J. Phys. A: Math. Gen. **33** (2000) L399–L407. Printed in the UK PII: S0305-4470(00)12142-0

LETTER TO THE EDITOR

Efficient measurement of the percolation threshold for fully penetrable discs

J Quintanilla†, S Torquato‡‖ and R M Ziff§

† Department of Mathematics, University of North Texas, Denton, TX 76203, USA
‡ Princeton Materials Institute and Department of Chemistry, Princeton University, Princeton, NJ 08544, USA
§ Department of Chemical Engineering, University of Michigan, Ann Arbor, MI 48109-2136, USA

E-mail: johnq@unt.edu, torquato@matter.princeton.edu and rziff@umich.edu

Received 28 February 2000, in final form 29 August 2000

Abstract. We study the percolation threshold for fully penetrable discs by measuring the average location of the frontier for a statistically inhomogeneous distribution of fully penetrable discs. We use two different algorithms to efficiently simulate the frontier, including the continuum analogue of an algorithm previously used for gradient percolation on a square lattice. We find that $\phi_c = 0.676\,339 \pm 0.000\,004$, thus providing an extra significant digit of accuracy to this constant.

(Some figures in this article appear in colour in the electronic version; see www.iop.org)

1. Introduction

In a previous paper [1], the first two authors studied percolation phenomena in a statistically inhomogeneous system of fully penetrable discs in the unit square. This model is a two-phase system consisting of an inhomogeneous distribution of fully penetrable discs whose particle volume fraction obeys some specified density gradient $\phi(x)$. Cluster formation naturally arises: some particles will be connected to the region of high volume fraction, forming a *percolating cluster*. The authors studied the fractal behaviour of the *frontier*, or the edge of this percolating cluster. They also measured the average location of the frontier, which is conjectured to be related to the percolation threshold for a system of statistically homogeneous discs. This type of analysis was first introduced by Rosso, Gouyet and Sapoval for lattice percolation [2,3] and by Rosso [4] for continuum percolation.

To simulate a system of inhomogeneous fully penetrable discs in the unit square, discs of radius R are centred on the points of an underlying inhomogeneous Poisson field [5]. For computational efficiency, we restrict the volume fraction to lie between two prescribed values, that is, $\phi_{\min} \leqslant \phi(x) \leqslant \phi_{\max}$. We choose the left edge $x = 0$ to correspond to a particle volume fraction of $\phi_{\min}$, while the volume fraction along the right-hand edge $x = 1$ is $\phi_{\max}$. These values obey the condition

$$\frac{\phi_{\min} + \phi_{\max}}{2} = 0.676\,37 \tag{1}$$

‖ Corresponding author.

so that, with a linear density gradient $\phi(x)$, the volume fraction along the vertical line $x = 1/2$ corresponds to the previous estimate of the percolation threshold [1], in which the estimated error was 5×10^{-5}.

The *effective system length* is defined by

$$\ell = \frac{L}{R(\phi_{\max} - \phi_{\min})}. \tag{2}$$

This dimensionless parameter ℓ is the inverse of the gradient of ϕ, measured in terms of the disc radius R and the length $L = 1$ of the unit square. Intuitively, ℓ is twice the number of adjacent discs that would span the system if the edges of the unit square corresponded to $\phi_{\min} = 0$ and $\phi_{\max} = 1$. For computational efficiency, these limits are not actually chosen in practice, and so ℓ allows us to compare systems with different values of R, $\phi_{\min}$ and $\phi_{\max}$. The effective system length is analogous to the lattice size of a lattice system.

The number density $\rho(x)$ is chosen as

$$\rho(x) = \frac{-\ln(1 - [x\phi_{\max} + (1-x)\phi_{\min}])}{\pi R^2}. \tag{3}$$

Assuming that R is small, $\rho(x)$ will be nearly constant in the vicinity of x. Therefore, the density profile $\phi(x)$ will be very close to [6]

$$\phi(x) = 1 - \mathrm{e}^{-\rho(x)\pi R^2} \tag{4}$$

and thus the profile will be very nearly linear in x.

In the previous paper [1], to simulate the frontier, the complete inhomogeneous distribution of discs within the unit square was generated and then the percolating cluster of connected discs was found. While this direct approach works, it is computationally inefficient since many discs are generated that are not on the frontier, thus wasting both execution time and memory capacity. To study the frontier more efficiently, one must first of all minimize the number of unnecessary discs generated.

In this letter, we introduce two new algorithms for simulating the frontier. One of these algorithms is the continuum analogue of a method previously used for measuring the percolation threshold for two-dimensional lattice percolation [7]. Unlike the previous paper [1,4], both of these new algorithms generate the frontier without generating the complete system within the entire unit square. The improved efficiency of these algorithms permit consideration of larger effective system lengths and hence a more precise evaluation of the percolation threshold.

2. Algorithms

To make the simulation of the frontier more efficient, we do not generate the locations of all discs in the unit square. Instead, we dynamically generate discs by first dividing the unit square into an $N \times N$ grid of subsquares. Only the discs within the necessary subsquares are generated. The side length of the subsquares (which determines N) is typically chosen to be one or two disc diameters, and therefore a disc in a subsquare cannot interact with the discs in nonadjacent subsquares.

If discs are to be generated within a given subsquare $[(i-1)/N, i/N] \times [(j-1)/N, j/N]$, first the potential number of discs M within the subsquare is determined from a Poisson distribution with mean $\rho_* = \rho(i/N)$ [8]. The centres of M discs are then randomly generated according to the uniform distribution on the subsquare. To enforce the linear density gradient, each generated centre (x, y) generated is then kept, independently of the other centres, with probability $\rho(x)/\rho_*$ or deleted with probability $1 - \rho(x)/\rho_*$. (For sufficiently large N, $\rho(x)$

will be nearly constant in the subsquare, and so thinning will only occur rarely.) The resulting point pattern is a realization of an inhomogeneous Poisson field with density function $\rho(x)$ [5]. Discs of radius R are then centred on these points to complete the model.

There are three stages of randomness in this algorithm for every needed subsquare: the Poisson deviate M, the positioning of the discs and the thinning procedure. For this reason, it is imperative to use a high-quality random-number generator with a very long period. In this letter, we primarily used the generator `ran2` of [8], although runs using the four-tap exclusive-or generator R(471, 1586, 6988, 9689) [9] were also performed for confirmation.

Once discs have been generated in a certain subsquare, this part of the realization of the unit square has been specified. Therefore, the subsquare should be marked as generated: these discs should not be erased, and new discs should not be placed into this subsquare. (For computational efficiency, we will relax this condition for the frontier-walk method, as discussed below.)

In lattice percolation, it is possible to simulate the frontier without generating any unnecessary sites [10]. The percolation threshold may then be found using three different methods: the average position of the occupied sites along the frontier, the average position of the vacant sites along the frontier or a weighted average of both [2]. All three methods converge to the percolation threshold; the weighted average method has the fastest rate of convergence.

In contrast, in the continuum model, some discs in the subsquares will not contribute to the construction of the frontier. Furthermore, there is no natural way to define the discs closest to the frontier that are not in the percolating cluster, and so those previous methods of characterizing the frontier on the lattice do not directly extend to the present model. Therefore, we calculate the percolation threshold by averaging the position of the arcs on the frontier; this reduces to a certain integral over the frontier [1].

We describe two methods for efficiently simulating the frontier: the gap-traversal method and the frontier-walk method. In both of these methods, periodic boundary conditions are enforced in the vertical direction; horizontal periodic boundary conditions are obviously inappropriate given the linear density gradient in volume fraction. In the gap-traversal method, the percolation threshold will be estimated after averaging multiple realizations of the frontier. However, in the frontier-walk method, one continuous and arbitrarily long walk along the frontier will be made.

2.1. Gap-traversal method

In this method, an initial disc is first found near the right-hand (highest-density) edge. Discs are progressively generated to the left until the frontier is reached, as follows.

(1) If the chosen disc intersects some other disc (called a neighbour) whose centre has a smaller x-coordinate (i.e. is further to the left), we move to it. This is repeated until we arrive at a disc, centred at (x_1, y_1), which does not have a left-hand neighbour. Since this disc has no left-hand neighbour, its leftmost point $(x_1 - R, y_1)$ must lie on the boundary between the particle and void phases; we must now determine whether this point actually lies on the frontier.

(2) Starting from this leftmost point, we traverse the *boundaries* of the discs until either (i) we come back to this starting point or (ii) the right-hand edge of the unit square is reached. While traversing these boundaries, we record the positions of the discs and the angles which bound the arcs on the boundary. We also keep track of the leftmost disc (x_2, y_2) found and the leftmost arc on this disc. (It is possible for one disc to have more than one arc.)

(3) There are three possible cases.

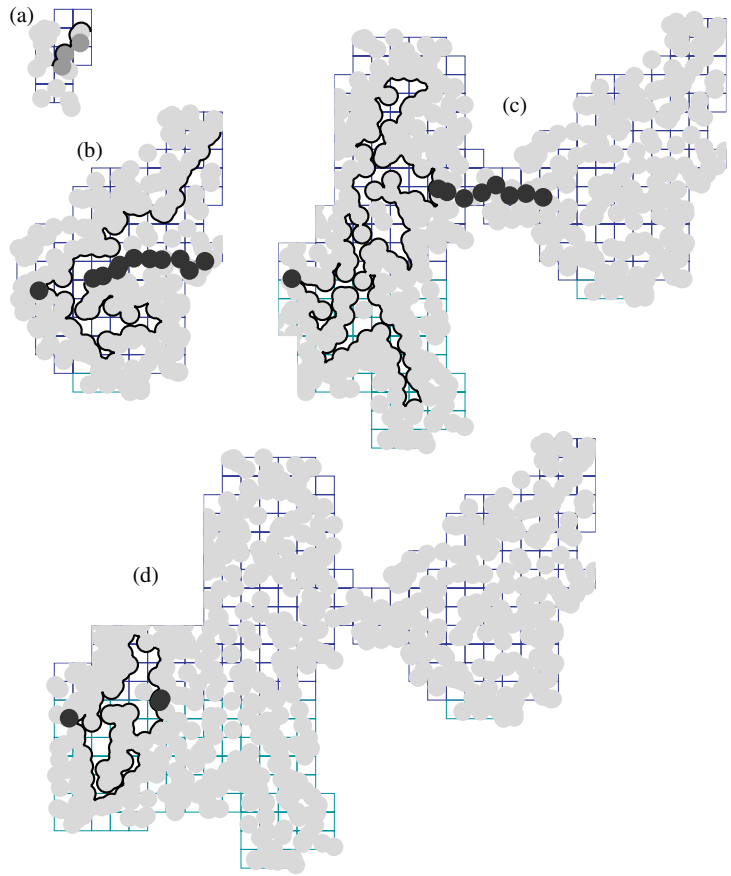

Figure 1. The gap-traversal method of simulating the frontier. The subsquares that have been considered are outlined. (*a*) A dead-end: the leftmost disc also contains the leftmost arc traversed. (*b*) Starting with a disc at the right-hand edge, we move to leftmost neighbours (shown with darkened discs) as far as possible. The boundaries are then traversed until the right-hand edge is reached. These arcs do not contain the leftmost point of the new leftmost disc (darkened to the left), and so the simulation continues. (*c*) A continuation of (*b*), except that we stop traversing the arcs when we return to the original leftmost disc. The previously generated discs from (*b*) are seen in the right-hand half of this figure. (*d*) The simulation continues.

(a) We have reached a *dead end* (see figure 1(*a*)) if $(x_1, y_1) = (x_2, y_2)$. The discs that have been constructed are not actually part of the percolating cluster, and we must restart at some other disc on the right-hand edge.
(b) We have found an *interior void* (see figures 1(*b*)–(*d*)) if $(x_1, y_1) \neq (x_2, y_2)$ but the point $(x_2 - R, y_2)$—the leftmost point of the leftmost disc—is not contained within the leftmost arc. We then iterate steps 1 and 2, moving further to the left in the unit square.
(c) We have *arrived at the frontier* (see figure 2) if $(x_1, y_1) \neq (x_2, y_2)$ and the point $(x_2 - R, y_2)$ is within the leftmost arc of the leftmost disc; this concludes the algorithm.

We see that the arcs generated in figure 1 are in fact the boundaries of interior voids in the percolating cluster of figure 2. We also see that much of the right portion of the unit square is ungenerated. If the full square were generated, nearly all of the discs in this region would

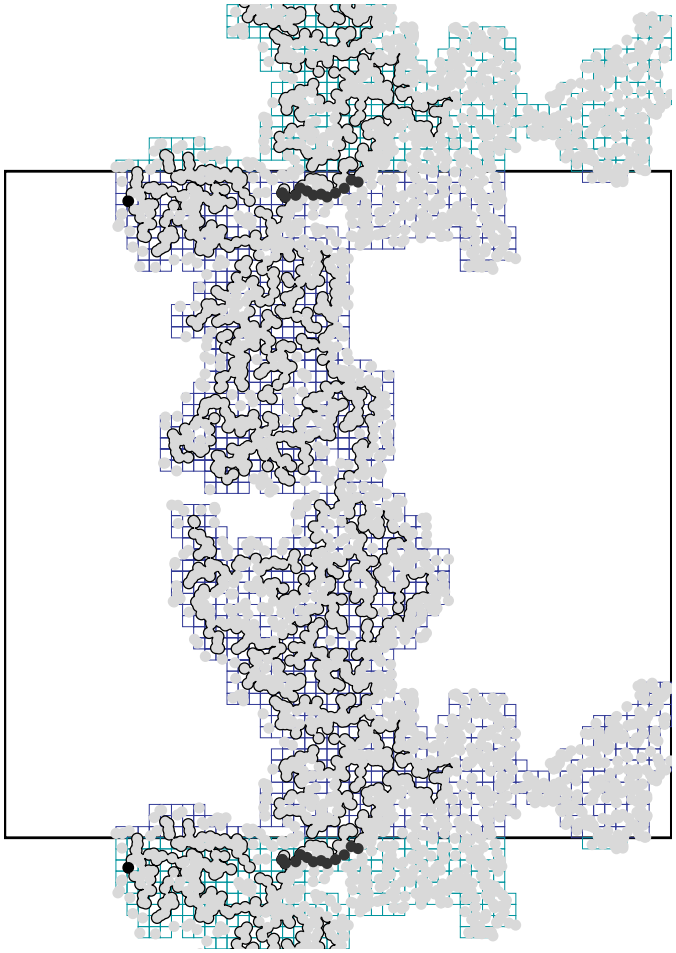

Figure 2. A simulated frontier for $\ell = 400$ with a 60×60 grid of subsquares using the gap-traversal method. The frontier contains the leftmost point of the leftmost disc (in the upper left-hand corner), and so the simulation stops. Periodic boundary conditions in the vertical direction are employed. The previous steps shown in figure 1 can be seen in the lower right-hand corner of the unit square. Only a portion of the unit square was generated in order to generate the frontier.

be part of the percolating cluster. Not explicitly simulating this region saves considerable computational resources, a saving which increases with the effective system length.

With this approach, separate realizations of the frontier are generated. The properties of these realizations must then be averaged to determine the statistics of the frontier.

2.2. Frontier-walk method

The second algorithm used is the continuum analogue of a method used previously for simulating the frontier in lattice gradient percolation [7]. With this method, a frontier of *arbitrary* length is created by cycling through the unit square. Therefore, unlike the gap-traversal method, there is no need to generate separate realizations of the model and then average the frontier statistics of these different realizations.

To start the walk, the right half of the top of the square is initialized with overlapping discs

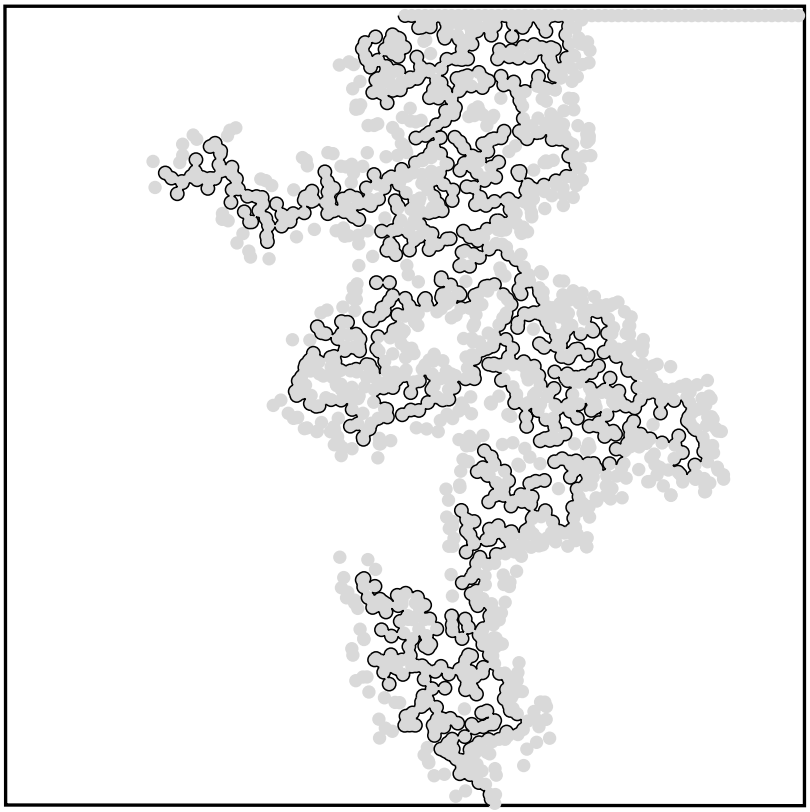

Figure 3. A realization of the first pass of the frontier-walk algorithm for $\ell = 400$. The right-hand half of the top of the square is initialized with overlapping discs with centres a radius apart, and the walk begins at the leftmost point of these initial discs. The walk hits the top of the square several times at the start until it moves downward. The grid of subsquares is not outlined in this figure. The walk may be continued indefinitely by wrapping around to the top of the square after erasing previously generated discs.

with centres a radius apart; see figure 3. (Recall that the density profile is chosen so that the percolation threshold occurs near the line $x = 1/2$.) The walk is started at the leftmost point of these initial discs and initially traverses the lower portion of the discs. After a while, the walk continues in the negative y direction. The frontier walk may be continued indefinitely by carefully resetting the subsquares in the grid. Every time the walk hits a new row in the y direction, the subsquares in that row are reinitialized: the previously generated discs from the earlier pass are eliminated and the subsquares are reset as unvisited. A walk of infinite length may thus be generated on a computer with finite memory. The statistics of the first pass through the unit square were discarded to eliminate any bias from the initialization.

For sufficiently large ℓ, the width σ of the walk is proportional to $\ell^{-3/7}$ [1,3]; in this letter, the constant of proportionality is approximately 0.55. Therefore, to minimize the possibility of the frontier extending beyond the square, the volume fractions $\phi_{\min}$ and $\phi_{\max}$ are chosen so that $\phi_{\max} - \phi_{\min} > 14\sigma$. In our simulations, these boundaries were never hit. We also did not observe the walk wrapping back to subsquares that were previously erased.

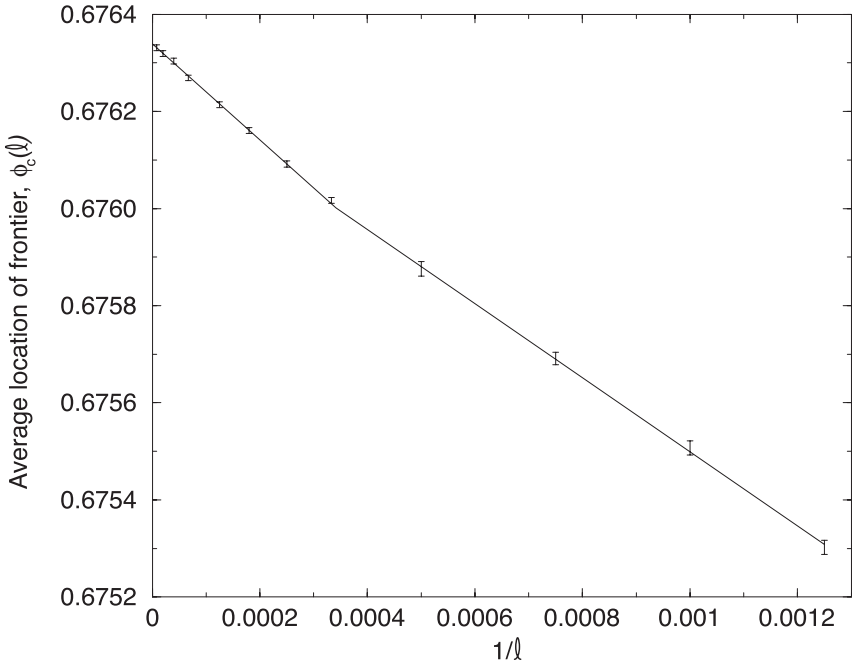

Figure 4. The average location $\phi_c(\ell)$ of the frontier. We see that $\phi_c(\ell)$ is linear only for $\ell \geqslant 4000$.

3. Results

Not surprisingly, the frontier-walk method is computationally superior to the gap-traversal method since no interior void is generated. With the frontier-walk method, systems with $\ell = 125\,000$ can be simulated on a 400 MHz microcomputer with 128 MB of RAM. Consequently, the results reported below are from the frontier-walk method. However, these two different algorithms produced consistent statistics, thus providing an important check of our simulations.

As a further check of our simulations, we compute the fractal exponents for the perimeter P and the width σ of the frontier as a function of ℓ. Expressions for these quantities may be found in [1]. Assuming that the fractal dimension of the percolation hull is $7/4$, as in lattice percolation [3, 11], then we expect that

$$P \propto \ell^{3/7} \qquad \text{and} \qquad \sigma \propto \ell^{-3/7}. \tag{5}$$

Using the above algorithms for systems with $4000 \leqslant \ell \leqslant 125\,000$, we empirically obtain these fractal exponents to three decimal places.

We assume that, as $\ell \to \infty$, the average value $\phi_c(\ell)$ of ϕ along the arcs of the frontier converges to the percolation threshold ϕ_c of homogeneous discs. (The computation of $\phi_c(\ell)$ is detailed in [1] but is called p in this reference.) While this is intuitively reasonable and the numerical evidence on lattices is quite compelling, a formal proof of this convergence has yet to be discovered.

For $3000 \leqslant \ell \leqslant 125\,000$, we calculated $\phi_c(\ell)$ accurate to less than 6×10^{-6}. To obtain this accuracy using a 400 MHz microcomputer, each value of $\phi_c(\ell)$ required 160–350 h of computer time to generate and measure the 1.3×10^{10}—2.0×10^{10} arcs on the frontier; larger

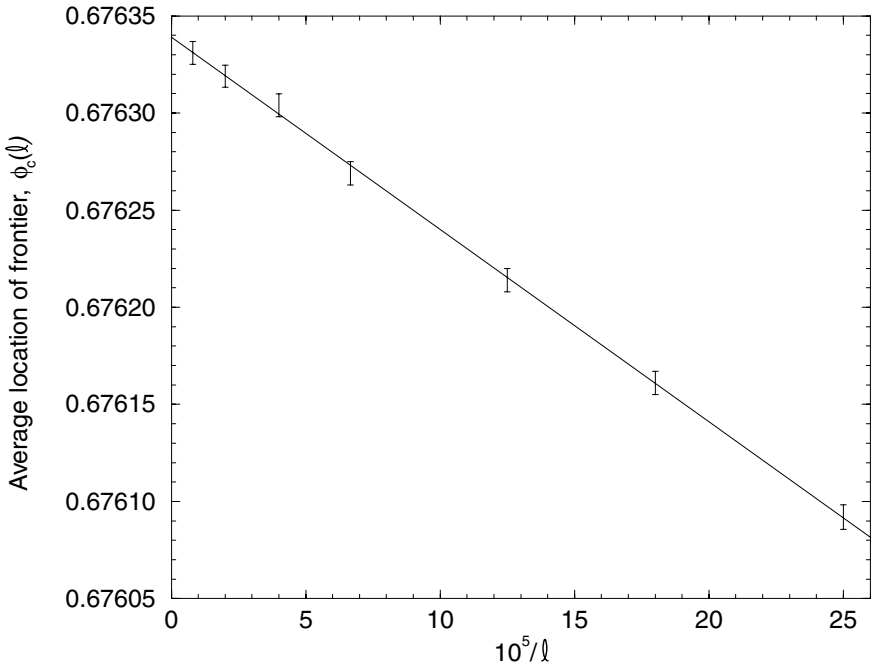

Figure 5. The average location $\phi_c(\ell)$ of the frontier. We see that $\phi_c(\ell)$ is linear for $\ell \geqslant 4000$.

values of ℓ required greater computational effort. We also computed $\phi_c(\ell)$ for $800 \leqslant \ell \leqslant 2000$ accurate to 15×10^{-6}; these values required roughly 24 h of computer time. These values were not computed as accurately since they turn out to be unimportant in finding the percolation threshold.

In figures 4 and 5, we present $\phi_c(\ell)$ as a function of $1/\ell$. We see that, for sufficiently large ℓ, the observed values of $\phi_c(\ell)$ appear to vary linearly with $1/\ell$. Using a regression fit for the data corresponding to $\ell \geqslant 4000$, we estimate the percolation threshold as

$$\phi_c = 0.676\,339(4) \tag{6}$$

which corresponds to

$$\rho_c R^2 = 0.359\,072(4) \tag{7}$$

in terms of the dimensionless number density. To obtain this result, we generated a total of approximately 1.6×10^{11} discs (including unnecessary discs), resulting in 1.2×10^{11} arcs on the frontiers, over nine weeks of CPU time.

In these evaluations, the number in parenthesis represents the error in the last digit (one standard deviation). These error estimates are found by using the errors of the individual measurements and computing the variance of the y-intercept in the regression model [12]. Our results are in agreement with previous estimates of the percolation threshold, shown in table 1, although an extra decimal place of accuracy has been added in this study.

We note that there appears to be a crossover point between two linear regimes, and so systems with $\ell < 4000$ should not be used in the regression fit to find the percolation threshold. For $\ell \geqslant 4000$, the slope of the regression line is roughly one, which is approximately equal to the corresponding slope in lattice gradient percolation by using only the occupied sites along

Table 1. Estimates of the percolation threshold ϕ_c for fully penetrable discs.

Roberts [13]	0.62
Domb [14]	0.67
Pike and Seager [15]	0.675 (2)
Fremlin [16]	0.667 (2)
Haan and Zwanzig [17]	0.683 (3)
Gawlinski and Stanley [18]	0.676 (2)
Rosso [4]	0.676 6(5)
Lorenz *et al* [19]	0.676 4(9)
Quintanilla and Torquato [1]	0.676 37(5)
Present work	0.676 339(4)

the frontier [2]. By contrast, the slope of the regression line is typically of the order of 0.01 using the weighted average approach in lattice gradient percolation [2, 20].

We also note that one could choose to measure ϕ_c using the centres of the discs on the frontier instead of the boundaries of these discs. Preliminary investigations indicate that the measurement of the percolation threshold is the same using this second approach. This and related items will be explored further in a future paper.

ST gratefully acknowledges the support of the Office of Basic Energy Sciences of the US Department of Energy under grant no DE-FG02-92ER14275.

References

[1] Quintanilla J and Torquato S 1999 *J. Chem. Phys.* **111** 5947
[2] Rosso M, Gouyet J F and Sapoval B 1985 *Phys. Rev.* B **32** 6053
[3] Sapoval B, Rosso M and Gouyet J F 1985 *J. Physique Lett.* **46** L149
[4] Rosso M 1989 *J. Phys. A: Math. Gen.* **22** L131
[5] Stoyan D, Kendall W S and Mecke J 1995 *Stochastic Geometry and Its Applications* 2nd edn (New York: Wiley)
[6] Quintanilla J and Torquato S 1997 *Phys. Rev.* E **55** 1558
[7] Ziff R M and Sapoval B 1986 *J. Phys. A: Math. Gen.* **19** L1169
[8] Press W H, Teukolsky S A, Vettering W T and Flannery B P 1995 *Numerical Recipes in C* 2nd edn (Cambridge: Cambridge University Press)
[9] Ziff R M 1998 *Comput. Phys.* **12** 385
[10] Ziff R M, Cummings P T and Stell G 1984 *J. Phys. A: Math. Gen.* **17** 3009
[11] Saleur H and Duplantier B 1987 *Phys. Rev. Lett.* **58** 2325
[12] Rice J A 1994 *Mathematical Statistics and Data Analysis* 2nd edn (Belmont, CA: Duxbury)
[13] Roberts F D K 1967 *Biometrika* **54** 625
[14] Domb C 1972 *Biometrika* **59** 209
[15] Pike G E and Seager C H 1974 *Phys. Rev.* B **10** 1421
[16] Fremlin D H 1976 *J. Physique* **37** 813
[17] Haan S W and Zwanzig R 1977 *J. Phys. A: Math. Gen.* **10** 1547
[18] Gawlinski E T and Stanley H E 1981 *J. Phys. A: Math. Gen.* **14** L291
[19] Lorenz B, Orgzall I and Heuer H-O 1993 *J. Phys. A: Math. Gen.* **26** 4711
[20] Ziff R M and Suding P N 1997 *J. Phys. A: Math. Gen.* **30** 5351