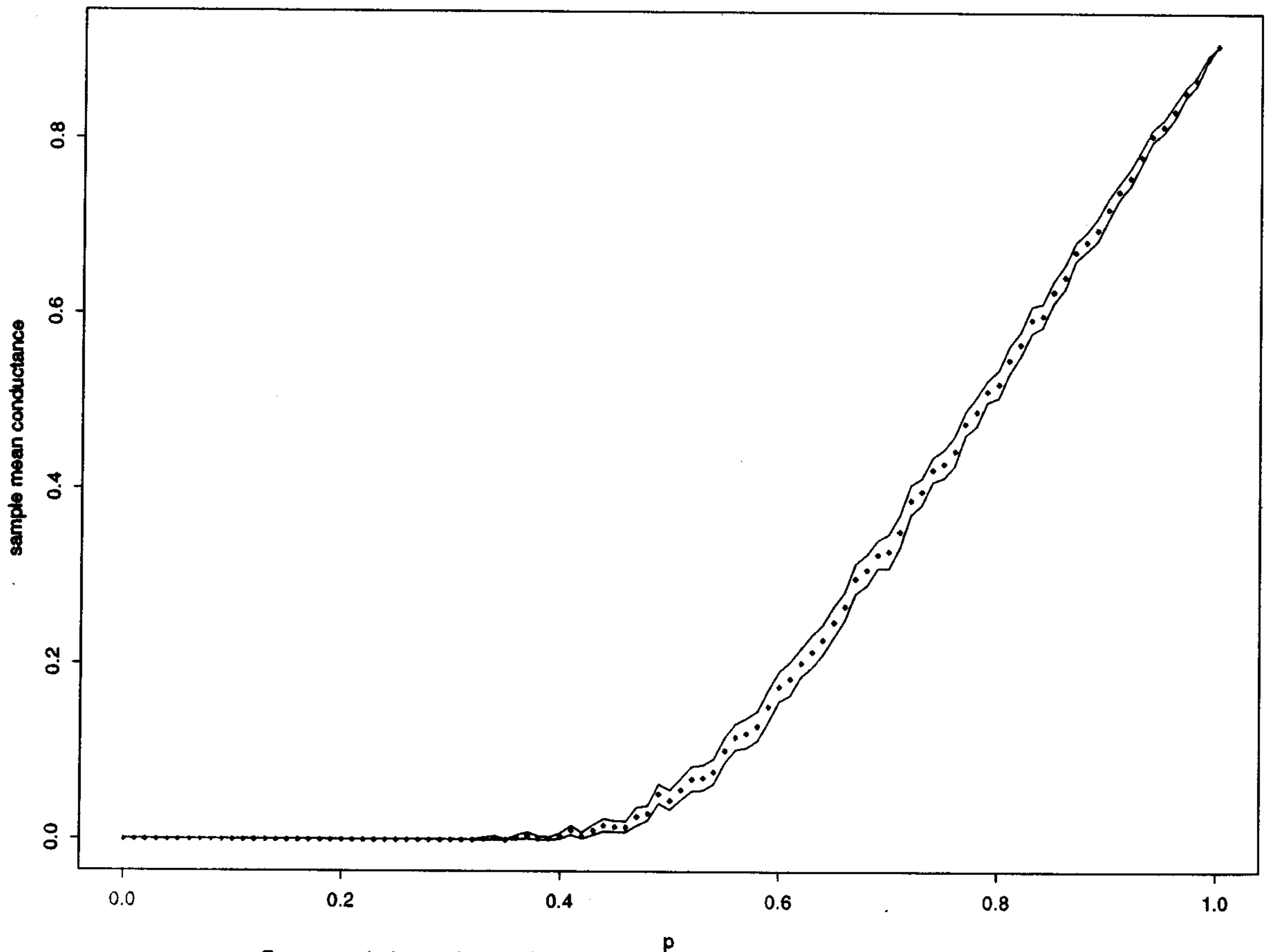
From (20), $x^{j+2} = L^{-1}(b - Ux^{j+1})$. Hence $\| x^{j+2} - x^{j+1} \| = \| L^{-1}(b - Ux^{j+1}) - L^{-1}(b - Ux^j) \| = \| L^{-1}b - L^{-1}Ux^{j+1} - L^{-1}b + L^{-1}Ux^j \| = \| -L^{-1}U(x^{j+1} - x^j) \| \leq \| L^{-1}U \| \| x^{j+1} - x^j \|$. Hence F is a contraction mapping if $\| L^{-1}U \| < 1$. It is very difficult to show this for a given matrix A, but since the Kirchoff matrix A is symmetric positive definate, this holds. In addition, the necessary hypothesis that the lower diagonal matrix L of the Kirchoff matrix A be invertible is satisfied. Hence the Gauss-Seidel method should produce a reliable solution vector x of interior potentials.

Now that the interior potentials are known, the total current I_n^2 can be calculated using Ohm's Law:

$$I_n^2 = \sum_{v \text{ a node adjacent to } G_{n0}^2} Potential(v)Conductance(edge\ to\ left\ of\ v). \tag{22}$$

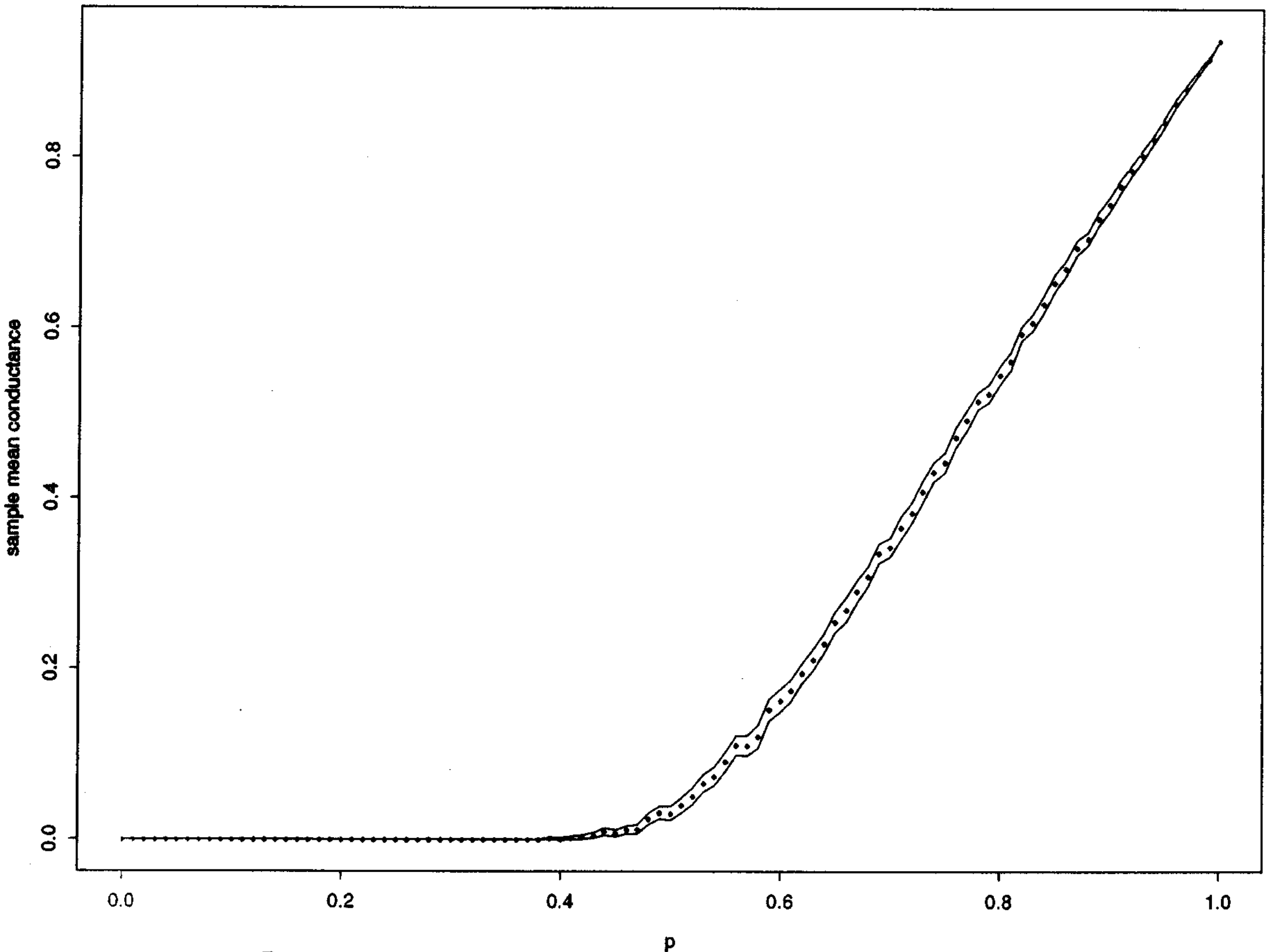
In addition to calculating sample mean conductances, the program SquareConductances.exe also computes sample standard deviations. For each p these sample standard deviations are used to calculate 95% confidence intervals about the true mean conductance. Figures 5, 6, and 7 are plots of $\bar{C}_n^2(p)$ versus p for square networks with 10, 15, and 20 nodes on a side respectively. In the plots, d is the dimension of the network G_n^2 , n is the number of nodes on a side of the network and m is the sample size of computed conductances used to calculate the sample mean for each p . The value $test$ is the small positive number which is used as a stopping criteria for the Gauss- Seidel iteration scheme- when $\| x^{j+1} - x^j \| \leq test* \| x^j \|$ the iteration scheme halts. In general, if $test = 10^{-z}$, where z is a positive integer, then C_n^2 is computed with an accuracy of about z significant digits. Finally, $seed$ is a negative integer which is used to initiate the sequence of random numbers produced by the random number generator. For a given negative integer, the sequence of random numbers will always be the same. This is convenient because a given experiment can be exactly replicated. For each graph, 95% confidence intervals about the true mean conductances are included, which are calculated by the Splus function *forward* , which I described in section 4.

Figure 5 Sample Mean Conductance V Probability d=2,n=10,m=100,test=.0001,seed=-1



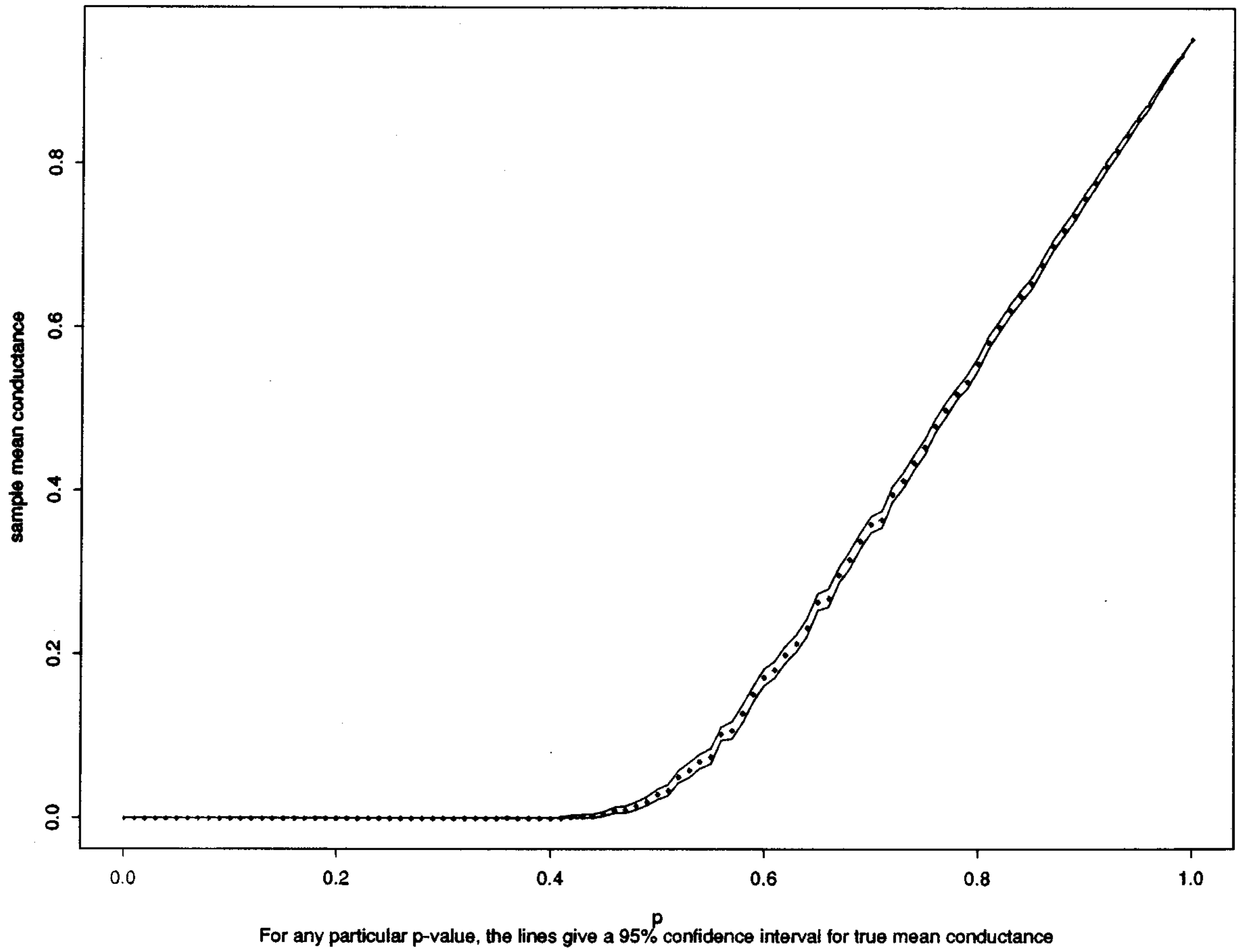
For any particular p-value, the lines give a 95% confidence interval for true mean conductance

Figure 6 Sample Mean Conductance V Probability d=2,n=15,m=100,test=.0001,seed=-1



For any particular p-value, the lines give a 95% confidence interval for true mean conductance

Figure 7 Sample Mean Conductance V Probability d=2,n=20,m=100,test=.0001,seed=-1



8.2 Analysis of the Approximating Graphs of the Functions h_n^2

The most striking feature of the graphs in Figures 5, 6 and 7 is that the average conductances are all zero or very near zero until a region where suddenly nonzero average conductances appear. The critical probability κ^2 must lie somewhere in this region. A comparison of Figures 5, 6 and 7 reveals that as n increases, the region of change from zero to nonzero average conductances narrows. This suggests that the program SquareConductances.exe can be used to approximate κ^2 to an accuracy as great as desired by studying larger networks. The problem of course is its enormous running time (on the MSCC computers SquareConductances.exe took about 10,000 minutes to complete for a network with $n = 20$).

Note that as n increases from 10 to 15 to 20, the region of change from zero to nonzero average conductances moves to the right. Nonzero average conductances begin to appear at about $p = .4$ for the case $n = 10$, at about $p = .44$ for the case $n = 15$ and at about $p = .47$ for the case $n = 20$. This suggests that as n gets large, nonzero average conductances will begin to appear at probabilities closer and closer to $\frac{1}{2}$. Hence this empirical approximation of the critical probability κ^2 agrees with its theoretical value of $\frac{1}{2}$. Since κ^2 is equal to the critical probability p_c^2 , p_c^2 is estimated by this algorithm.

Another feature of the graphs in Figures 5, 6 and 7 is that they are approximately linear for values of p greater than the approximated critical probability. Comparing Figures 5 and 6 with Figure 7 shows that as n increases, the graph gets more linear. Moreover, in section 3 I showed that for given n, $h_n^2(1) = \frac{n}{n+1}$, so that this linear portion has slope approximately equal to $\frac{2n}{n+1}$. This leads to my conjecture that as $n \longrightarrow \infty$, h_∞^2 is of the following form:

$$h_\infty^2 = \begin{cases} 0 & if\ p < \kappa^2 = \frac{1}{2} \\ 2p-1 & if\ p \geq \frac{1}{2}. \end{cases}$$

The conjectured graph of the function h_∞^2 is depicted in Figure 8.

9 The Approximating Graphs of the Functions h_n^3

I wrote a Fortran program titled CubicConductances.exe which creates values of the function h_n^3 . The program is exactly analogous to SquareConductances.exe except that sample mean conductances across the network G_n^3 are calculated for the 31 probabilities $p = 0.01, 0.04, 0.07, 0.10, 0.13, 0.16, ..., 0.94, 0.97, 1.00$ instead of the 101 probabilities $p = 0.00, 0.01, 0.02, 0.03, ..., 0.98, 0.99, 1.00$ as in SquareConductances.exe. The reason for this is that the Gauss-Seidel algorithm used to solve for the interior potentials of a cubic network takes n times as long as the Gauss-Seidel algorithm used to solve for the interior potentials of a square network. Ohm's Law, Kirchoff's Law, and the Gauss-Seidel iteration scheme are essentially unchanged. The only difference is that each interior node of a three dimensional lattice network has six neighboring edges whereas each interior node of a two dimensional lattice network has four neighboring edges. Figures 9 and 10 depict plots of $C_n^3(p)$ versus p for three dimensional networks with 3 and 5 nodes on a side, respectively.

Figure 8 Conjectured Graph of the Two Dimensional Function h as n approaches infinity

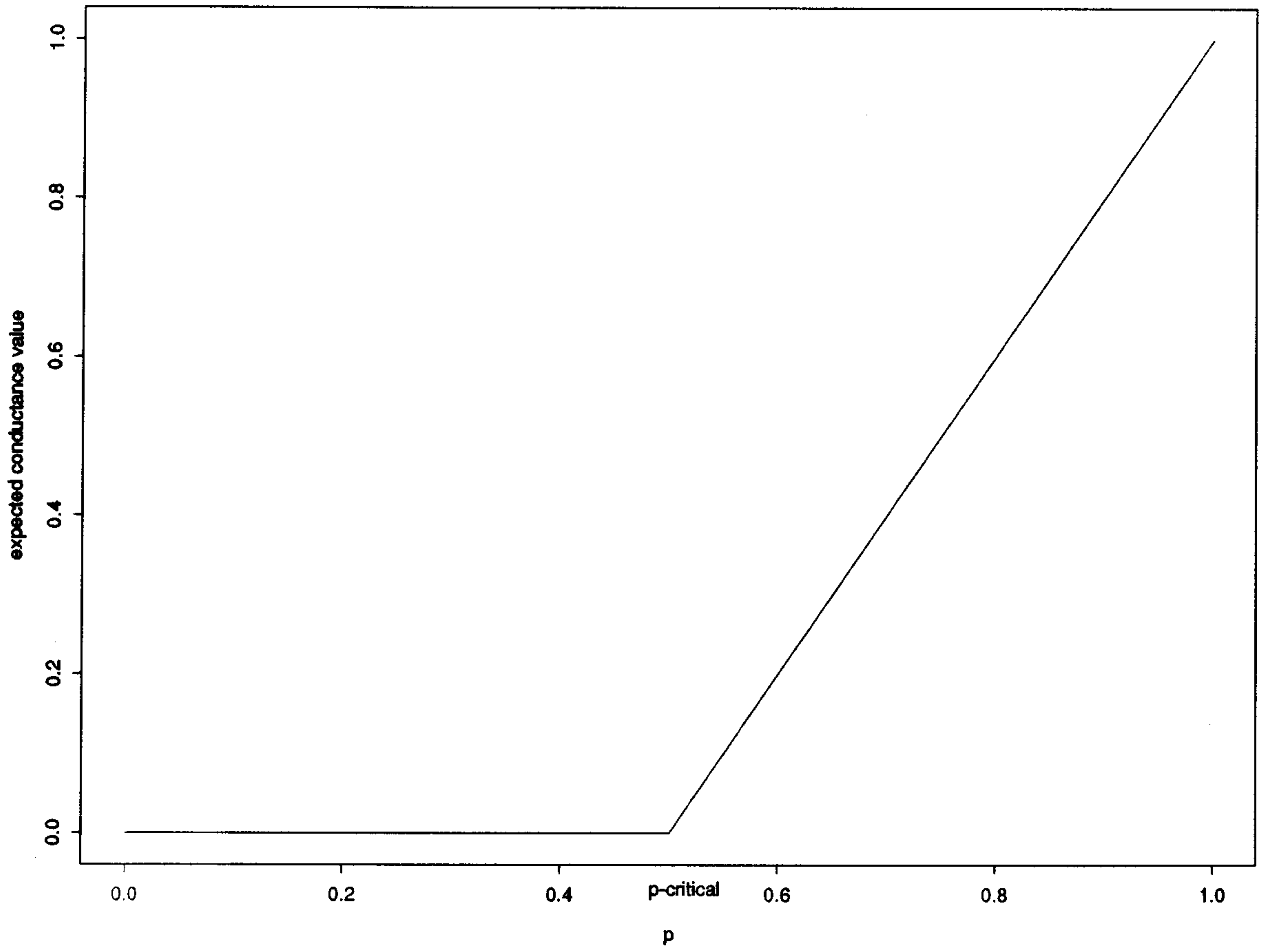
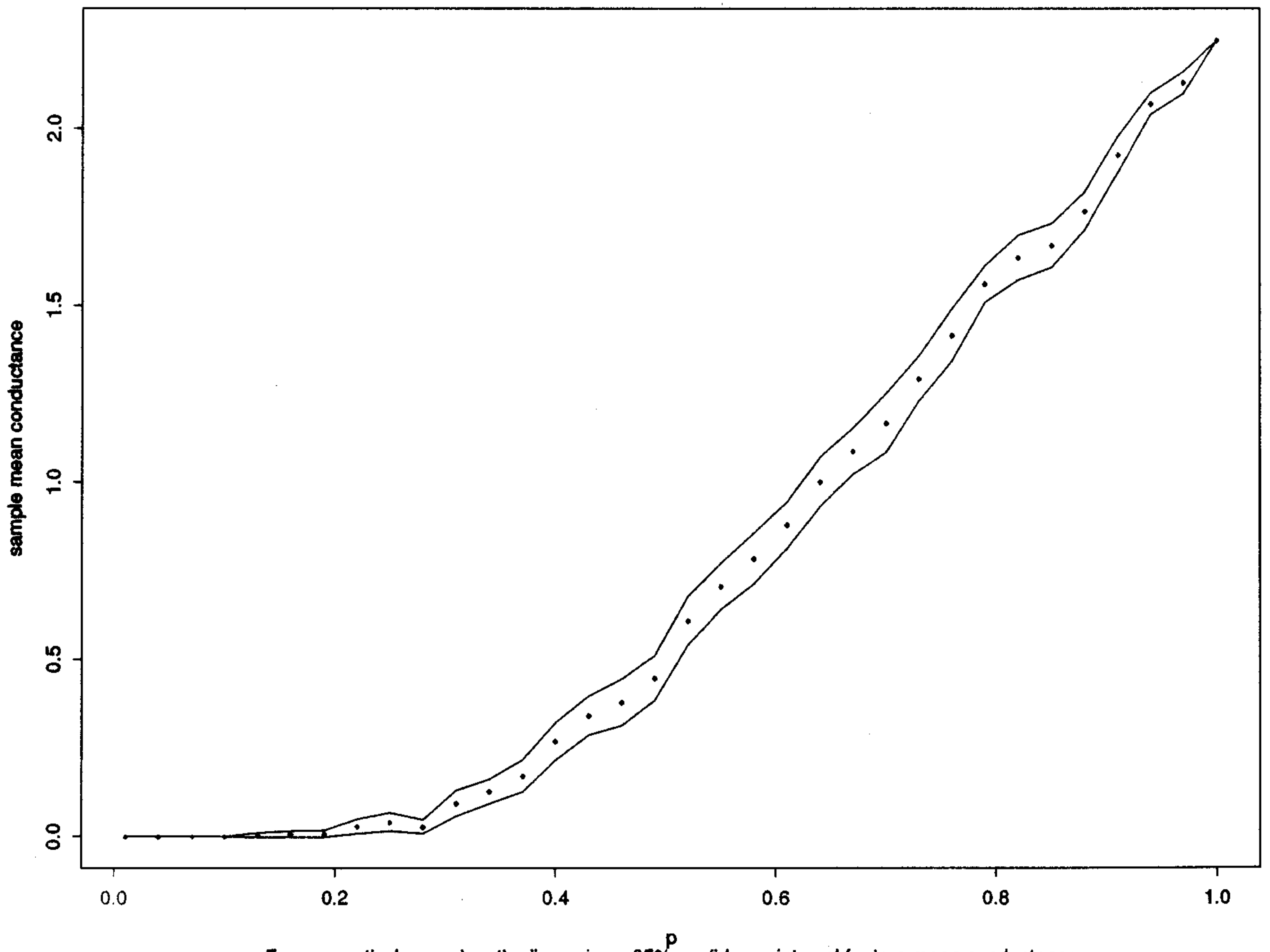
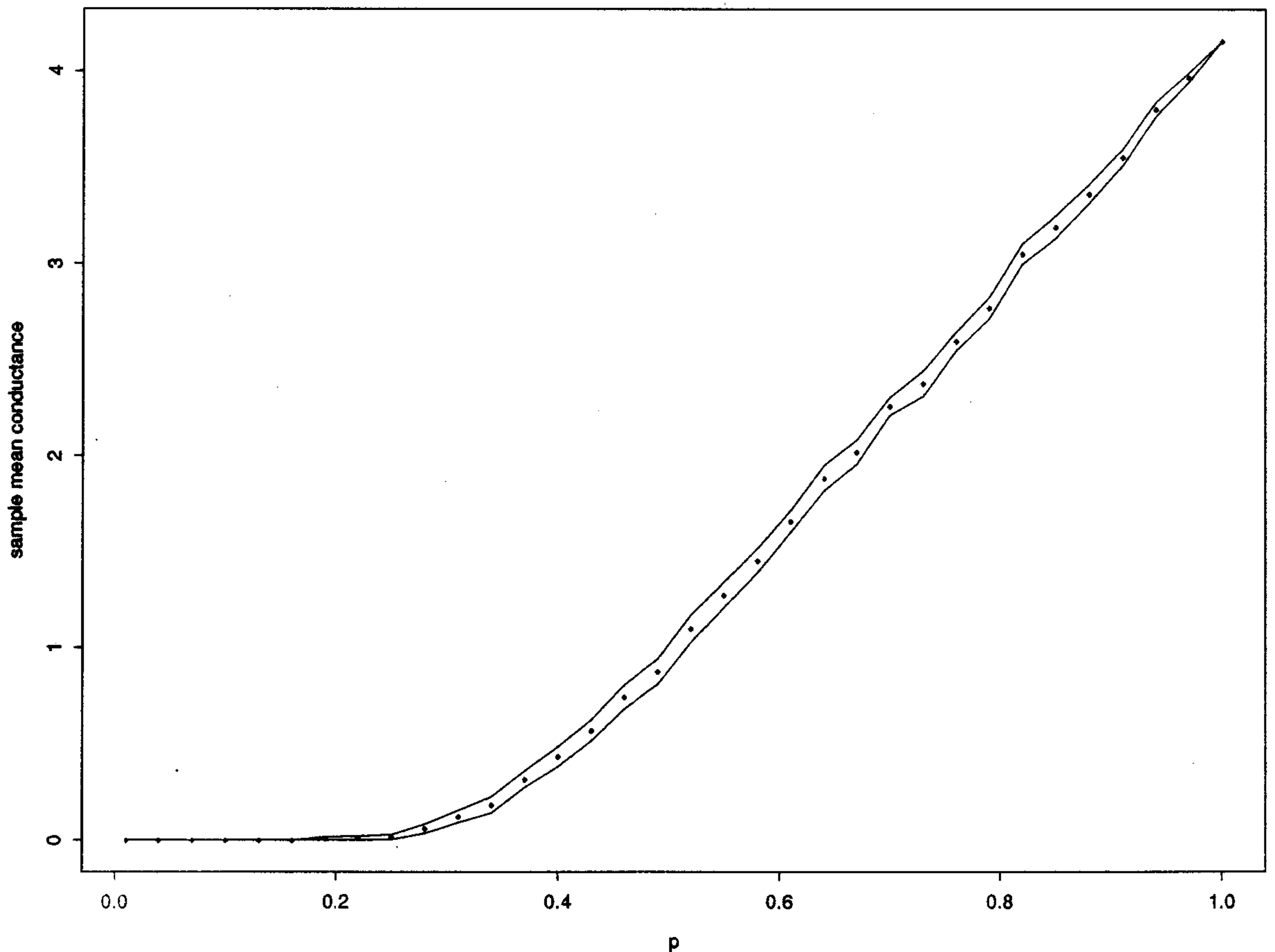


Figure 9 Sample Mean Conductance V Probability d=3,n=3,m=50,test=.0001,seed=-1



For any particular p-value, the lines give a 95% confidence interval for true mean conductance

Figure 10 Sample Mean Conductance V Probability d=3,n=5,m=50,test=.0001,seed=-1



For any particular p-value, the lines give a 95% confidence interval for true mean conductance

9.1 Analysis of the Approximating Graphs of the Functions h_n^3

Notice that in Figure 9 nonzero average conductances begin to appear approximately at about $p = .20$, and in Figure 10 nonzero average conductances begin to appear at about $p = .25$. I conjecture that just as in the two dimensional case, as n increases nonzero average conductances begin to appear at larger and larger probabilities. Grimmett (p13) proved that for $d \geq 1$, $p_c^{d+1} < p_c^d$. Hence $\kappa^{d+1} < \kappa^d$ for $d \geq 1$, and thus $p_c^3 = \kappa^3 < p_c^2 = \frac{1}{2}$. This gives the modest estimate that $\kappa^3 \in (\frac{1}{4}, \frac{1}{2})$. To obtain better estimates for the critical probability κ^3 , the algorithm CubicConductances.exe needs to be optimized.

10 Improving the Algorithms

One simple way of quickening the programs SquareConductances.exe and CubicConductances.exe is to use a relaxation parameter in the Gauss-Seidel iteration scheme. The Gauss-Seidel iteration scheme could also be made faster by lowering the value of $test$. I used the value of .0001 for the algorithms. Lowering this value would drastically reduce the run time, but would sacrifice much accuracy. For example, on a three dimensional network with 5 nodes on a side, it is known that $h_5^3(1) = \frac{25}{6} \doteq 4.167$. When CubicConductances.exe is run on the network G_5^3 and $test = .0001$, the sample mean computed for $p = 1$ is 4.158, .10 away from the true mean 4.167 . On the other hand, when $test = .001$ the sample mean computed for $p = 1$ is 4.080, .17 away from the true mean 4.167 . Another way to decrease computer run time would be to modify SquareConductances.exe and CubicConductances.exe to calculate sample means for a small range of probabilities. For example, if the goal is to approximate κ^2 , running SquareConductances.exe for probabilities $p = .47, .48, .49, .50, .51, .52, .53$ would be sufficient. Another way of speeding up the routines is to use smaller sample sizes to compute the sample means. The 95% confidence intervals are fairly narrow in all of the graphs, so it appears the sample size could be substantially lowered before a substantial amount of accuracy is lost.

11 Conclusion

In this paper, I studied the behavior of electrical current flowing across disordered square and cubic resistor networks composed of random resistors. With the construction of forward and inverse problems, I emperically studied the map $h : P \longrightarrow E_p(C_n^d)$ and its inverse. Considering that the 95% confidence intervals in Figures 5, 6, 7 are fairly narrow, I conclude that the algorithm SquareConductances.exe accurately solves the forward and inverse problems in two dimensions. In addition, the graphs created by the algorithm SquareConductances.exe clearly validate that the critical probability κ^2 exists. Figure 7 shows that with 95% confidence, the critical probability κ^2 lies somewhere between .47 and .53 . Since I proved in section 6 that $\kappa^2 = \frac{1}{2}$, this is an accurate approximation.

Since I could not run the algorithm CubicConductances.exe on large networks due to computer time restraints, CubicConductances.exe only modestly solves the forward and inverse problems. As well, it only modestly estimates the unknown critical probability κ^3 . At best, Figure 10 shows that with 95% confidence κ^3 is greater than .25 . In conclusion, the algorithm SquareConductances.exe accurately solves the forward and inverse problems and estimates κ^2 , but the algorithm CubicConductances.exe must be optimized and run on larger networks if it is to produce accurate solutions to the forward and inverse problems and accurately estimate κ^3 .

12 References

1 Grimmett, Geoffrey. Percolation . New York: Springer-Verlag, 1989.

2 Kesten,Harry. Percolation Theory for Mathematicians . Boston:Birkhauser, 1982.

13 Appendices

```
*       PROGRAM SquareConductances.exe
*
*   This program deals with square resistor networks.  Potentials of
* zero are assigned to the boundary nodes on the left side of the square
* network and potentials of one are assigned to the boundary nodes on
* the right side of the square network.  The boundary nodes on the
* top and bottom sides of the network are insulated.
*   This program has a controlling loop over probabilities valued 0,.01
* .02,.03 . . . .98,.99,1.00.  For each probability p, a random
* generator (the function RANDOM, in the file RANDOM.F) assigns each edge
* of the network a conductance value MU with probability p and a conductance
* value GAMMA with probability 1 - p.  MU and GAMMA are prompted for, and can
* be given nonnegative values.  Then the program calculates the effective
* conductance between the left and right sides of the square.  This procedure
* is repeated M times for each probability p, and the average of the
* effective conductances is computed.  As well, for each p the sample
* standard deviation of the conductances is calculated.  The probabilities,
* the corresponding average conductances, and the sample standard deviations
* are written to a file called 2Dexps.dat.  This program utilizes the
* Gauss-Seidel iteration scheme to calculate the interior potentials
* of the network.  A value TEST is prompted for; the smaller the value
* of TEST the more accurate the interior potentials will be.  In general
* if TEST is of order 10^-z, where z is a positive integer, then the
* effective conductance computed will have z significant digits.
* The Gauss-Seidel iteration scheme is faster than the alternative
* method of calculating interior potentials, via setting up the
* matrix equation Ax=b and using Lipackd routines.  However, the
* Gauss-Seidel scheme used in this program does not include a relaxation
* parameter, which would speed the program up much more.

*
* INDEXING
*
*   The indexing that I used was the following:  the leftmost boundary
* node on on the top face is the first node, and from there proceed from
* left to right and then top to bottom.  Hence for a network with N nodes
* on a side there are 4N*N**2 total nodes in the network, and the last node
* is the rightmost node on the bottom face.
*
*   The interior nodes have there own indexing:  the leftmost interior node
* adjacent to the top side of the network is the first node, and from
* there proceed from left to right then top to bottom.
*
*   Edges are indexed in the following manner:  Horizontal edges are
* numbered from top to bottom, left to right, and vertical edges are
* numbered from left to right, top to bottom.

*
* VARIABLES
* N      The number of nodes on a side
* M     The number of iterations of the random generator
* NUMEGS The number of edges of the network
* LIMIT  The maximum number of edges for a network
* MAXDIM The maximum number of nodes on a side
* HORCDS Stores the conductances of the horizontal edges
*        moving left to right then top to bottom
* VERCDS Stores the conductances of the vertical edges
*        moving top to bottom then left to right
* CURNTS Stores the currents of the edges adjacent to West face
*        boundary nodes
* A      The solution vector of interior node potentials
* ALAST Used to tell when the solution vector A has converged to
*        within TEST of the true potentials
* IXINTS An array containing the indices of the network corresponding
*        to the interior nodes
* INVIXS The inverse of IXINTS- an array containing the indices of the
*        interior nodes corresponding to the network indices
* IPVT   Used for the DGEFA and DGESL routines
* ()BNDS Stores the indices of the (N)orth, (E)ast
*        (S)outh, and (W)est boundary nodes
* LIXHOR LIXHOR and RIXHOR store the indices of the nodes straddling
* RIXHOR the horizontal edges
* TIXVER TIXVER and BIXVER store the indices of the nodes straddling
* BIXVER the vertical edges
* RE     The resistance from the left to right face
* CT     The total current flowing from left to right
* CDTS   An array of conductances from the left to right face.  Each
*        element holds one conductance corresponding to one set of
*        edge conductances
* STDEVS Vector of length 101, which will be filled with the sample
*        standard deviations
* P      Vector of probabilities
* ECD    The answer, which is the average of the elements of CD
* EXPS   Store all the expectations
* MU     One of the possible values for the conductance of an edge
* GAMMA  The other possible conductance value
* SEED   The value used as a paramater to RANDOM
* TEST   The small value which determines how many Gauss-Seidel iterations
*        take place.
* ANORM  The infinity norm of the vector of interior potentials, ALAST
* UNORM  The infinity norm of the difference between the most recently
*        calculated vector of interior potentials A and the previously
*        calculated vector of interior potentials ALAST

      INTEGER N, M, NUMEGS, LIMIT, MAXDIM, SEED,S,
     $ NBHORS(1:4),IXEGS(1:4)
      PARAMETER (LIMIT = 420)
      PARAMETER (MAXDIM=20)
      REAL A(MAXDIM**2),ALAST(MAXDIM**2),CDS(1:4),ECD,CT,
     $ P(101),MU,GAMMA, EXPS(101), TEST, STDEVS(1:101),CDTS(1:101)
       REAL HORCDS(1:LIMIT), VERCDS(1:LIMIT), ANORM,UNORM
       INTEGER IXINTS (1:LIMIT), INVIXS(1:LIMIT)
       INTEGER NBNDS(1:MAXDIM),EBNDS(1:MAXDIM)
       INTEGER SBNDS(1:MAXDIM),WBNDS(1:MAXDIM)
       INTEGER LIXHOR(1:LIMIT),RIXHOR(1:LIMIT)
       INTEGER TIXVER(1:LIMIT),BIXVER(1:LIMIT)
*
* SUBROUTINES CALLED
*
*
* SUBROUTINES USED FOR INDEXING
*
* INDINTNODES     Fills IXINTS with the indices of the interior nodes
* INVINDINTNODES Fills INVIXS with the indices of the network
*                 corresponding to the indices of the interior nodes
* INDBNDNODES     Fills NBND, EBND, WBND, and SBND with the indices
*                 of the network.  Boundary nodes are ordered starting
*                 in the leftmost position in the South face moving
*                 counterclockwise about the network.
* HORMAP          Fills LIXHOR and RIXHOR
* VERMAP          Fills TIXVER and BIXVER
*
* OTHER SUBROUTINES
*
* FILLCDS         Gets the neighboring conductances for any interior node
* RANDOM          A random number generator, called from external.  The code
*                 was found in NUMERICAL RECIPES.  A value SEED is required
*                 to start RANDOM, and the same sequence of random numbers
*                 is always produced for a given SEED.
*
*******
*** BEGIN
*
       INTRINSIC SQRT,ABS,REAL,MOD
       PRINT *, 'Enter the size of the square resistor (1-20)'
       READ *, N
```

```
      NUMEGS = 2*N*(N+1)

      PRINT *
      PRINT *, 'Enter the number of conductances to compute'
      READ *, M

      PRINT *
      PRINT *,'Enter MU(>=0), the conductance value with probability p'
      READ *, MU

      PRINT *
      PRINT *, 'Enter GAMMA(>=0), the conductance value with prob. 1-p'
      READ *, GAMMA

      PRINT *
      PRINT *, 'Enter TEST, the approximation level for Gauss-Seidel'
      READ *,TEST

      DO 100 I = 0,100
        P(I+1) = REAL(I)/REAL(100)
100   CONTINUE
* Fill P with probabilities .00 to 1.00

      CALL INDINTNODES (IXINTS, N)
* Fill IXINTS with the indices of the interior nodes

      CALL INVINDINTNODES (INVIXS, N)

      CALL INDBNDNODES (NBNDS, EBNDS, SBNDS, WBNDS, NMAXDIM)
* Fill NBNDS, EBNDS, SBNDS, and WBNDS with the indices of the boundary nodes

      CALL HORMAP (LIXHOR, RIXHOR, N)
* Fill LIXHOR and RIXHOR with the indices of nodes straddling horizontal edges

      CALL VERMAP (TIXVER, BIXVER, N)
* Fill TIXVER and BIXVER with the indices of nodes straddling vertical edges
* We now have the indices of the boundary and interior nodes

      DO 110 I = 1,101
        STDEVS(I) = 0
110   CONTINUE

* Initialize STDEVS to zero

*
*
**** CONTROLLING DOUBLE LOOP

       PRINT *
       PRINT *, 'Please enter a negative integer for a SEED'
       READ *, SEED
* Get a value for SEED to use in the random number generating function
* RANDOM

       DO 120 PP = 1,101
       ECD = 0
* Initialize the expected conductance to zero

       DO 130 J = 1, M

* Use the random number generator, found in NUMERICAL RECIPES,
* to determine the conductance of each edge.
       DO 6000 I = 1, NUMEGS/2
         IF (RANDOM(SEED) .LE. P(PP)) THEN
           HORCDS(I) = MU
           ELSE
           HORCDS(I) = GAMMA
         ENDIF
         IF (RANDOM(SEED) .LE. P(PP)) THEN
           VERCDS(I) = MU
           ELSE
           VERCDS(I) = GAMMA
         ENDIF
```

```
6000   CONTINUE

        DO 140 I = 1,N**2
          A(I) = 0
140     CONTINUE
* Initialize A with zeros

150     DO 160 I = 1,N**2
          ALAST(I) = A(I)
160     CONTINUE

        DO 170 S = 1, N**2

        CALL FILLCDS (CDS,HORCDS,VERCDS,IXINTS,LIXHOR,RIXHOR,TIXVER,
     $  BIXVER,NBHORS,IXEGS,N,NUMEGS, S, MAXDIM)
* Get the neighboring conductances surrounding the Sth interior node.
* CDS(1) is the left neighbor's conductance, CDS(2) is the right
* neighbor's conductance, CDS(3) is the above neighbor's conductance,
* and CDS(4) is the bottom neighbor's conductance.

* Take care of top, left corner
        IF (S .EQ. 1) THEN
          IF (CDS(1)+CDS(2)+CDS(4) .GT. .5) THEN
            A(S) = (A(S+1)*CDS(2)+A(S+N)*CDS(4))/
     $             (CDS(1)+CDS(2)+CDS(4))
          ENDIF

* Take care of top, right corner
        ELSEIF (S .EQ. N) THEN
          IF (CDS(1)+CDS(2)+CDS(4) .GT. .5) THEN
            A(S) = (A(S-1)*CDS(1)+CDS(2)+A(S+N)*CDS(4))/
     $             (CDS(1)+CDS(2)+CDS(4))
          ENDIF

* Take care of bottom, left corner
        ELSEIF (S .EQ. N**2 - N + 1) THEN
          IF (CDS(1)+CDS(2)+CDS(3) .GT. .5) THEN
          A(S) = (A(S+1)*CDS(2)+A(S-N)*CDS(3))/
     $               (CDS(1)+CDS(2)+CDS(3))
         ENDIF

* Take care of bottom, right corner
        ELSEIF (S .EQ. N**2) THEN
          IF (CDS(1)+CDS(2)+CDS(3) .GT. .5) THEN
            A(S) = (A(S-1)*CDS(1)+CDS(2)+A(S-N)*CDS(3))/
     $               (CDS(1)+CDS(2)+CDS(3))
         ENDIF

* Take care of left edge, excluding corners
        ELSEIF ((MOD(S-1,N) .EQ. 0) .AND.
     $         (S .NE. 1) .AND. (S .NE. N**2 - N + 1)) THEN
          IF (CDS(1)+CDS(2)+CDS(3)+CDS(4) .GT. .5) THEN
            A(S) = (A(S+1)*CDS(2)+A(S-N)*CDS(3)+A(S+N)*CDS(4))/
     $               (CDS(1)+CDS(2)+CDS(3)+CDS(4))
         ENDIF

* Take care of right edge, excluding corners
        ELSEIF ((MOD(S,N) .EQ. 0) .AND.
     $         (S .NE. N) .AND. (S .NE. N**2)) THEN
          IF (CDS(1)+CDS(2)+CDS(3)+CDS(4) .GT. .5) THEN
            A(S) = (A(S-1)*CDS(1)+CDS(2)+A(S-N)*CDS(3)+A(S+N)*
     $ CDS(4))/(CDS(1)+CDS(2)+CDS(3)+CDS(4))
         ENDIF

* Take care of top edge, excluding corners
        ELSEIF ((S .GT. 1) .AND. (S .LT. N)) THEN
          IF (CDS(1)+CDS(2)+CDS(4) .GT. .5) THEN
            A(S) = (A(S-1)*CDS(1)+A(S+1)*CDS(2)+A(S+N)*CDS(4))/
     $               (CDS(1)+CDS(2)+CDS(4))
         ENDIF

* Take care of bottom edge, excluding corners
        ELSEIF ((S .GT. N**2-N+1) .AND. (S .LT. N**2)) THEN
          IF (CDS(1)+CDS(2)+CDS(3) .GT. .5) THEN
```

```
            A(S) = (A(S-1)*CDS(1)+A(S+1)*CDS(2)+A(S-N)*CDS(3))/
     $             (CDS(1)+CDS(2)+CDS(3))
          ENDIF
        ELSE
          IF (CDS(1)+CDS(2)+CDS(3)+CDS(4) .GT. .5) THEN
            A(S) = (A(S-1)*CDS(1)+A(S+1)*CDS(2)+A(S-N)*CDS(3)+
     $      A(S+N)*CDS(4))/(CDS(1)+CDS(2)+CDS(3)+CDS(4))
          ENDIF

        ENDIF

170    CONTINUE

       ANORM = 0
       UNORM = 0
       DO 500 I = 1,N**2
         IF (ABS(A(I)-ALAST(I)) .GT. UNORM) THEN
           UNORM = ABS(A(I)-ALAST(I))
         ENDIF
* Fill UNORM with the maximum component value of ABS(A(I)-ALAST(I))

         IF (ABS(ALAST(I)) .GT. ANORM) THEN
           ANORM = ABS(ALAST(I))
         ENDIF
* Fill ANORM with the maximum component value of ABS(ALAST(I))

500    CONTINUE

       IF (UNORM .GT. TEST*ANORM) GOTO 150
* Test if another iteration is necessary

*
* Now calculate the current across the network CT

       CT = 0
* Initialize the value of the CT to zero.
```

```
        DO 180 K = 1, N
          CT = CT + HORCDS(1 + (N+1)*(K-1))*A(1 + N*(K-1))
180     CONTINUE

          CDTS(J) = CT
          ECD = ECD + CT
130     CONTINUE

        ECD = ECD/M
        DO 190 I = 1,M
          STDEVS(PP) = STDEVS(PP) + (CDTS(I)-ECD)**2
190     CONTINUE

          STDEVS(PP) = STDEVS(PP)/(M-1)
          STDEVS(PP) = SQRT(STDEVS(PP))

          EXPS(PP) = ECD
120       CONTINUE

        CALL OUTPUT(N,P,EXPS,M,MU,GAMMA,STDEVS)

***********

        END

* PROGRAM FORWARDGS.F completed


        SUBROUTINE FILLCDS(CDS,HORCDS,VERCDS,IXINTS,LIXHOR,RIXHOR,TIXVER,
     $ BIXVER,NBHORS,IXEGS,N,NUMEGS, Q, MAXDIM)

        INTEGER NBHORS(1:4), IXEGS(1:4), MAXDIM,
     $ N, Q, NUMEGS, IPTQ
        REAL HORCDS(NUMEGS), VERCDS(NUMEGS),CDS(1:4)
        INTEGER IXINTS(NUMEGS), LIXHOR(NUMEGS), RIXHOR(NUMEGS)
        INTEGER TIXVER(NUMEGS), BIXVER(NUMEGS),NBHORS(1:4)
        IPTQ = IXINTS (Q)
* IPTQ contains the node index of the Qth interior node.
* The IXINTS (Q)th node has neighboring nodes with indices
* IPTQ - 1 to the left, IPTQ + 1 to the right,
* IPTQ - (N+2) above, and IPTQ + (N+2) below.
* Now the subroutine builds the array NBHORS, containing the indice
* numbers of the nodes surrounding the Qth interior node.

* The first two positions in the array are occupied by the
* indices of the nodes forming horizontal edges, and the second
* two positions are occupied by the indices of the nodes forming
* vertical edges.

       NBHORS (1) = IPTQ - 1
       NBHORS (2) = IPTQ + 1
       IF (IPTQ .LE. 2*N + 1) THEN
         NBHORS (3) = IPTQ - N - 1
       ELSE
         NBHORS (3) = IPTQ - N - 2
       ENDIF

       IF (IPTQ .GE. N**2 + 2*N - 1) THEN
         NBHORS (4) = IPTQ + N + 1
       ELSE
         NBHORS (4) = IPTQ + N + 2
       ENDIF

* Now the subroutine will fill the CDS array with the conductances.
* CDS(1) contains the conductance of the left edge, CDS(2) the
* right, CDS(3) the above, and CDS(4) the below.

* This loop locates the indices of the edges which are straddled
* by two nodes with known indices.

      DO 200 J = 1, NUMEGS
          IF (LIXHOR(J) .EQ. NBHORS(1)) THEN
            IF (RIXHOR(J) .EQ. IPTQ) THEN
```

```
              IXEGS(1) = J
              IXEGS(2) = J + 1
            ENDIF
          ENDIF

          IF (TIXVER(J) .EQ. NBHORS(3)) THEN
            IF (BIXVER(J) .EQ. IPTQ) THEN
              IXEGS(3) = J
              IXEGS(4) = J + 1
            ENDIF
          ENDIF

200     CONTINUE

        CDS(1) = HORCDS(IXEGS(1))
        CDS(2) = HORCDS(IXEGS(2))
        CDS(3) = VERCDS(IXEGS(3))
        CDS(4) = VERCDS(IXEGS(4))
        END

*
*       SUBROUTINES FOR INDEXING
*
*
* HORMAP  This subroutine fills the inputed array B1 with the node indices
* which are to the left of each edge, i.e. the ith element of B1 contains
* the index of the left node which is straddled by the ith horizontal
* edge.  Similarly the array C1 is filled with the node indices which are
* to the right of each edge, i.e. the ith element of C1 contains the index
* of the right node which is straddled by the ith horizontal edge.

        SUBROUTINE HORMAP (B1, C1, N)

        INTEGER COUNTB, COUNTC, N
        INTEGER B1(2*N*(N+1)),C1(2*N*(N+1))
        COUNTB =-1
        COUNTC = -2
```

```
      DO 210 I = 1, N
        COUNTB = COUNTB + 1
        COUNTC = COUNTC + 2
        DO 220 J = 1, N+1

          B1(N*(I-1) + COUNTB + J) = I*N + J + COUNTC
          C1(N*(I-1) + COUNTB + J) = I*N + J + COUNTC + 1

220     CONTINUE
210   CONTINUE

      END

*
* VERMAP  This function is similar to HORMAP, it fills B1 and C1 with the
* indices of nodes which straddle vertical edges to the left and right
* respectively.

      SUBROUTINE VERMAP (B1, C1, N)

      INTEGER N, B1(2*N*(N+1)),C1(2*N*(N+1))

        DO 230 I = 1, N
          DO 240 J = 1, N+1

            IF (J .EQ. 1) THEN
              B1((N+1)*(I-1) + J) = I
            ELSE IF (J . EQ. N+1) THEN
              B1((N+1)*(I-1) + J) = 2*N + N**2 - 1 + I
            ELSE IF (J .EQ. 2) THEN
              B1((N+1)*(I-1) + J) = I + N + 1
            ELSE
              B1((N+1)*(I-1) + J) = B1((N+1)*(I-1)
     $        + J - 1) + N + 2
            END IF

                IF ((B1((N+1)*(I-1)+J) .GT. N**2 + 2*N -1) .OR.
     $                  (B1((N+1)*(I-1)+J) .LE. N)) THEN
                      C1((N+1)*(I-1) + J) = B1((N+1)*(I-1)+J)
     $                + N + 1
                    ELSE
                      C1((N+1)*(I-1) + J) = N+2 + B1((N+1)*
     $                (I-1) + J)
                    ENDIF
240           CONTINUE
230         CONTINUE

            END

* INDINTNODES takes an array of integers and an integer N as paramaters
* The subroutine inputs the index of the ith interior node in the ith position
* of the array.

            SUBROUTINE INDINTNODES (ARRAY, N)

            INTEGER N, ARRAY(N**2)
            DO 250 I = 1, N
              DO 260 J = 1, N

                ARRAY (N*(I-1) + J) = I*N + 2*I + J - 1
260           CONTINUE
250         CONTINUE
            END

*
* INVINDINTNODES   This subroutine fills the ith position of the
* inputed ARRAY with the index of the node corresponding to the
* ith interior node.  It is the inverse routine to INDINTNODES.

           SUBROUTINE INVINDINTNODES(ARRAY, N)

           INTEGER N, ARRAY(N**2 + 4*N)

           DO 270 I = 1,N
```

```
          DO 280 J = 1,N
            ARRAY(I*N + 2*I + J - 1) = N*(I-1)+J
280       CONTINUE
270     CONTINUE
        END

* INDBNDNODES takes four arrays of integers and an integer N as paramaters.
* The four arrays correspond to the North, East, South, and West boundary node
* The subroutine inputs the index of the ith boundary node in the ith position
* of the appropriate array.

        SUBROUTINE INDBNDNODES (AN,AE,AS,AW,N,MAXDIM)

        INTEGER N, MAXDIM
        INTEGER AN(MAXDIM),AE(MAXDIM),AS(MAXDIM),AW(MAXDIM)

        DO 290 I = 1, N
          AN (N + 1 - I) = I
          AE (N + 1 - I) = N + I*N + 2*I
          AS (I) = N**2 + 3*N + I
          AW (I) = I*N + 2*I - 1
290     CONTINUE
        END

       SUBROUTINE OUTPUT(N,P,EXPS,M,MU,GAMMA,STDEVS)
       INTEGER N, M
       REAL EXPS(101),P(101),STDEVS(101),MU,GAMMA
       PRINT *
       PRINT *, 'The expectations of the conductances'
       PRINT *, ' will be written to the file '
       PRINT *, '2Dexps.dat.'
       PRINT *, 'in the following format:'
       PRINT *, '  the first row has the value of N'
       PRINT *, '  the second row has the value of M'
       PRINT *,'  the third row has the value of MU'
       PRINT *,'  the fourth row has the value of GAMMA'
       PRINT *,'  the next 101 rows contain the probabilities'
```

```
      PRINT *,'   0.00-1.00'
      PRINT *,'  the next 101 rows contain the corresponding'
      PRINT *,'  average values of the computed conductances'
      PRINT *,'  the final 101 rows contain the standard deviations'
      PRINT *,'  for the calculated conductances'
      PRINT *,'  There is one standard deviation for each probability'
      PRINT *

      OPEN (1,FILE='2Dexps.dat',ERR=999)
      WRITE(1,*) N
      WRITE(1,*) M
      WRITE(1,'(D16.8)') MU
      WRITE(1,'(D16.8)') GAMMA
      DO 300 I=1, 101
        WRITE (1,'(D16.8)') P(I)
300    CONTINUE
      DO 310 I = 1, 101
        WRITE (1, '(D16.8)') EXPS(I)
310   CONTINUE
      DO 320 I = 1, 101
        WRITE (1, '(D16.8)') STDEVS(I)
320   CONTINUE
      RETURN
999   STOP 'Error in writing to 2Dexps.dat'
      END
```

```
inverse _ function(x,y,smean,m,sigma,d) {

# x is the vector of probabilities
# y is the vector of sample mean conductances
# smean is the sample mean calculated by making m measurements of
# effective conductances and averaging them
# m is the sample size
# sigma is the sample standard deviation
# d is the dimension of the network

# Assuming that the data is normally distributed,
# this function returns a 95% confidence interval about the proportion p
# of materials, one having conductance 1 and the other having conductance 0.

# Assign lowerlimit and upperlimit the lower and upper bounds of the 95%
# confidence interval about the true mean conductance.

lowerlimit _ smean - (1.96*sigma)/sqrt(m)
upperlimit _ smean + (1.96*sigma)/sqrt(m)


l _ length(y)
if (d == 2) k _ 1001
else k _ 331
hors _ rep(0,k)
vers _ rep(0,k)


# Fill the vector hors with the probabilities .000,.001,.002,...,.998,
# .999,1.000 if d=2, and .01,.013,.016,.019,.022,...,.997,1.000 if d=3

if (d == 2) hors _ c(0:1000)/1000
else hors _ seq(.01:1,by=.003)

# Fill the vector vers with interpolated sample mean conductances; i.e.
# between any two values of y insert 9 values equidistant from each other
# Hence vers contains the values of y with added values
```

```
for (i in 1:(l-1)) {
for (j in 1:10) {
vers[j+(i-1)*10] _ y[i+1] - (j/10)*(y[i+1] - y[i])
}
}
vers[(l-1)*10+1] _ y[1]

# Now find the index upperindex which indicates the element of vers closest
# to upperlimit and find the index lowerindex which indicates the element
# of vers closest to lowerlimit

upperdists _ rep(0,k)
lowerdists _ rep(0,k)
upperminimum _ 10
lowerminimum _ 10

upperindex _ 0
lowerindex _ 0

for (i in 1:k) {

upperdists[i] _ abs(upperlimit-vers[i])
lowerdists[i] _ abs(lowerlimit-vers[i])

if (upperdists[i] < upperminimum) {
upperminimum _ upperdists[i]
upperindex _ i}

if (lowerdists[i] < lowerminimum) {
lowerminimum _ lowerdists[i]
lowerindex _ i}

}


# Now hors[lowerindex] is the lower confidence limit for the true probability
# and hors[upperindex] is the upper confidence limit for the true probability

return(hors[lowerindex],hors[upperindex])

}
```