Journal of Statistical Physics, Vol. 68, Nos. 3/4, 1992

Implementation of the FCHC Lattice Gas Model on the Connection Machine

M. Hénon[1]

The 4-dimensional FCHC lattice gas model has been implemented on a Connection Machine CM-2 with 16K processors. Symmetries are used to reduce the collision table to a size that fits into local memory. This method avoids the degradation of the Reynolds coefficient R_*, but at the price of increased computing time. Bit shuffling between parallel lattices is introduced to reduce the discrepancy between measured viscosities and those predicted from the Boltzmann approximation. Thereby a model with a negative shear viscosity is obtained: a fluid having a uniform initial velocity is unstable and organized nonuniform motions develop. Because of the buildup of very strong correlations between the parallel lattices, the discrepancy with the Boltzmann values decreases only very slowly with the number of parallel lattices.

KEY WORDS: Lattice gases; computational fluid dynamics; parallel computing.

1. INTRODUCTION

Here in Nice, work on lattice gases has been concentrated on the three-dimensional FCHC model[1,2] and has followed two directions: (i) building efficient implementations; (ii) using them for various applications. Examples of applications have been described elsewhere,[3-8] and the present paper will be concerned mostly with the implementation problem.

Our first priority has been to maximize the *Reynolds coefficient*, in order to allow simulations of higher Reynolds numbers and/or to decrease the computing costs. A brief history of our efforts in that direction, and a list of the successive models which have been developed (FCHC-1 to FCHC-8), are presented in ref. 6 (see also ref. 9 for details).

[1] CNRS, URA 1362, Observatoire de la Côte d'Azur, B.P. 229, 06034 Nice Cedex 4, France.

0022-4715/92/0800-0353$06.50/0

It was realized at an early stage that in order to achieve a high Reynolds coefficient, one must use *detailed collision rules*,[9] in which an explicit collision table gives the output state for every possible input state. However, the number of possible input states in the FCHC lattice is large: 2^{24} in the simplest case, and more if rest particles are added. This makes the problems of building and implementing the table nontrivial.

We have spent some time on the first problem: how to build a collision table which maximizes the Reynolds coefficient, under given constraints. This has already been described elsewhere,[10, 5, 9, 6] and this problem can be considered as essentially solved. So I will assume here that we already have a detailed collision table to begin with.

The next problem is the implementation of the table in a computer. Because of the large number of entries, in practice only a *deterministic* collision table can be accommodated: for a given input state s there is only one allowed output state s'. Note that this contrasts with the two-dimensional FHP lattice, where a probabilistic collision algorithm is generally used. A consequence is that the collision operator is no longer strictly invariant under isometries. However, because of the large number of different collisions, it is possible to achieve statistical isotropy to better than 1.5%.[5]

Even so, a large table must be accommodated. We note also that this table is accessed with high frequency (once for every node and for every time step) and in a quasirandom fashion; therefore it must be stored in a way that allows fast random access to individual entries. Fortunately, we had access to a CRAY-2, which satisfies these requirements. In fact, it even has room for the addition of up to three rest particles, which improves the performance.[6] Programs have been running on the CRAY-2 since 1987.[5]

In January 1990, a Connection Machine Model CM-2 was installed at the Institut National de Recherche en Informatique et en Automatisme (INRIA) near Nice. I will describe here a new implementation which has been developed for that machine.[2] We were motivated by the following considerations: (i) the use of a massively parallel machine seems natural for a lattice gas simulation; (ii) parallel computers are open toward the future: there is room for a considerable improvement of their performances, in contrast to serial computers, which appear to have more or less reached the limits of presently available technologies; (iii) interactive use should be easier (see Section 4); (iv) more computer time should be available.

The architecture of the CM-2 is drastically different from that of conventional computers; therefore the collision algorithm had to be organized

[2] That work actually started in September 1989: thanks to the Connection Machine Network Server facility (CMNS) provided by Thinking Machines Corporation, a large part of the code could be written and tested before the arrival of the CM-2.

in a completely different way. In keeping with our previous philosophy, the ground rule for this new implementation was: nothing should be sacrificed concerning the Reynolds coefficient, which should keep its optimized value.

2. IMPLEMENTATION ON THE CONNECTION MACHINE

2.1. The Problem of the Collision Table

The use of a Connection Machine for lattice gas computations is a natural idea, and in fact some of the first simulations of the FHP lattice were done on a CM-1.[11] For the FCHC lattice, to the best of my knowledge, the first implementation on a Connection Machine was presented by Boghosian.[12]

In the case of the FCHC lattice, as usual, we run into a problem because of the size of the collision table. The situation is summarized in Table I, which applies for the CM-2 now available at INRIA. This machine is made up of $2^9 = 512$ *groups*. Each group consists of $2^5 = 32$ physical processors (so that the total number of physical processors is $2^{14} = 16\text{K}$),[3] plus one Weitek coprocessor and some additional circuitry. The memory is completely distributed among the processors: each physical processor has 2^{18} bits of attached memory.[4] The total amount of memory is therefore 2^{32} bits.

On the other hand, the collision table contains in the simplest case (no rest particles) $2^{24} = 16777216$ entries. The input state, coded as a 24-bit integer, can directly be used as the address into the table, since all such integers are present. Thus, each entry contains only the output state, i.e., 24 bits. In practice it is convenient to round up this last number to 32 (see below). We obtain a table size equal to 2^{29} bits (Table II, line 1). Thus, the full collision table could in principle be implemented, by spreading it over all processors.

[3] We follow the usual convention: $K = 2^{10}$, $M = 2^{20}$.
[4] Bits are a more natural unit than bytes on the CM-2.

Table I. Size of CM-2 Memory (INRIA)

	one physical processor	one group = 32 physical processors	one CM-2 = 512 groups
Memory (Mbits)	0.25	8	4096
Access time (μsec) for a 32-bit word	25	60	4000

The problem is access time (Table I, second line). The access from one processor to its memory is fast. Inside a group, access is also comparatively fast. But communication between groups is much slower, by about a factor 100.[5] Therefore this kind of communication should preferably not be used for frequent operations, such as collision table lookup. This was confirmed by attempts at implementing the full collision table (see Appendix, Section A.5).

In practice we must therefore consider that each processor has access to the memory of its group only, i.e., to a maximum of 8 Mbits of memory. But memory is also needed for other things, especially to store the state of the lattice, and in practice one cannot reasonably use much more than 50% of the memory, or 4 Mbits, for the collision table. The size of the full collision table is thus seen to be too large by more than two orders of magnitude.[6]

It is therefore necessary to devise a strategy which can work with smaller tables.[7] A first solution to this problem was proposed by Boghosian,[12] who used a method of *partial collisions*: each collision is decomposed into a sequence of 2 or 3 partial collisions; in each partial collision, only a subset of the 24 velocities participates. (A similar approach was developed by Somers and Rem, on a different machine.[13]) The size of the tables is then drastically reduced: for instance, if only 12 velocities participate, the number of entries is reduced to 4096. With this approach,

[5] These times have been measured for the PARIS instructions U_MOVE_1L, AREF32_SHARED, GET, respectively, and for a 32-bit word.

[6] The total number of processors is irrelevant here. On the other hand, the problem would be alleviated if higher density memory chips were used.

[7] The CM-2 will contain multiple copies of these tables (one in each group).

Table II. Size of Collision Tables

		Without duality		With duality	
	Entry size	Number of entries	Table size (Mbits)	Number of entries	Table size (Mbits)
Full table	32	16777216	512	9740686	297
Minimal reduced table	64	18736	1.1	10805	0.7
After momentum normalization	64	316301	19.3	182475	11.1
After ascent (12 steps)	64	29312	1.8	16875	1.0
With 7 rest particles	288	29312	8.1	16875	4.6
Somers and Rem algorithm	32	106496	3.25		

however, the collision algorithm is no longer optimal, because only a small subset of the possible collisions is allowed. As a result, the Reynolds coefficient R_* is degraded, typically by a factor of the order of 2.

This would have contradicted our ground rule. Therefore a different strategy was implemented. The basic idea is to use the symmetries of the problem to reduce the quantity of information which has to be stored. We will say that two states belong to the same *species* if one can be obtained from the other by an isometry. This is an equivalence relation; the species represent a partition of the full set of states.

We will generally consider models which violate *semi-detailed balance*.[2, 6] In that case, and in the absence of rest particles, there are no global constraints on the collision rules. Each collision can be optimized separately: one has only to determine the "best" output state for each input state considered in isolation. This optimization problem is clearly the same for two input states which belong to the same species. This suggests that it should be sufficient to store a *reduced table*, in which only one state for each species appears. This distinguished state will be called the *deputy* of the species. The number of species is found to be only 18736 (Table II, line 2), i.e., almost 1000 times less than the number of states. On the other hand, we lose a factor 2 because the reduced table is sparse: only some input states are present. Therefore the input states must also be stored, and each entry now has 64 bits. The reduced table fits comfortably into the memory of one group.[8]

If the collision table is invariant under duality (exchange of particles and holes), then one can extend the definition of a species by including also the dual states. In this case a further reduction is possible: the number of species drops to 10805. However, Somers and Rem have shown[13] that the best values of R_* are obtained with schemes which are not duality-invariant.

The computation of the collision, for a given input state s, consists of the following three steps:

1. Find the deputy $\hat{s}$ of s, and the isometry I which transforms s into $\hat{s}$.
2. Look up the reduced table to obtain the output state $\hat{s}'$ corresponding to $\hat{s}$.
3. Apply the inverse isometry I^{-1} to $\hat{s}'$ to obtain the output state s'.

[8] For models which satisfy semi-detailed balance, the construction of a reduced table is more delicate, because global constraints must be satisfied. This problem has been studied in ref. 14.

Two problems have to be solved: (i) How does one choose the deputy of each species? (ii) For a given input state s, how does one find the deputy $\hat{s}$ and the reducing isometry I? These two problems are related, since the criterion of choice of the deputies will determine the procedure. Unfortunately, no definition of the deputies is known such that $\hat{s}$ and I can be found by a simple procedure. One could, for instance, simply write each state s as a 24-bit integer, and choose as deputy of a species the state for which this integer is largest. However, with such a definition, there seems to be no way to find the deputy short of trying all 1152 isometries, which is clearly impractical.

A compromise is to accept a table which is not fully reduced: a species may be represented by more than one deputy. Many methods can be imagined. In the next section we describe the strategy which was found to give the best results. Other strategies which were tried, and also a new algorithm proposed by Somers and Rem,[15] are described in the Appendix.

2.2. Reduction Algorithm

The strategy consists of two phases: *momentum normalization*, followed by *normalized ascent*.

2.2.1. Momentum Normalization. A first idea which comes to mind is to specify that the reduced table will contain only the states which have a *normalized momentum*.[16] We use here a definition which differs slightly from that given in ref. 16. The coordinates of the momentum $\mathbf{q}$ are defined by

$$q_\alpha = \sum_{i=1}^{24} s_i c_{i\alpha} \qquad (\alpha = 1,..., 4) \tag{1}$$

where s_i is the Boolean variable representing the presence or absence of a particle with velocity $\mathbf{c}_i$. The momentum is said to be normalized if the coordinates satisfy

$$q_1 \geqslant q_2 \geqslant q_3 \geqslant q_4 \geqslant |q_1 - q_2 - q_3| \tag{2}$$

This new definition corresponds to a simpler algorithm; it was also found to give a smaller reduced table.

It is not difficult to show that a given input state s is reduced to a state with a normalized momentum by the following sequence of 11 steps. Each step consists of an *optional isometry*: a given isometry is applied if a given inequality is satisfied. Isometries are defined as in ref. 16:

S_α: symmetry with respect to the plane $x_\alpha = 0$.

$P_{\alpha\beta}$: symmetry with respect to the plane $x_\alpha = x_\beta$.

Σ_1: symmetry with respect to the plane $x_1 + x_4 = x_2 + x_3$.

Σ_2: symmetry with respect to the plane $x_1 = x_2 + x_3 + x_4$.

1. If $q_1 < 0$, apply S_1.
2. If $q_2 < 0$, apply S_2.
3. If $q_3 < 0$, apply S_3.
4. If $q_4 < 0$, apply S_4.

The q_α are now positive or zero.

5. If $q_1 < q_2$, apply P_{12}.
6. If $q_3 < q_4$, apply P_{34}.
7. If $q_1 < q_3$, apply P_{13}.
8. If $q_2 < q_4$, apply P_{24}.
9. If $q_2 < q_3$, apply P_{23}.

The q_α are now sorted in nonincreasing order.

10. If $q_1 + q_4 < q_2 + q_3$, apply Σ_1.
11. If $q_2 + q_3 + q_4 < q_1$, apply Σ_2.

The momentum **q** is now normalized.

Note that q_1, q_2, q_3, q_4 always refer to the most recently derived state.

This procedure brings the number of distinct states down to 316301 (Table II, line 3). Unfortunately, this number is still too large.

If the collision table is invariant under duality, the following step can be executed before the momentum normalization: if the number of particles exceeds 12, the input state is replaced by its dual. When this is done, another duality must also be performed at the end of the procedure. This has the effect of reducing the number of states from 16777216 to 9740686. Momentum normalization then produces a final set of 182475 states only. However, this is still too large.

2.2.2. Normalized Ascent. In order to reduce the number of states still further, we follow an idea already mentioned and we look at the 24-bit integer which represents a state s. This integer will be called the *code* of s and will be represented by $[s]$. We will try to maximize the code. However, instead of looking for the strict maximum inside the species to which s belongs, we shall be content with a heuristic procedure which only approximates the maximum. One simple procedure, which we call *ascent*, is as follows: we define, once and for all, a sequence of isometries. Each of these isometries in turn is considered and is applied if it has the effect of

increasing the code. If the sequence of isometries is well chosen, we can expect that the end state will belong to a small subset of states with large codes.

It was found that better results are achieved with a modified procedure, which we call *normalized ascent* (see below, Section A.3). Each isometry is applied only if the following two conditions are met: (i) the code increases; (ii) the momentum remains normalized. A number of experiments were made, with various sequences of isometries. In practice, the most efficient isometries are found by applying the algorithm in parallel to all states, and counting how many states remain after every step.[9] The following sequence of 12 isometries was finally selected:

$$\Sigma_1, P_{34}, \Sigma_2, S_4, P_{23}, P_{34}, P_{12}, S_3, S_4, P_{34}, \Sigma_1, P_{23} \tag{3}$$

We have thus the following 12 additional steps:

1. If $[\Sigma_1 s] > [s]$ and $q_1 + q_4 = q_2 + q_3$, apply Σ_1.
2. If $[P_{34} s] > [s]$ and $q_3 = q_4$, apply P_{34}.
3. If $[\Sigma_2 s] > [s]$ and $q_2 + q_3 + q_4 = q_1$, apply Σ_2.
4. If $[S_4 s] > [s]$ and $q_4 = 0$, apply S_4.
5. If $[P_{23} s] > [s]$ and $q_2 = q_3$, apply P_{23}.
6. If $[P_{34} s] > [s]$ and $q_3 = q_4$, apply P_{34}.
7. If $[P_{12} s] > [s]$ and $q_1 = q_2$, apply P_{12}.
8. If $[S_3 s] > [s]$ and $q_3 = 0$, apply S_3.
9. If $[S_4 s] > [s]$ and $q_4 = 0$, apply S_4.
10. If $[P_{34} s] > [s]$ and $q_3 = q_4$, apply P_{34}.
11. If $[\Sigma_1 s] > [s]$ and $q_1 + q_4 = q_2 + q_3$, apply Σ_1.
12. If $[P_{23} s] > [s]$ and $q_2 = q_3$, apply P_{23}.

Here again, s, q_1, q_2, q_3, q_4 refer to the most recently derived state.

This reduces the number of distinct states to 29312 for a nondual table, or 16875 for a dual table (Table II, line 4), and so at last we obtain a reduced table which fits into the available memory, without degrading the Reynolds coefficient.

[9] Only a restricted set of isometries was considered: the 24 isometries which are symmetries with respect to a hyperplane. This is because (i) this restricts the search space to a more manageable size; (ii) these isometries are comparatively easy to implement on the CM-2, as a series of bit swaps.

2.3. Computing Speed

The price we have to pay is in computing time. The computation of a collision now involves the application of a sequence of 23 isometries[10]; a search in the reduced table; and the application of the 23 inverse isometries. The table search is itself time-consuming because the table is sparse: only some input states are present. Therefore one cannot simply identify the input state with the address, and a binary search is necessary.[11]

Table III shows the time spent by one CM-2 processor on the various phases of a node update, in microseconds. The total time is 6130 μsec. For the INRIA machine, with 16K processors, this gives a speed of 2.7×10^6 node updates per second; for a "full" CM-2, with 64K processors, the speed would be 10.7×10^6 node updates per second. This is about 3 times slower than a CRAY-2 in quadri-processor mode.[7]

Propagation takes only 250 μsec or about 4% of the total time (this figure is for a 3-dimensional simulation).

2.4. Adjustable Viscosity

For a given collision table and for a given density d the algorithm produces a fixed viscosity ν. For some applications, however, a variable viscosity is desirable. Clearly it is not possible to obtain a viscosity smaller than the nominal value ν, since the table has already been optimized to give the lowest possible value. It is possible, however, to obtain a higher viscosity, simply by omitting the collision step at some nodes and/or some time steps.

[10] The fact that these isometries are optional, i.e., applied only to a subset of nodes, does not result in any time saving because the CM-2 is an SIMD machine.

[11] A hashing scheme should perhaps be considered. However, this would increase the size of the table.

Table III. Time Required by the Various Steps of One Node Update (in microseconds)

Computation of number of particles and momentum	500
Momentum normalization	890
Ascent	1320
Table lookup	2260
Inverse isometries	910
Propagation	250
Total	6130

We define a *collision probability* $f \leqslant 1$. The output state will be computed by the collision algorithm with probability f, and will be set equal to the input state with probability $1-f$. This corresponds to using new transition probabilities $\hat{A}(s \to s')$ given by

$$\hat{A}(s \to s') = \begin{cases} fA(s \to s') & \text{if} \quad s' \neq s \\ fA(s \to s) + 1 - f & \text{if} \quad s' = s \end{cases} \tag{4}$$

The quantity $1/\lambda$ defined by (3.25) in ref. 10 becomes

$$\frac{1}{\hat{\lambda}} = f \frac{1}{\lambda} \tag{5}$$

since it implies only terms with $s' \neq s$, and the Boltzmann viscosity given by (3.38) in ref. 10 becomes

$$\hat{\nu} = \frac{1}{3}\left(\frac{\lambda}{f} - \frac{1}{2}\right) \tag{6}$$

The nominal value corresponds to $f = 1$:

$$\nu = \tfrac{1}{3}(\lambda - \tfrac{1}{2}) \tag{7}$$

By adjusting f between 1 and 0, one can therefore in principle obtain any viscosity $\hat{\nu}$ in the range $\nu \leqslant \hat{\nu} \leqslant +\infty$.

One way to implement this scheme is to generate a random number at each node and each time step to decide if the collision takes place. This works; but the generation of a large amount of random numbers is time-consuming. A more economical method is to *decimate* the collision table: for a fraction $1-f$ of the entries, chosen at random, the tabulated value of the output state is replaced by the value of the input state. This is done once and for all when the table is loaded into memory. Measurements of $\hat{\nu}$ give values in good agreement with (6).

2.5. Applications

The models which have been implemented so far are FCHC-3, FCHC-6, FCHC-7, FCHC-8 (see ref. 6 for the definitions), a new model FCHC-9 described below in Section 3, and a few other experimental models. Among the applications which have already been developed, we mention the following.

1. Systematic measurements of the viscosity ν and of the Galilean factor g for various models.

2. Measurements of correlations (in progress).
3. Flow behind a grid.
4. Instability of a modified Kolmogorov flow. Results have been reported in ref. 8.
5. Elaboration of a model with a negative viscosity.

Only the last topic will be described here.

3. NEGATIVE-VISCOSITY MODEL

One of our long-time goals has been to devise a model with a negative viscosity. This model could in principle also be made to exhibit a positive but arbitrarily small viscosity, and therefore arbitrarily large Reynolds coefficients, by using, for instance, the scheme described in Section 2.4. Such a model would also be of interest from the point of view of statistical mechanics (see ref. 6, Section 9.1). We recall that a necessary condition to obtain a negative viscosity is that the collision rules violate semi-detailed balance.[10]

3.1. Parallel Lattices and Bit Shuffling

In ref. 6 it was found that for models which violate semi-detailed balance, the measured viscosity is unfortunately systematically larger than the predicted value, based on the Boltzmann approximation.[10] It was therefore suggested to use *parallel lattices*, or *coupled lattices*, in which corresponding bits are randomly shuffled. In the limit of a large number of lattices, the viscosity of such a model should approach the Boltzmann value.

Parallel lattices are simply identical copies of a given basic lattice; they can be thought of as "parallel universes." At time $t = 0$, the particle population of each lattice is computed from the same velocity field $\mathbf{u}$ and density d, but using different random numbers. Thus, the macroscopic quantities are the same in all lattices (apart from statistical fluctuations), although the microscopic realizations are different. This situation normally persists in time because of the shuffling.

Implementing parallel lattices on the CM-2 is quite easy: all that is necessary is to add dimensions to the basic geometry. The computing time and the memory requirements increase, being proportional to the number Y of parallel lattices; in compensation we might expect a decreased level of noise, since averages will be taken on a larger number of nodes.

It is convenient to think of each lattice as a symbolic plane, where one

dimension stands for space and the other for velocity. The parallel lattices are then viewed as parallel planes, stacked in a third dimension. For a given node and a given velocity, in each lattice one bit represents the presence or absence of a particle. This corresponds to a point in the symbolic plane. For the set of parallel lattices, we have then a collection of Y bits, which will be called a *column*, since it corresponds to a column of points in the symbolic space.

The shuffling operates on columns: at the end of each time step, the bits of each column are randomly mixed. Various mixing strategies were tried, with various degrees of randomness:

1. We take $Y=2^y$ coupled lattices. This is implemented on the CM-2 by creating y additional dimensions, each with size 2. Along each of these additional dimensions, the bits are swapped with probability $\frac{1}{2}$. A different random choice is made for each column, i.e., for each node and for each velocity. This uses random numbers generated on the CM-2.
2. To save time, a single choice is made for all nodes. Thus, for each velocity, and along each additional dimension, the bits are swapped at all nodes with probability $\frac{1}{2}$. This uses random numbers generated on the front end.
3. The shuffling provided by the above method is far from random: only 2^y different permutations can be created, out of a total number of $Y!$ possible permutations of the Y bits in a column. It might be feared that this shuffling is insufficient. Therefore another method was tried: for each velocity, a fully random permutation is selected. This permutation is then applied at every node. This again uses random numbers generated on the front end.

Surprisingly, no measurable difference was found between the results of these three methods: the effects are insensitive to the shuffling strategy. Method 2 was therefore adopted since it is the most economical in terms of computing time.

Results for model FCHC-6 (no rest particles) are shown in Fig. 1 (top curve). The abscissa is the number Y of coupled lattices, on a logarithmic scale. The ordinate is the kinematic shear viscosity, measured as described in ref. 6: the density has a constant value d and the initial velocity has two nonzero components, $u_x=u_{x0}$, $u_y=u_{y0}\cos kx$, with u_{x0}, u_{y0}, k constants (this allows a simultaneous measurement of ν and of the Galilean factor g). Each value of ν was obtained as an average over several runs; the rms error, determined from the internal dispersion, is of the order of 5×10^{-5}. A four-dimensional lattice was used, with a total number of nodes

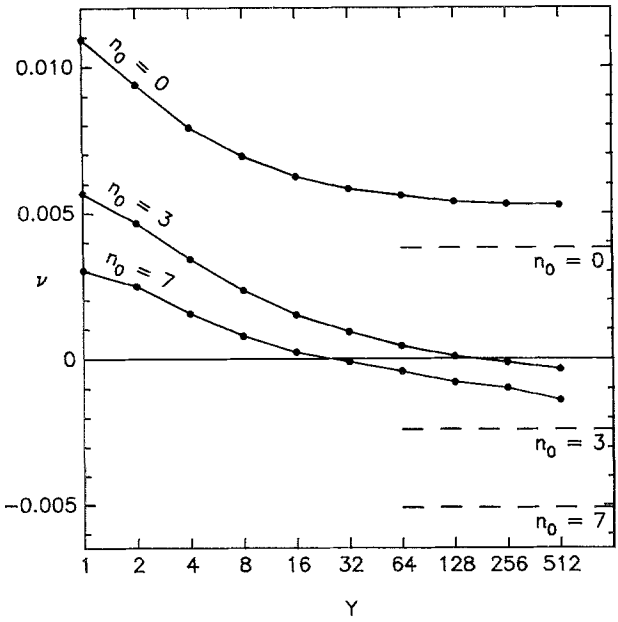

Fig. 1. Kinematic viscosity ν as a function of the number Y of parallel lattices, for models violating semi-detailed balance, with 0, 3, and 7 rest particles, and for a density $d = 0.5$. Dashed lines: values of ν computed under the Boltzmann approximation.

(including the parallel lattices) equal to 2^{18}. The density $d = 0.5$ was chosen, since this is the value at which ν is minimal. Other parameter values are $u_{x0} = 0$, $u_{y0} = 0.2$, $k = 2\pi/16$. The Boltzmann value $\nu = 0.0038$ is also shown as a dashed line.

As predicted, the viscosity decreases when the number of parallel lattices increases, and it seems to converge toward the Boltzmann value for $Y \to \infty$. However, the convergence is disappointingly slow. Y was pushed to high values (which would be impractical in a simulation); even for $Y = 512$ the discrepancy between the measured viscosity and the Boltzmann value is still about 20% of the discrepancy for a single lattice.

Similar computations were made for model FCHC-3, which does satisfy semi-detailed balance. The viscosity as a function of the number of parallel lattices is shown on Fig. 2, again for $d = 0.5$. The rms error bars are also represented. Note that the vertical scale of that figure is dilated 10 times with respect to that of Fig. 1. Two major differences with the previous case of no semi-detailed balance are found: (i) even with a single lattice, the measured viscosity is close to the Boltzmann value; (ii) the discrepancy decreases quickly when the number of parallel lattices

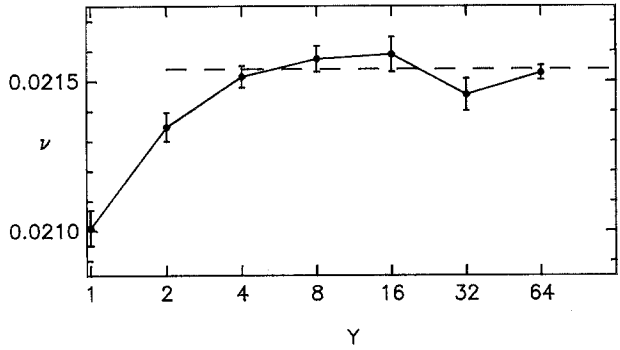

Fig. 2. Kinematic viscosity ν as a function of the number Y of parallel lattices, for a model satisfying semi-detailed balance, and for a density $d=0.5$. Dashed line: value of ν computed under the Boltzmann approximation.

increases: already with 4 lattices there is no measurable difference. We remark also that the sign of the discrepancy has changed: the measured viscosity is *smaller* than the Boltzmann value.

3.2. Addition of Rest Particles

Another way to decrease the viscosity is to add rest particles. This is easily implemented in the CM-2 code. However, the number of states increases, and therefore the size of the collision table increases, too.

When there are rest particles, the reduced collision table is best organized as a row-and-column table, where the row is determined by the *moving input state*, i.e., the set of the moving particles of the input state, and the column is given by the number of rest particles of the input state. This number can vary from 0 to a fixed maximal value n_0. Thus, the number of rows is the same as before, but one row contains now n_0+2 numbers (one moving input state and n_0+1 output states). If duality is used, the largest practical value of n_0 is about 7: one row takes up then $9\times 32=288$ bits, and the table size is 4.6 Mbits (Table II), or a little over one half of the total memory.

On the CRAY-2, with a full table, models with up to 3 rest particles can be implemented (model FCHC-7 in ref. 6). Figure 1 (middle curve) shows the viscosity computed on the CM-2 as a function of the number of parallel lattices for that model. Here a total number of 2^{20} nodes and a value $u_{x0}=0.05$ were used. Again the viscosity decreases when Y increases, and this time we finally reach our goal: a negative value of the viscosity is found with 256 parallel lattices.

The lower curve on Fig. 1 shows the results for $n_0=7$. A negative

value is now reached with 32 parallel lattices. This model will be called *FCHC-9*. The convergence toward the Boltzmann value is even worse than in the previous cases. We note also that the difference between $n_0 = 3$ and $n_0 = 7$ is not large; this suggests that it would not be worthwhile to increase further the number of rest particles.

3.3. Properties of Model FCHC-9

The viscosity in a lattice gas is known to depend somewhat on the velocity.[17] This is an effect which does not appear in leading order when the Navier–Stokes equations are derived by a multiscale expansion.[2] However, subdominant terms, such as velocity-dependent corrections to the viscosity, can become relevant when the viscosity is very small, as in the present case. Figure 3 shows, for model FCHC-9, that ν indeed varies with the parameters u_{x0} and u_{y0} of the initial velocity field. For low values of u_{x0}, the viscosity even changes sign when the wave amplitude u_{y0} increases. In a typical simulation, the velocities are of the order of 0.2. Therefore,

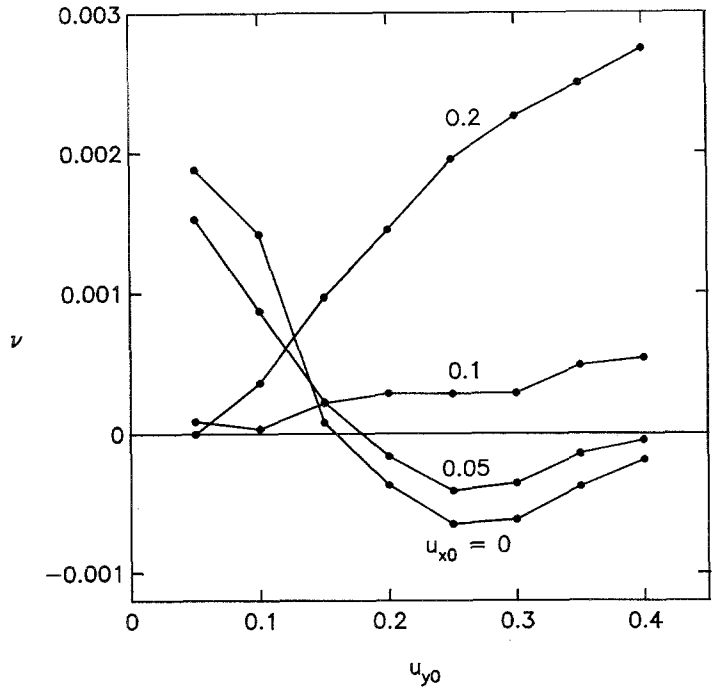

Fig. 3. Kinematic viscosity ν as a function of the parameters u_{x0} and u_{y0} of the initial velocity field, for model FCHC-9 (7 rest particles, 32 parallel lattices) with a density $d = 0.5$.

unfortunately, the viscosity will not be defined to better than about 0.001 in such a simulation. In model FCHC-9, where the average value is close to 0, even the sign of the viscosity will be undefined. This means that the Reynolds number is essentially undefined.

Figure 4 shows how the viscosity changes with the density d. Note that there is a small region around $d=0.5$ where the viscosity is almost constant.

The leading-order viscous effect involves a fourth-order tensor which is isotropic when all the FCHC symmetries hold. This is not strictly the case here because we use a deterministic collision table (see Introduction). Moreover, velocity-dependent corrections involve a sixth-order tensor which will not in general be fully isotropic. Therefore the isotropy of the viscosity tensor was also studied. No significant variation of ν was found under permutations of axes. However, a small anisotropy was detected for a wavevector oriented at 45° to the x and y axes.

No significant dependence of the viscosity on the wavenumber k was observed in the standard axes. With a wavevector oriented at 45°, a barely significant dependence was found; a rough expression for the viscosity, derived from a few measurements, is $\nu \approx 0.00052 - 0.0006k^2$.

We can verify that the viscosity is negative by starting with a fluid in uniform motion (Fig. 5). The lattice dimensions are 128×128, with periodic boundary conditions; the density is $d=0.5$; the fluid has initially a uniform velocity $u_{x0}=0.1$ along the x axis (this value was chosen because it gives rise to a well-marked instability). No space averaging is done: individual nodes are shown. Time averaging is done over 200 time steps. Black represents a positive velocity along the x axis.

A spontaneous organization into alternating horizontal currents develops. The vertical wavelength is of the order of 5 lattice nodes. From the observed doubling time, of the order of 1000, one deduces then a

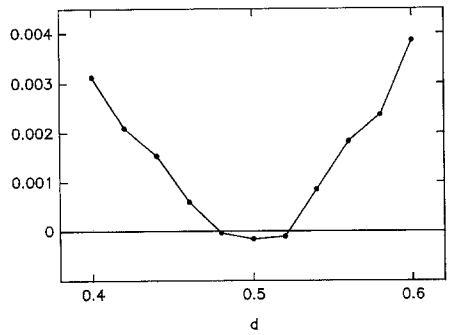

Fig. 4. Kinematic viscosity ν as a function of the density d for model FCHC-9.

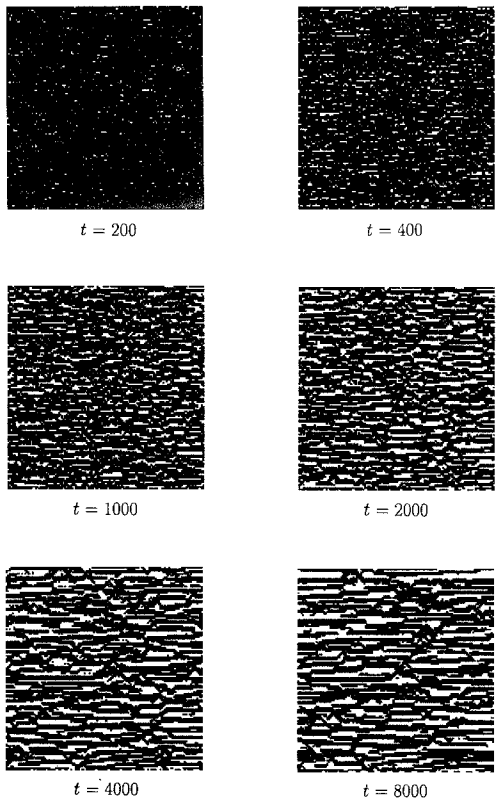

Fig. 5. Instability of a fluid with a negative viscosity (model FCHC-9). The fluid has initially a uniform velocity $u_{x0} = 0.1$ along the x axis.

viscosity of the order of $\nu = -0.0005$. Note that once the pattern is fully formed, it becomes essentially frozen.

A model with negative viscosity was obtained earlier by Rothman,[18] but with a nonlocal collision algorithm: information is extracted from the neighboring nodes in order to obtain a better estimate of the velocity gradient.

3.4. Correlations Between Parallel Lattices

It was suspected that the poor relaxation of the viscosity toward the Boltzmann value in the present model FCHC-9 might be due to the buildup of correlations between the parallel lattices. Therefore the following statistics were made. We consider a fluid with density $d = 0.5$ and velocity

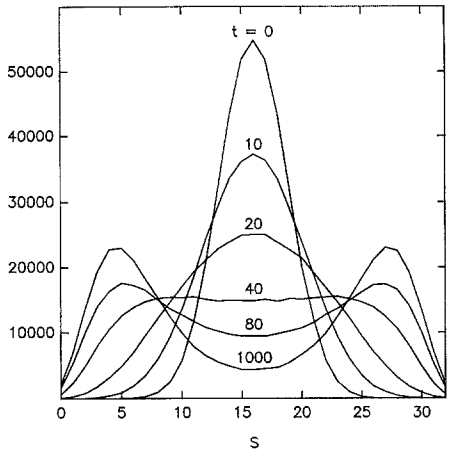

Fig. 6. Statistics on the values of S (number of bits 1 in a column) for model FCHC-9.

$\mathbf{u} = 0$. At each time step, and in each column, we count how many bits are equal to 1. This number will be represented by S; we have $0 \leqslant S \leqslant Y$ ($Y = 32$ in the present case). Next we make statistics on S: we consider all columns and we count how many times each value of S appears. Each lattice has a size 128×128; therefore the number of columns is $128 \times 128 \times 24 = 393216$. Each bit has a probability $\frac{1}{2}$ of being equal to 1; if the bits of a column are independent, we should get a binomial distribution. This is indeed what we observe at the initial time $t = 0$ (Fig. 6). As time goes on, however, the distribution progressively changes into a completely different shape, with two maxima. This means that in any given column, either most of the bits are 0, or most of them are 1. This explains why the device of parallel lattices turns out to be so inefficient: the lattices tend to become copies of each other, and no longer carry independent information.

Experiments indicate that the time scale for the buildup of the correlations is proportional to the number Y of parallel lattices.

As a check, similar statistics were made for model FCHC-3, which satisfies semi-detailed balance. Again 32 parallel lattices were used. No correlations appear in that case; the initial binomial distribution remains unchanged.

Vergassola suggested[19] that the development of strong correlations in model FCHC-9 is related to the fact that the viscosity is close to zero. This was verified by modifying the density so as to increase ν (Fig. 4): the correlations quickly decrease as d moves away from 0.5.

4. CONCLUSIONS

1. Apart from the problem of the collision table, the implementation of a lattice gas scheme on the Connection Machine CM-2, using the PARIS language, was found to be quite easy and natural. Perhaps this is due in part to the fact that one of the applications initially envisaged for this machine[20] was artificial intelligence, hence the emphasis on Boolean operations and other bit manipulations. We list some specific instances:

- Fields of arbitrary lengths can be used. For instance, since each component of the momentum can take only integer values from -6 to $+6$, it can be represented as a 4-bit field. This results in important economies of memory and computing time.
- The memory is addressable at the bit level. For instance, the state of a node is represented by a 24-bit field; each bit, corresponding to a particular velocity, is individually addressable.
- A single instruction, using the NEWS structure, propagates all particles with a given velocity to the next node in a given direction parallel to one of the axes. Periodic boundary conditions are automatically implemented.
- Thanks to the concepts of *geometry* and of *virtual processors*, the lattice can have an arbitrary number of dimensions and an arbitrary number of nodes. One has only to specify the number of dimensions and the size of each dimension; the details of the internal implementation are invisible to the programmer.
- With the *context* flag, an arbitrarily defined subset of the nodes can be activated at any given time. This makes it simple to program operations which concern only the fluid or only the walls, for instance.

2. One may try to briefly evaluate the respective performances of the CRAY-2 and of the Connection Machine for lattice gas problems. (This judgment might be biased, however, because I have no personal experience with the CRAY machines.) The CM-2 appears to be more user-friendly. It has very good input–output and graphic facilities, and can be operated in an interactive mode, almost as easily as an ordinary workstation. It can also be operated from a distance, through the X Windows system. On the other hand, the CRAY-2 is faster and has more memory than the CM-2 available to us. Thus, the two machines appear as complementary. The CM-2 is more comfortable for testing and debugging, experimenting with parameter values, and running small to medium simulations. The CRAY-2 is preferable for large simulations, such as those reported in ref. 7.

3. The present results demonstrate that it is possible to build a lattice gas model with a negative viscosity, using purely local collision rules. Nothing in principle prevents such a model from existing, provided that semi-detailed balance is violated. However, actually obtaining a specific instance turned out to be more difficult than anticipated.

Unfortunately, this negative-viscosity model is rather heavy. With 7 rest particles, more than half of the memory is used by the collision table. On the 16K INRIA Connection Machine, the maximum number of nodes which can be accommodated in the remaining memory is 2^{21}. Since there are 32 parallel lattices, the maximum number of actual lattice nodes is limited to 2^{16}.

There is also a price to pay in the computing time, which is multiplied by 32. This is only partially recouped by the decrease in the noise level at macroscopic output, since the parallel lattices are strongly correlated. Shuffling particles between the parallel lattices also takes time.

Another problem, already noted, is that the viscosity is rather ill-defined.

Finally, the gain in the Reynolds coefficient turns out to be offset by the smaller size of the lattice. In order to have a nonzero Galilean factor g, we must move away from $d=0.5$; a value $d=0.45$ seems reasonable. Measurements give then $g=0.063$. With $\nu=0.001$ (Fig. 4), we obtain from (40), (41), and (53) in ref. 6 a value of $R_*=27$, which seems to be an improvement by a factor 2 over the best previous value $R_*=13.5$ for model FCHC-8. However, since the available number of nodes is divided by 32, the characteristic scale l_0 (expressed in lattice units) in a three-dimensional simulation will be divided by $32^{1/3} \simeq 3.2$. Therefore the Reynolds *number*, given by (50) in ref. 6, is actually smaller than before.

Therefore this model does not seem to be usable for large-scale fluid simulations. It might be of interest, however, for fundamental statistical mechanics studies on the behavior of a system in the vicinity of $\nu=0$.

It would be highly desirable to find another way to coerce the viscosity into approaching the Boltzmann value. For this, a detailed understanding of the correlations will probably be necessary.

4. As pointed out in the Introduction, the present scheme achieves only approximate isotropy. Exact isotropy could be obtained by the following modification of the collision algorithm[13, 15]: a *random* isometry (i.e., randomly chosen among all 1152 isometries) is first applied to the input state; then the deterministic algorithm is applied; finally the inverse of the initial random isometry is applied.

In the CRAY-2 implementation, where the computation of a collision consists of a single table lookup, this modification would considerably degrade the performance, and for this reason it has never been

implemented. In the present scheme, on the other hand, the computing cost of a collision is already large and it might be of interest to add the randomizing isometries.

APPENDIX. ALTERNATIVE REDUCTION STRATEGIES

We describe briefly some other strategies which were tried in order to reduce the size of the collision table.

A.1. ASCENT ONLY

In this and the following three sections, for historical reasons, only the case of a duality-invariant table is considered. The results should therefore be compared with those of the last two columns of Table II.

Starting from the initial set of 9740686 states, we apply directly the ascent procedure, as described in Section 2.2.2, without paying any attention to the momentum. In other words, we simply try to maximize the code.

The best sequence of isometries which was found produced after 25 steps a reduced set of 62358 states. By comparison, the method of Section 2.2 produces in the dual case a reduced set of 16875 states after a total of 23 steps.

A.2. MOMENTUM NORMALIZATION WITH ASCENT

Here we use momentum normalization as described in Section 2.2.1, with the following variation: when there is an indifferent choice, i.e., when the test produces a zero value, then instead of doing nothing, we apply the isometry if it increases the code. For instance, the first step becomes:

1. If $q_1 < 0$, or if ($q_1 = 0$ and $[S_1 s] > [s]$), apply S_1.

Again for historical reasons, in this and the following two sections, the normalization differs slightly from that described in Section 2.2.1: the last two steps are replaced by

10. If $q_2 + q_3 < q_1 + q_4$, apply Σ_1.
11. If $q_4 < 0$, apply S_4.

After the 11 steps of this modified normalization, the number of states is reduced from 9740686 to 43469.

A.3. MOMENTUM NORMALIZATION, FOLLOWED BY UNNORMALIZED ASCENT

This strategy is quite similar to that described in Section 2.2, the only difference being that during the ascent phase, we do not ask that the momentum remains normalized. Thus, for instance, step 1 of the ascent becomes simply

1. If $[\Sigma_1 s] > [s]$, apply Σ_1.

In the best case, the number of states is reduced after 12 ascent steps to 21433.

A.4. PARTIAL NORMALIZATION OF SECOND-ORDER MOMENTUM

A natural idea is to extend the normalization approach, using not only the first-order momentum $\mathbf{q}$ but also the second-order momentum X. This is a symmetric 4×4 tensor:

$$X_{\alpha\beta} = \sum_i s_i c_{i\alpha} c_{i\beta} \tag{A1}$$

A full normalization of the second-order momentum would be too complicated and time-consuming. A partial normalization can be achieved as follows. First we use the isometries $P_{\alpha\beta}$ to put the diagonal elements in nonincreasing order: $X_{11} \geqslant X_{22} \geqslant X_{33} \geqslant X_{44}$. Then we use the isometries S_α to make as many nondiagonal elements positive or zero as possible. Since the product $S_1 S_2 S_3 S_4$ (symmetry with respect to the origin) leaves the second-order momentum invariant, only three of these isometries are independent; therefore only three nondiagonal terms can be made nonnegative. We can, for instance, use S_2, S_3, and S_4 to make respectively X_{12}, X_{13}, and X_{14} nonnegative.

Since we do not want to lose the benefit of the previous first-order normalization, these isometries are applied only if they leave the first-order momentum invariant.

This algorithm represents 8 additional optional isometries. For a dual table, the number of states was found to be reduced from 182475 to 47261. This is not as efficient as the strategy described in Section 2.2. Also the computation of the second-order momentum components, and their updating after each isometry, are time-consuming. Therefore this approach was not pursued.

A.5. FULL TABLE IMPLEMENTATION

I have tried implementing the full collision table, distributing it between all processor memories. It is then necessary to use the router communication mechanism of the CM-2, with the PARIS instruction GET, to retrieve an output state from the table. As mentioned in ref. 21, Section 7.4, this instruction has two drawbacks: (i) it is time-consuming, because each node must first send a request to the appropriate processor, containing the desired table entry, and then this entry must be sent back to the node; (ii) it uses large amounts of temporary storage, because the path from every node to the corresponding table entry must be temporarily stored. Experiments show indeed that from 500 to 1000 bits of memory per lattice node are required. This restricts the size of the lattices which can be simulated. Moreover, the time taken by the GET instruction is found to increase sharply when the memory limit is approached; apparently a traffic jam begins to develop.

The lattice gas problem may in fact represent one of the worst possible cases for the use of the router, because (i) the probabilities of the input states are very uneven, so that some table entries will be queried simultaneously by many lattice nodes, while other entries are not used; (ii) the communication pattern is completely disordered: any node may query any table entry, in a quasirandom fashion; (iii) the communication pattern is entirely different from one time step to the next, so that there is no possibility to use a router compiler.

On a larger CM-2 machine, the strategy of full table implementation might be viable. Good results were reported, for instance, on a machine where each processor has 4 times more memory.[22] When ample room is available, the GET instruction uses about 4000 μsec. This compares favorably with the figures quoted in Table III.

A.6. NEW REDUCTION ALGORITHM OF SOMERS AND REM

A new method for reducing the size of the table has been recently proposed by Somers and Rem.[15] Its structure is similar to that of the present paper: for a given input state s, one finds an isometry I which transforms s into a deputy[12] $\hat{s}$; then a reduced collision table is looked up to find the corresponding output state $\hat{s}'$; and finally the inverse isometry I^{-1} is applied to obtain s'. The difference lies in the choice of the deputies and in the method used to determine I and $\hat{s}$. For each set of two opposite velocities, a "parity bit" (my notation) is defined, equal to 1 if one particle

[12] Called a *representative* in ref. 15.

is present, and to 0 if the two particles are both present or both absent. The set of the parity bits forms a 12-bit integer s_d. Isometries are then applied to reduce s_d to a normalized form. The number of entries in the reduced table is 106496. This table can be set up essentially as a full table: the address can be simply computed from the input state, and therefore only the input state has to be stored.

As a result, the reduced table fits into the allowed 4 Mbits on the CM-2 (Table II, line 5). I have made a first try at coding Somers and Rem's method on the CM-2. The result is a total time of 1450 μsec for one node update (1270 for collisions and 180 for propagation, for a three-dimensional simulation).[13] This is about 4 times faster than the method of the present paper (Table III). For a full CM-2 with 64K processors, the speed would thus be 45 million node updates per second.

The reasons for the better performance of Somers and Rem's algorithm appear to be that (i) with the addition of a few auxiliary tables, the whole collision algorithm can be programmed essentially as a sequence of lookups, plus a few simple operations;[15] (ii) in particular, the reducing isometry depends only on s_d, which is 12 bits long, and can therefore be read at once from a table with only 4096 entries; (iii) as already noted, the reduced collision table can be set up as a full table, with immediate access to the required entry: no binary search is required.

On the other hand, it should be noted that the reducing scheme of the present paper achieves a smaller reduced table. In particular, enough room is left for the addition of several rest particles. This made possible the model with negative viscosity described in Section 3.

ACKNOWLEDGMENTS

I thank B. Boghosian, C. Caquineau, R. Fournier, D. Lloyd Owen, D. Ray, and H. Scholl for their help with the use of the Connection Machine, and a referee for constructive comments. This work was supported by the European Community under grant SC1-0212-C and by DRET under grant 90/1444. The computations were done on the Connection Machine Model CM-2 of the Centre Régional de Calcul PACA, antenne INRIA-Sophia-Antipolis through the R3T2 network.

[13] In ref. 15, an additional step is executed in order to ensure exact isotropy: a random isometry is first applied to the input state. At the end of the process, the inverse isometry is applied. This additional step has been dropped in the CM-2 adaptation, in order to make it faster, and to allow a fair comparison with the scheme of the present paper, which produces only statistical isotropy (Section 1). Reintroducing this randomizing step would approximately double the computing time on the CM-2.

REFERENCES

1. D. d'Humières, P. Lallemand, and U. Frisch, Lattice gas models for 3D hydrodynamics, *Europhys. Lett.* **2**:291–297 (1986).
2. U. Frisch, D. d'Humières, B. Hasslacher, P. Lallemand, Y. Pomeau, and J.-P. Rivet, Lattice gas hydrodynamics in two and three dimensions, *Complex Systems* **1**:649–707 (1987).
3. J.-P. Rivet, Simulation d'écoulements tri-dimensionnels par la méthode des gaz sur réseaux: premiers résultats, *C. R. Acad. Sci. Paris II* **305**:751–756 (1987).
4. J.-P. Rivet, Hydrodynamique par la méthode des gaz sur réseaux, Thèse, Université de Nice (1988).
5. J.-P. Rivet, M. Hénon, U. Frisch, and D. d'Humières, Simulating fully three-dimensional external flow by lattice gas methods, *Europhys. Lett.* **7**:231–236 (1988).
6. B. Dubrulle, U. Frisch, M. Hénon, and J.-P. Rivet, Low viscosity lattice gases, *J. Stat. Phys.* **59**:1187–1216 (1990).
7. J.-P. Rivet, Brisure spontanée de symétrie dans le sillage tridimensionnel d'un cylindre allongé, simulé par la méthode des gaz sur réseaux, *C. R. Acad. Sci. Paris II* **313**:151–157 (1991).
8. M. Hénon and H. Scholl, Lattice gas simulation of a non-transverse large-scale instability for a modified Kolmogorov flow, *Phys. Rev. A* **43**:5365–5366 (1991).
9. M. Hénon, Optimization of collision rules in the FCHC lattice gas, and addition of rest particles, in *Discrete Kinetic Theory, Lattice Gas Dynamics and Foundations of Hydrodynamics*, R. Monaco, ed. (World Scientific, Singapore, 1989), pp. 146–159.
10. M. Hénon, Viscosity of a lattice gas, *Complex Systems* **1**:763–789 (1987).
11. J. B. Salem and S. Wolfram, Thermodynamics and hydrodynamics with cellular automata, in *Theory and Applications of Cellular Automata*, S. Wolfram, ed. (World Scientific, Singapore, 1986), pp. 362–366.
12. B. Boghosian, Three-dimensional FCHC lattice gas simulation on the CM2 Connection Machine, presented at the NATO Advanced Research Workshop on Lattice Gas Methods for PDE's: Theory, Applications and Hardware, Los Alamos, September 6–8, 1989.
13. J. A. Somers and P. C. Rem, The construction of efficient collision tables for fluid flow computations with cellular automata, in *Cellular Automata and Modeling of Complex Physical Systems*, P. Manneville, ed. (Springer, 1989), pp. 161–177.
14. G. Westland, Optimizing a reduced collision table for the FCHC-lattice gas automaton, Master's thesis, State University of Utrecht (April 1991).
15. J. A. Somers and P. C. Rem, Obtaining numerical results from the 3D FCHC-lattice gas, *Lecture Notes in Physics* **398**:59–78 (1992).
16. M. Hénon, Isometric collision rules for the four-dimensional FCHC lattice gas, *Complex Systems* **1**:475–494 (1987).
17. K. Diemer, K. Hunt, S. Chen, T. Shimomura, and G. Doolen, Density and velocity dependence of Reynolds numbers for several lattice gas models, in *Lattice Gas Methods for Partial Differential Equations*, G. Doolen *et al.*, eds. (Addison-Wesley, 1990), pp. 137–177.
18. D. H. Rothman, Negative-viscosity lattice gases, *J. Stat. Phys.* **56**:517–524 (1989).
19. M. Vergassola, Private communication (1990).
20. W. D. Hillis, *The Connection Machine* (MIT Press, Cambridge, Massachusetts, 1985).
21. Thinking Machines Corporation (Cambridge, Massachusetts), Introduction to Programming in C/Paris, Version 5 (June 1989).
22. S. Chen, Private communication (22 March 1991).

Michel Hénon