Algorithm to extract the spanning clusters and calculate conductivity in strip geometries

F. Babalievski
Höchstleistungsrechenzentrum at the Kernforschungsanlage Jülich, D-52425 Jülich, Germany and Institute of General and Inorganic Chemistry, 1113 Sofia, Bulgaria
(Received 9 May 1994)

I present an improved algorithm to solve the random resistor problem using a transfer-matrix technique. Preconditioning by spanning cluster extraction both reduces the size of the matrix and yields faster execution times when compared to previous algorithms.

PACS number(s): 02.70.−c, 64.60.Ak, 71.30.+h, 05.70.Jk

INTRODUCTION

The method most frequently used for solving numerically the random resistor network (RRN) problem has changed over time surprisingly often: relaxation methods for solving Kirchhoff's equations were adopted in the 1970s, while the early 1980s were the time of the random walk method; then the transfer-matrix (TM) approach [1] came into fashion; and next the node-elimination method came forth in the 1990s [2, 3]. A "Fourier acceleration" method was also proposed in mid 1980s [4]. Renewed interest in direct methods to solve the set of Kirchhoff equations arose after the paper of Edwards *et al.* had been published in 1988 [5]: the standard algebraic multigrid (AMG) method, generally used for solving large linear sparse systems, was applied. In a recent paper [6] the standard Kirchhoff set was reduced by a Green's function formulation of Kirchhoff's laws.

The random walk method is probably the worst among the methods listed above. Although in some applications the random walk method could be more suitable than the others, the main reason for its frequent use appears to be the nice exposition given in Stauffer's famous introductory book [7]. This method faces the same problem as many iterative methods for solving Kirchhoff's equations: its performance decreases rapidly at the critical region of a metal-insulator-like phase transition—a so-called critical slowing down (CSD). Random walkers diffuse anomalously slow at criticality ($p = p_c$); hence the diffusion constant (i.e., the conductivity) estimations require more computer time at $p = p_c$ than for $p > p_c$. In the same way network size scaling at criticality leads to a faster increase of numerical efforts than the number of resistors involved. The origin of the CSD is not so transparent when Kirchhoff's equations are solved iteratively. In that case the CSD amounts to increasing the number of iterations needed to reach a certain precision. Probably the CSD stems from the fractal geometry of the resistor network. Such a geometry leads to a multifractal distribution of voltage drops [8] across the net (if an external voltage is applied) and in this way reduces the speed of convergence in iterative solutions.

The AMG, the transfer-matrix, and the node-elimination methods are free of the CSD in a sense specific for each method. The speculation that the AMG method eliminates the critical slowing down completely (or almost completely) relies mostly on numerical data [5], which are far from exhaustive. The node-elimination method calculates directly (without any voltage evaluations) the network conductance by applying consecutively the star-triangle transformation in a way to reduce the number of sites in the network until one resistor is left. The computational effort is proportional to the number of resistors, so it is faster at $p = p_c$ than for any $p > p_c$. In a similar manner the TM approach is faster at the critical point for a given system size, but here computations scale with size in a nontrivial way, which will be discussed further. It is important to note that a modification of the TM approach in order to evaluate the voltage drop distribution is possible [9]; such modification is impossible within the node-elimination method.

For all three methods, the "preconditioning" of the system by extracting the connected (spanning cluster) or biconnected (percolation backbone) component could significantly improve the method performance. For node-elimination and AMG approaches such an extraction could be easily implemented.

In this paper I am concentrating on the TM approach, presenting a modified algorithm, that allows preconditioning by extraction of spanning clusters. A conceptually important feature of any TM approach is that one does not have to consider the entire system (or its states) at a time in order to calculate its physical properties. Typically, one only requires information about state n in order to proceed to state $n+1$ and subsequently discards the information about state n. In contrast, the known ways [5, 10–12] to extract the backbone requires that the percolation structure is kept in computer memory in its entirety.

I present a TM algorithm that is improved in comparison to the previous TM formulations in two ways. It has inherited the important feature of "voltage-source bookkeeping" from an earlier modification of the "canonical" TM approach made by the author [13] for application to quasicrystalline and random lattices. This feature makes possible a better utilization of the dilute structure of the random networks. Second, the system is preconditioned by spanning clusters extraction. The specific method of extraction reduces significantly the memory requirements, which otherwise are very restrictive.

I. THE TRANSFER-MATRIX APPROACH

The TM approach to the numerical solution of the RRN problem was presented first by Derrida and Vannimenus in 1982 [1] and has been elaborated subsequently by several groups [15–17]. Characteristic for the TM approach in two dimensions (2D) is the use of infinitely long strips of finite width L cut from the resistor network and, analogously, in 3D the use of "bars." The similarity to the transfer-matrix method of the statistical mechanics of spin systems consists of the introduction of a matrix $A(M)$, which represents the properties (in the RRN problem the conductivities; see below) of the semi-infinite strip between $-\infty$ and strip slice M. As, e.g., in [1], a strip slice of a resistor network on the square lattice may consist of the vertical resistors in column M and the horizontal resistors that connect columns M and $M-1$. Knowledge of the conductivity matrix $A(M)$ and the resistor configuration in slice $M+1$ is sufficient to calculate $A(M+1)$ for the next slice. Iteration for all subsequent slices finally obtains the conductivity of the whole strip. In the case of a resistor network of unit resistors and insulators, the long edges of the strip are thought of as electrodes and the resistors in the upper and the lower layers are taken to have zero resistance.

If the resistor network has a fractal structure, its conductance tends to zero as the system size increases — in analogy to its mass density decreasing to zero. More quantitatively, the infinite spanning cluster at the percolation threshold p_c is a fractal for which finite-size scaling theory shows [18] that its conductivity should scale with the system size L as $L^{-t/\nu}$, where ν is the percolation correlation-length exponent and t is the percolation transport exponent.

The TM approach has been used first for obtaining precise estimates of t for percolation on the square and the cubic lattices [15, 17]. In Refs. [15, 17] the matrix $A(M)$ is updated after the addition of every single resistor (the program is published in [16]) instead of using matrix equations as in [1]. Thus the calculations are simplified and accelerated.

In order to define the matrix $A(M)$, we attach voltage sources to the open ends of the resistors at the right end column of the growing semi-infinite strip. The matrix $A(M)$ is defined by attaching to the sites of the current right end column of the semi-infinite strip voltage sources V_i, where i labels the row position in the strip and thus assumes values from 1 to L, the width of the strip. Since the voltage-current relations in the network are linear, the current from any selected sources, say, source j, is a linear function of the voltages V_i,

$$I_j = \sum_i^L A_{ji}(M) V_i. \tag{1}$$

The relation (1) defines the matrix elements $A_{ij}(M)$ of the $A(M)$. From now on I will suppress the argument M when no confusion can arise.

When a *horizontal* resistor R is added to row k the matrix A changes to A' with matrix elements

$$A'_{ij} = A_{ij} - \frac{A_{ik}A_{kj}R}{1 + A_{kk}R}. \tag{2}$$

For infinite R, a case that we encounter in insolator-resistor mixtures, Eq. (2) simplifies to

$$A'_{ij} = A_{ij} - \frac{A_{ik}A_{kj}}{A_{kk}}. \tag{3}$$

When we add a *vertical* resistor between two adjacent sites k and l of a new column four matrix elements change,

$$\begin{aligned} A'_{kl} &= A_{kl} - 1/R, \\ A'_{lk} &= A_{lk} - 1/R, \\ A'_{kk} &= A_{kk} + 1/R, \\ A'_{ll} &= A_{ll} + 1/R. \end{aligned} \tag{4}$$

From Eq. (1) it is clear that, in the limit $M \longrightarrow \infty$, e.g., the difference $A(M)_{LL} - A(0)_{LL}$ tends to the transverse conductance of the strip of width L. From an analysis of the conductivity scaling of strips with different L one obtains the conductivity scaling exponent. For the percolation cluster at p_c, this exponent equals the ratio t/ν.

The advantages of the TM approach have been described in the pioneering works [1] and [16]. Here, I would like to point out to the reader its main drawback: the size of the matrix A and the computational effort grow very fast with the strip width L.

In particular, the size of the matrix grows as L^2 and L^4 for 2D and 3D, respectively. If we consider a site percolation model in 2D, then the addition of every new column leads to adding an average of $p^2(L-1)$ horizontal and $p^2(L-2)+2p$ vertical resistors. Taking the size of the matrix A into account, we find that Eq. (2) is applied proportionally $\propto L^3$ times whereas only operations proportional to L are required for Eqs. (4). Thus it is clear that for widths larger than $10-15$ lattice spacings more than 90% of the time is spent on calculating the relations (2). In 3D the situation is even worse: the upper bound for the computational efforts scales as L^6. But, in fact, this bound is overestimated: Ref. [16] points out that the computational effort scales as L^4 due to the fact that the matrix A is sparse, i.e., most of its elements equal zero. In the next section, I will describe a modification of the TM approach that overcomes these problems in part.

II. MODIFIED ALGORITHM

The site-percolation case will be considered without loss of generality. The voltage-source bookkeeping procedure is described in Sec. II A. Second (Sec. II B) comes the method for extracting the spanning clusters and finally (Sec. III C) I present the main steps in the complete algorithm.

A. Conductivity calculations

Let us reconsider Eq. (1) in the case of a general resistor network. Let some network nodes of an arbitrary resistor network be connected to external voltage sources

V_i. Then I_j [Eq. (1)] is the current from source j and the absolute values of the off-diagonal elements of matrix $A_{ij}, i \neq j$, represent the conductances between voltage sources i and j. The diagonal elements give the conductance between the respective source and the "ground" — the other sources set to zero voltage. In fact, we can likewise interpret the matrix elements A_{ij} introduced in Eq. (1) in Sec. I. The difference is that the number of sources in the present case does not strictly depend on the strip width and their connection to the right-hand end is not mandatory (see Fig. 1 in Ref. [13]). Equations (2)–(4) still apply in the present general case if we allow in the general geometry the "addition of a vertical resistor" as the connection with a resistor of two sites with voltage sources attached to them and the "addition of a horizontal resistor" as the insertion of a resistor between a site and the voltage source previously attached to this site. In other words, adding a horizontal resistor creates a new site and moves the voltage source to it.

Based on these general concepts, we may formulate as algorithm the conductivity calculations Eqs. (2)–(4) for the resistor strip case and the construction of the strip. The algorithm has three main steps.

(i) Add a new site to the right-hand end of the already existing strip and attach a voltage source to this site.

(ii) Find the neighbors of this site among the sites already present and connect them with resistors. Update the matrix using Eq. (4). The algorithm should ensure that the neighbors have their own voltage sources attached.

(iii) If a site with its attached voltage source is located within the bulk of the growing strip then detach the source to free it for subsequent attachment to a new site on the growing edge of the strip. To this end, we (a) insert an insulating "resistor" between the network site and the previously attached source. Then we (b) update the TM according to Eq. (3). The information about the voltage source index is kept on a stack, ready to be (c) used again when a new site is added to the right-hand end of the strip. We have done nothing more than add a bridge of infinite resistance between two points of the network, which does not alter its conductivity properties, and move a voltage source.

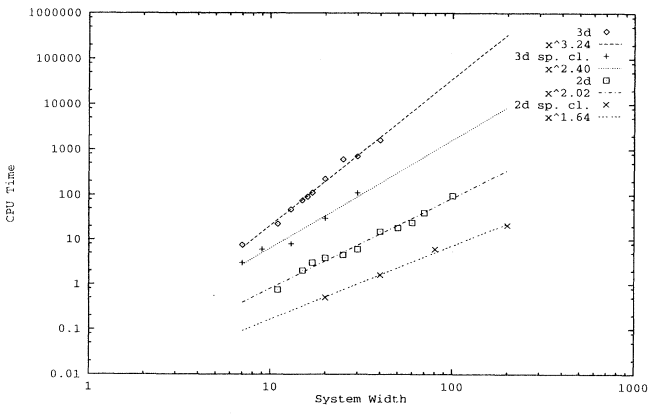

FIG. 1. Comparison between the performance with and without extracting spanning clusters. (Time units are Sun SPARC 10 computer workstation CPU seconds.) $x\hat{}(\ldots)$ is the power law fitted to the respective data set.

Since a regular lattice structure of the resistor bonds is not a prerequisite for the conductivity calculation outlined above, an algorithm based on steps (i)–(iii) is useful for calculation of percolative conductivities of *quasicrystalline* and *random* lattices [13, 14].

Since we have to only update the TM for lattice sites that are actually connected to the strip by resistors of finite resistance and since we always apply the simpler Eq. (3) instead of Eq. (2), the outlined procedure is already faster than the standard algorithm [16].

The matrix size, i.e., the number of matrix elements, instead of being L^2 is only approximately equal to p^2L^2 for the square lattice and approximately equal to p^2L^4 instead of L^4 for the cubic lattice. The scaling with L of the matrix size is not altered, but the way of handling the voltage source numbers allows for a significantly smaller prefactor and (more important) it facilitates the system preconditioning, which does lead to an improvement of memory and performance scaling.

B. Spanning cluster extraction

We achieve an improvement of the scaling of the matrix size as a function of L by the extraction of spanning clusters. The spanning clusters are defined as the percolating clusters that connect the top and the bottom edges of the strip or, respectively, the bottom and the top faces of the bar in 3D. At p_c the spanning clusters in strip geometries represent the incipient infinite percolation cluster. Its fractal dimension d_f is 91/48 in 2D and around 2.5 in 3D [7]. If only voltage sources connected to the spanning clusters contribute to the matrix size, then this size should scale as $L^{2(d_f-1)}$, where the exponent $d_f - 1$ reflects the system width dependence of the scaling of the spanning cluster's sites found in a $(d-1)$-dimensional cut [19]. Thus, in 2D the matrix size scales as $L^{1.79}$ instead of L^2. In 3D the number of the matrix elements is proportional to L^3 rather than L^4.

How does one extract the spanning clusters in strip geometries? Several general algorithms exist to solve this problem [10–12]. However, they all require that the percolation structure has been created beforehand and is stored in its entirety leading to large computer memory requirements (e.g., for a bar $10^5 \times 100 \times 100$ one has to consider 10^9 sites and 10^8 matrix elements).

I now present an algorithm that partly resolves this problem, which would otherwise limit the strip length. The method for extracting the spanning cluster is based on the Hoshen-Kopelman algorithm [20, 7] for cluster counting. As is well known, this algorithm requires only consecutive $(d-1)$-dimensional cuts of the lattice to be kept during its lattice "sweeping." Cluster information is stored in one one-dimensional array — the array of cluster sizes and pointers, sometimes denoted as the array of "labels of labels" (LOL) [7]. The index into this array represents the cluster labels and its elements are either "cluster roots" — then containing the size of a specific cluster — or pointers to these cluster roots — i.e., negative numbers whose absolute value is equal to

the index of the array element corresponding to the cluster root. Moreover, the cluster root element may be used to store other information about its cluster, e.g., whether the cluster touches the upper and/or the lower layer of the strip.

Running the Hoshen-Kopelman algorithm requires that the percolation structure be scanned twice in order to extract the spanning clusters. These two runs are required because if we reach a site during the first run we cannot decide whether this site's cluster will eventually turn out to span. To avoid storage of the entire cluster in memory, we perform the second sweep based on a repetition of the pseudorandom number sequence that created the first sweep cluster. During the second sweep the LOL array is examined to decide which cluster a site belongs to and whether this cluster spans. Only sites belonging to spanning clusters enter the conductivity calculations.

Thus, instead of storing the percolation cluster structure itself, we only store the LOL array. The key question for the proposed algorithm is the size of the LOL array that has to be retained in memory between the two Hoshen-Kopelman sweeps. To keep its size small I apply a procedure to recycle unused labels [21, 22]. The size of the resulting LOL arrays turns out to be less than 0.5% of the memory required to retain the whole cluster structure in memory.

C. The complete algorithm

The full algorithm, including both the spanning clusters extraction and the conductivity-calculation procedures, may be summarized as follows.

(i) Scan the random structure with the Hoshen-Kopelman procedure constructing the LOL array by a label recycling technique. After the sweep, keep the LOL array in memory.

(ii) Repeat the scan using *the same pseudorandom number sequence.* After creation of a new site, decide by comparing the new and the stored LOL array whether this site belongs to a spanning cluster. If it does then the site enters the TM conductivity calculations. These proceed according to step (i)–(iii) as Sec. II A.

(iii) When the second scan terminates calculate the transverse conductance per unit length as

$$\Sigma_L = \frac{A(M)_{LL} - A(M_0)_{LL}}{M - M_0}, \tag{5}$$

where I have used $M_0 = M/5$ to reduce boundary effects.

III. PERFORMANCE SCALING RESULTS

I have developed the algorithm described in the preceding section in conjunction with a study [23] on the conductivity of several distinct percolation models and it has not only been applied to standard percolation.

I compare the TM algorithm proposed in this paper (the modified algorithm) to the previously published [16, 13] TM algorithms (the standard algorithms). As a standard algorithm I used mostly the algorithm proposed in [13] which was described in Sec II A. The code published in [16] was run with a technical improvement (zero elements check in the most-inner loop) only to be seen that the performance scaling is the same for both "standards." It is worth to note that the performance scaling $\sim L^4$ in 3D, reported in [16], is an overestimation probably due to that technical item.

In Fig. 1 I display the amount of computer time required for the conductivity calculations by the modified and the standard algorithm on two different percolation models, namely, ordinary site percolation and one-step bootstrap percolation [23]. In the bootstrap percolation model [24] one generates a site configuration in several steps. First, one randomly occupies a specific small fraction of the lattice sites. Subsequently, one determines all empty lattice sites with at least two occupied neighbors and occupies these empty sites as well. The steps are repeated until no empty sites with two occupied neighbors remain. If such procedure stops after its first step I call it one-step bootstrap percolation. The percolation-transport and correlation-length exponents of the one-step bootstrap percolation model almost equal those of ordinary site percolation [23]. The computer time for these two models, when running the standard algorithm in 3D, is scaled in the same way—even with the same prefactor—so the averaged results are given on one curve (3D) in Fig. 1. This coincidence encouraged using the data available [23] in 2D for the comparison in the next paragraph. (In 2D the two algorithms were applied to different models: the standard algorithm to the bootstrap model and the modified to ordinary percolation.)

As expected from the arguments in Sec. II, the modified algorithm displays the better scaling properties throughout. In three-dimensional site percolation the computer time scales as $\sim L^{3.24}$ for the standard algorithm, whereas the modification needs time proportional to $L^{2.40}$ only. Similarly, in 2D we observe that the standard requires time approximately $L^{2.02}$, whereas approximately $L^{1.64}$ suffices for the modification. One can see that these values are appreciably smaller than the respective upper bonds given in Sec. I.

The errors in the above values, given by the least-squares fitting procedure, were smaller than the uncertainty coming from the eventual correction-to-scaling terms. A more careful analysis is needed, but a reasonable value for the error should be within 0.1–0.2. All the tests were made at the percolation threshold for the respective model. The strip (bar) length was of the order 10^6 in two dimensions and 10^5 in 3D. Several strips were calculated for each width and model. If one has to compare with statistics accumulated by means of another method on square (cubic) samples, a strip $10^6 \times 100$ may correspond to several hundred runs on, say, a $100\sqrt{2} \times 100\sqrt{2}$ sample. [In Ref.[25] it was found numerically that at p_c the average length of the spanning clusters is around $2.0L$ and their number is $\cong (3/8)(M/L)$, where M is the strip length.]

The modified model was applied as well in studying properties of some "percolation-generated" fractals. After the extraction of spanning clusters at the percolation thresholds one adds new sites on the cluster perimeter in order for some aerogel structures to be modeled [26, 27].

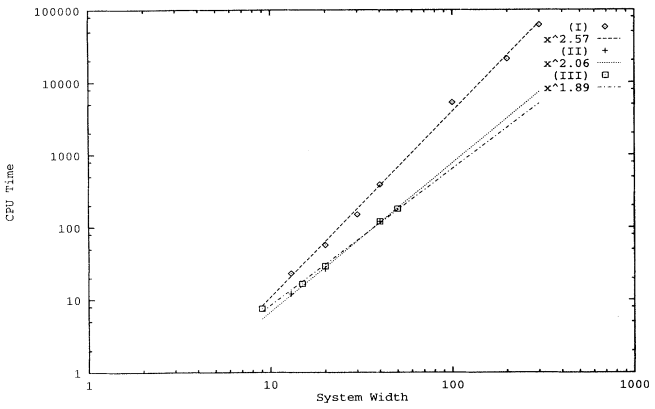

FIG. 2. Performance scaling for modified percolation models where, after extraction of spanning clusters, additional loops were closed (I) by addition of new sites on the cluster perimeter (II) by adding sites as in (I) but with a lower density, and (III) by increasing the connectivity range to second neighbors. (Time units are Sun SPARC 10 workstation CPU seconds.)

In Fig. 2 I display timing results for such fractals (cf. [23]). The data sets (I) and (II) correspond to two ensembles with a higher (I) and a lower (II) fraction of additionally occupied perimeter sites of the spanning clusters. The set (III) has been taken for a model in which one considers second nearest neighbors in the spanning clusters as connected. As can be seen, depending on the model, the computer time scaling may vary significantly.

CONCLUSION

The modified TM algorithm proposed in this work reduces significantly the computational efforts required for obtaining the conductivity scaling for fractal structures in 2D and especially in 3D. In contrast to the $L^{3.2}$ computer time requirements of conventional TM algorithms in 3D, the time requirements of the algorithm proposed in this work scales approximately as $L^{2.4}$ for percolation clusters at p_c. Extracting the percolation cluster backbone instead of spanning clusters only would ensure further improvement of the performance.

Probably the main disadvantage of the modification of the TM approach described here is the complexity of the algorithm. Therefore, I have made the program publicly available [22].

ACKNOWLEDGMENTS

I would like to thank H. J. Herrmann and Many Particle Group at the HLRZ for their hospitality. I am grateful to W. Vermöhlen for his help during my stay at the HLRZ. S. Melin and H. Puhl helped me use the computational facilities at the HLRZ. I acknowledge H. J. Herrmann for comments on the preliminary version of the manuscript. Special thanks go to S. Schwarzer, who greatly helped me in improving the final text. This work was supported by the European Communities commission on science Grant No. CIPA3511PL920176 and by Bulgarian NSF Grant No. F-119/91.

[1] B. Derrida and J. Vannimenus, J. Phys. A **15**, L557 (1982).
[2] D. Gingold and C. Lobb, Phys. Rev. B **42**, 8220 (1990).
[3] D. Frank, C. Lobb, Phys. Rev. B **37**, 302 (1988).
[4] G. G. Batrouni, A. Hansen, and M. Nelkin, Phys. Rev. Lett. **57**, 1336 (1986); G. G. Batrouni, and A. Hansen, J. Stat. Phys. **52**, 747 (1988).
[5] R. Edwards, J. Goodman, and A. Sokal, Phys. Rev. Lett. **61**, 1333 (1988).
[6] K. Wu and R. M. Bradley, Phys. Rev. E **49**, 1712 (1994).
[7] D. Stauffer, *Introduction to Percolation Theory* (Taylor & Francis, London, 1985); D. Stauffer and A. Aharony, *Introduction to Percolation Theory*, 2nd ed. (Taylor & Francis, London, 1992).
[8] S. Roux and A. Hansen, in *Disorder and Fracture*, edited by S. Roux, A. Hansen, and E. Guyon (Plenum Press, New York, 1990).
[9] D. J. Bergman, E. Duering, and M. Murat, J. Stat. Phys. **58**, 1 (1990).
[10] R. Tarjan, SIAM J. Comput. **1**, 146 (1972).
[11] H. J. Herrmann, D. Hong, and H.-E. Stanley, J. Phys. A **17**, L261 (1984).
[12] S. Roux and A. Hansen, J. Phys. A **20**, L1281 (1987).
[13] F. Babalievski, Z. Phys. B **84**, 429 (1991).
[14] F. Babalievski, Physica A **182**, 325 (1992).
[15] B. Derrida, D. Stauffer, H. J. Herrmann, and J. Vannimenus, J. Phys. (Paris) **44**, L701 (1983).
[16] B. Derrida, J. Zabolitzky, J. Vannimenus, and D. Stauffer, J. Stat. Phys. **36**, 31 (1984).
[17] J. Normand, H. J. Herrmann, and M. Hajjar, J. Stat. Phys. **52**, 441 (1988).
[18] M. P. Nightingale, Physica A **83**, 561 (1976); *Finite-Size Scaling*, edited by J. L. Cardy (North-Holland, Amsterdam, 1988).
[19] T. Vicsek, *Fractal Growth Phenomena* (World Scientific, Singapore, 1989).
[20] J. Hoshen and R. Kopelman, Phys. Rev. B **14**, 3428 (1976).
[21] *Applications of the Monte Carlo Method in Statistical Physics*, edited by K. Binder (Springer, Heidelberg, 1987), Chap. 8.
[22] The program is available via e-mail to banchem@bgearn.bitnet, specifying "TMcond" in the subject line.
[23] F. Babalievski, Physica A **211**, 1 (1994).
[24] J. Adler, Physica A **171**, 453 (1991).
[25] R. A. Monetti, E. V. Albano, Z. Phys. B **90**, 351 (1993).
[26] E. Stoll and E. Courtens, Z. Phys. B **81**, 1 (1990).
[27] H. Nakanishi, Physica A **196**, 33 (1993).