Cellular Automata and Parallel Processing for Practical Fluid-Dynamics Problems

H. Abarbanel
K. Case
A. Despain
F. Dyson
M. Freedman
C. Max
D. Nelson
O. Rothaus

September 1990

JSR-86-303

JASON
The MITRE Corporation
7525 Colshire Drive
McLean, Virginia 22102-3481
(703) 883-6997

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE September 20, 1990	3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE Cellular Automata and Parallel Processing for Practical Fluid-Dynamics Problems	5. FUNDING NUMBERS PR - 8503Z
6. AUTHOR(S) H. Abarbanel et.al	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) The MITRE Corporation JASON Program Office A10 7525 Colshire Drive McLean, VA 22102	8. PERFORMING ORGANIZATION REPORT NUMBER JSR-86-303
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Department of ~~Defense~~ Energy ~~Strategic Defense Initiative Organization~~ Washington, DC ~~20301~~ 20585	10. SPONSORING/MONITORING AGENCY REPORT NUMBER JSR-86-303
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

During the 1986 JASON Summer Study a group of JASONs undertook to examine, under the sponsorship of the Department of Energy and DARPA, the utility of cellular automata in physical science calculations, especially in fluid dynamics. We expanded the scope of our study to include several related topics which we concluded would be of some interest to our sponsors. These include: (1) a comparison of cellular automata (CA) techniques to "conventional" methods of solving the partial differential equations of fluid dynamics or other physical situations; and (2) the utility and status of using parallel or concurrent processing machines for doing either CA or conventional fluids calculations.

14. SUBJECT TERMS cellular automata; vlsi chip, impressible fluid algorithmns; two, three, or four dimensions; octogonal quasilattice	15. NUMBER OF PAGES
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT SAR

Contents

Sponsoring/Monitoring agency was U. S. Dept. of Energy, Office of Program Analysis/ER-32. Washington, DC 20585.

Dist. "A" per Charles mandelbaum. Office of Program Analysis/ER-32. U. S. Department of Energy/ER-32. Washington, DC 20585.
VHG 11/07/90

1 INTRODUCTION

During the 1986 JASON Summer Study a group of JASONs undertook to examine, under the sponsorship of the Department of Energy and DARPA, the utility of cellular automata in physical science calculations, especially in fluid dynamics. We expanded the scope of our study to include several related topics which we concluded would be of some interest to our sponsors. These include: (1) a comparison of cellular automata (CA) techniques to "conventional" methods of solving the partial differential equations of fluid dynamics or other physical situations; and (2) the utility and status of using parallel or concurrent processing machines for doing either CA or conventional fluids calculations.

To assist our study we hosted a series of external briefers who kindly gave of their time and expertise. On the subject of CA and applications to fluid dynamics we heard from Dr. Jay Boris of the Naval Research Laboratory who spoke on "Cellular Model for Tasking and Correlation," Dr. T. Toffoli of MIT who spoke about "Primitives of Computation and of Physics as Applied to Cellular Automata," Dr.Gary Doolen of the Los Alamcs National Laboratory who addressed us on the topic of "Lattice Gases," Dr. S. Wolfram of the University of Illinois speaking on "CA and Hydrodynamics," Dr. S. Omohundro of Illinois speaking on "Applications of the Connection Machine Architecture," Dr. B. Nemnich who spoke on "The Connection Machine," and Dr. P. Collela from the Lawrence Livermore National Laboratory speaking on "Multiple Scale Problems in Hydrodynamics." On the subject of parallel processing we heard, in addition to the talks of Nemnich and Omohundro, from Dr. J. Barhen of Oak Ridge National Laboratory on "Hypercube Computer: Architecture and Algorithms for Advanced Applications," and from Dr. J. Fier of the AMETEK Computer Research Division on "Hypercube Architecture and Applications." To all these workers in the field we give our thanks for their assistance.

The work reported on in the present study was begun at JASON in the summer of 1986. Since that time there has been a considerable maturation of the field of cellular automata. The reader desiring further background may refer to two excellent review volumes:

1. Complex Systems volume 1, no. 4 (August 1987); this volume is based largely on presentations of a workshop held in Santa Fe, NM in October of 1986.

2. Lattice Gas Methods for Partial Differential Equations, edited by G. Dollan et al. (Addison-Wesley, N.Y., 1989).

Our own work has been organized along the following lines:

- We have examined the relationship, in terms of effort and efficiency, of doing fluid dynamics calculations using cellular automata versus more conventional spectral or finite difference methods. We conclude that cellular automata calculations are likely to be competitive with standard finite element or spectral methods for the Navier-Stokes equations primarily for low Reynolds numbers and Mach numbers. An exception to this may occur in complex geometries or with boundary conditions where conventional methods are often quite difficult.
- We have reviewed the derivation of ordinary Navier-Stokes Newtonian fluid dynamics from kinetic theory and compared it to the derivation of fluid dynamics from CA. Only for low Mach numbers do we have some confidence that fluid dynamics is being simulated in these calculations.
- We have looked into two schemes for carrying out CA calculations in three dimensions–one uses the notion of tiling or covering 3-D space with a quasi-periodic lattice of Penrose type, and the other investigates the idea of doing CA computations in dimensions larger than three and then projecting the results back into three dimensions. In each case the issue is achieving enough structure in the underlying covering of 3-D space to assure correct tensorial characteristics of the quantities entering the Navier Stokes equations.
- We have formulated in an abstract fashion the problem of representing in a physical 3-D computational structure 2-D highly parallel CA computations.

Perhaps it is useful to define what we mean by some of the terms used already in this introduction, especially those which will appear throughout the report. First of all, we need to address the meaning of cellular automata.

The idea of cellular automata is that by using very restricted information on the position and velocities of individual particles in their microscopic collision and interaction one can, nonetheless, arrive at a good representation of macroscopic dynamical equations such as the Navier-Stokes equations, since the latter involve averaging over larger numbers of individual particles. Furthermore, the averaging is over space and time scales large compared to the microscopic dynamics, so one might think that a crude representation of the latter could result in realistic and acceptable macroscopic physics. The generality of macroscopic equations such as the diffusion equation or the fluid dynamics equations which are parametrized by a few transport coefficients in which all the microphysics is buried would also support this point of view. The key notion of CA, as opposed to the ideas of molecular dynamics, is to very crudely represent each individual particle motion. Crude here means giving the position on a simple lattice covering the space of interest and giving the velocity as one of a few choices such as plus or minus unity. In other words, by representing the phase space coordinates of individual particles by only a few bits of information, one hopes that the aggregate average needed to construct the macroscopic velocity or density is accurately and efficiently computable.

2 CELLULAR AUTOMATA RULE FINDING FOR USE IN 3-D FLUID DYNAMICS

There is some difficulty in finding simple cellular automata dynamics in three dimensions which adequately simulate the Navier-Stokes equation. From a purely formal (i.e., nonphysical) point of view, there are a number of ways of overcoming these difficulties, and as we will show, many of the purely formal procedures have a straightforward physical interpretation. As a consequence of these investigations, we can present what appears to be the simplest, and most straightforward, cellular automata simulation of 3-D fluid motion.

To begin, whether in two- or three-dimensions, our lattice sites will always be the points with all integer coordinates, i.e., the usual integral lattice. In addition are given a collection of vectors e^1, e^2, ... e^N each vector belonging to the lattice. In the list, a given vector may occur repeatedly. At any instant of time, the total state of our automaton is described by an N-dimensional vector of zeros and ones given for each lattice site. In other words, the state of our automata at time t is described by specifying $A_a(x,t), a = 1,2,\ldots N$, and x running through the lattice sites. For fixed x and t, the vector $[A_1(x,t), A_2(x,t), \ldots, A_N(x,t)]$ is called the state vector at site x, time t.

The evolution or dynamics in our automaton is given as follows:

$$A_a(x, t+1) = F_a, a = 1,2,\ldots N,$$

where each F_a is a fixed function of the state vectors of the automaton at time t at sites in a fixed neighborhood of x. What we mean to say here is simply that the rule for advancing in time is the same function of neighboring states at each site and all times. You may think of the F_a as including both the collision laws and the motion of the more conventional description of cellular automata, but in general they have no such simple physical description.

In order to bring out clearly what the essential properties of the dynamics are, we will write:

$$F_a = A_a(x - e^a, t)/\Omega_a \qquad (2-1)$$

and demand that identically for all total states of the automata at time t, we have:

$$\sum_a \Omega_a = 0 \tag{2-2}$$

$$\sum_a e^a \Omega_a = 0. \tag{2-3}$$

The Ω_a's are simply the measure of the difference in the dynamics of the automaton from straight collisionless motion of the particles. They are rarely mentioned specifically for most of the familiar constructs, but they do play an important formal role, as we shall see momentarily. It is quite easy to see that conditions in Equation (2-2) are met, as a matter of fact, for the familiar square and hexagonal lattice fluid dynamics constructs, without even specifically bringing in the Ω's.

For our original list of vectors e^a, we are going to insist that all the tensors $\sum_a e^a, \sum_a e^a \otimes e^a, \sum_a \overset{a}{e} \otimes \overset{a}{e} \otimes e^a$ and $\sum_a e^a \otimes e^a \otimes e^a \otimes e^a$ be isotropic.

Thus, for example, if we take in <u>two dimensions</u> the 20 vectors: $(\pm 1, \pm 1)$ once each, $(\pm 1, 0)$ 4 times each, and $(0, \pm 1)$ 4 times each, we satisfy the isotropy requirements. In <u>three dimensions</u>, we may take the 24 vectors $(\pm 1, \pm 1, 0)$ once each, $(\pm 1, 0, 0)$ twice each, $(0, \pm 1, 0)$ twice each, and $(0, 0, \pm 1)$ twice each to satisfy the isotropy requirements.

In general, for a given set of e's, it seems quite easy to produce a large number of F_a's satisfying the requirements (2-1) and (2-2) above. Subsequently, we will produce F_a's for the e's described in the paragraph immediately above.

Suppose we assume that our automata locally equilibrate in space. Denote by E the admittedly vague notion of expectation operator for the local spatial equilibration, and put $f_a(x,t) = E(A_a(x,t))$. By virtue of the requirements (2-1) and (2-2), we obtain:

$$\sum_a \mathbf{f_a}(x, t+1) = \sum_a \mathbf{f_a}(x - e^a, t) \tag{2-4}$$

$$\sum_a e^a \mathbf{f_a}(x, t+1) = \sum_a e^a \mathbf{f_a}(x - e^a, t). \tag{2-5}$$

Denote $n(x,t) = \sum_a \mathbf{f_a}(x,t)$ and $n(x,t)\ u(x,t) = \sum e^a \mathbf{f_a}(x,t)$. In the continuum limit of long times and large lattices, the equations above become:

$$\frac{\partial n}{\partial t} + \nabla \cdot nu = 0 \tag{2-6}$$

$$\frac{\partial}{\partial t}(nu) + \sum_a e^a(e^a \cdot \nabla \mathbf{f_a}) = 0. \tag{2-7}$$

It is convenient to refer to $n(x,t)$ as particle density, $u(x,t)$ as average velocity, and nu as momentum density. With this convention the requirements (2-1) and (2-2) provide for conservation of particle number and momentum. For if we pick an initial state for the automaton in which only finitely many of the state vectors are non-zero, we get from (2-1) that

$$\sum_x n(x,t+1) = \sum_x n(x,t)$$

or in the continuum limit

$$\frac{d}{dt}\int n(x,t)d\vec{x} = 0.$$

Similarly (2-2) gives conservation of momentum in time.

Analogously, if the initial state for the automaton is periodic in the spatial variables, so also are subsequent states and with the same periods, and the total particle number and total momentum over a periodic rectangular parallelogram or parallelepiped is conserved in time. Imposing periodocity is equivalent, of course, to running the dynamics on a 2- or 3-D torus.

At this point, we make the bold assumption that the functions $\mathbf{f_a}(x,t)$ can be determined from the equilibrium parameters $u(x,t)$ and $n(x,t)$ by a Chapman Enskog expansion. Arguing just as in Wolfram[1] we end with Navier-Stokes like equations for a continuum fluid.

For the rest of this section, we are going to show that, at least in some cases, the formal procedures described above really work. Along the way, we will produce a particularly simple way of running 3-D fluid dynamics at the cellular automata level.

We will begin with a 4-D cellular automaton with the lattice sites being as usual the points with all integral coordinates. Lattice points are labelled (x,y,z,w). We take the sublattice of integral points for which the coordinate sum is even, which is of index 2 in the full integral lattice. This lattice is connected with a very interesting tessalation of the 4-D space, but this does not concern us here. In the sublattice we take all points of least nonzero distance from the origin, these being precisely the 24 vectors: $(\pm1, \pm1, 0, 0)$ and its permutations. These 24 vectors are the list $e^a, a = 1, 2, \ldots, N = 24$. It is easy to verify that the requirements of isotropy are met by this list, that is, for example:

$$\sum_a (e^a)_i (e^a)_j (e^a)_k (e^a)_\ell = \delta_{ij}\delta_{k\ell} + \delta_{ik}\delta_{j\ell} + \delta_{i\ell}\delta_{jk}.$$

There are a number of available scattering laws to give a good momentum scramble, e.g.:

A) Binary: (1, -1, 0, 0) + (-1, 1, 0, 0) $\leftrightarrow$ (0, 0, 1, -1) + (0, 0, -1, 1)

B) Ternary: (1, -1, 0, 0) + (0, 1, -1, 0) + (-1, 0, 1, 0) $\leftrightarrow$ (-1, 1, 0, 0) + (0, -1, 1, 0) + (1, 0, -1, 0)

C) Ternary: (1, 0, 0, -1) + (1, 0, 0, 1) + (-1, 1, 0, 0) $\leftrightarrow$ (0, 1, 0, 1) + (0, 1, 0, -1) + (1, -1, 0, 0) and

D) Ternary: (1, 0, 0, -1) + (1, 0, 0, 1) + (-1, 1, 0, 0) $\leftrightarrow$ (1, 1, 0, 0) + (0, 1, 1, 0) + (0, -1, -1, 0)
and so avoid undesirable conservation laws. If we run a 4-D cellular automaton with a suitable collection of such scattering laws on the integral lattice, there is however one inevitable conservation law, which will not concern us in the applications we make. This arises simply from the fact that the dynamics takes place in two uncoupled sets, namely the sublattice of points whose coordinate sum is even and the coset of points whose coordinate sum is odd.

The application of this 4-D cellular automaton to 3-D problems is now quite straightforward. Take any initial state in the 3-D section (x,y,z,w = 0), and extend it to 4-D by repeating it in every section w = k, k an integer. If we have deterministic scattering laws, then throughout the evolution all sections w = k remain the same as the section w = 0. [Even if we have probabilistic scattering laws, we can maintain the identity of sections by insisting that the choice made at a site (x,y,z,0) in the section w = 0 be repeated at all sites (x,y,z,w)].

All sections remain the same, thus macroscopic particle density, momentum density and average velocity vector are independent of w. since we get the same average over a region and any translation of the region along the w-axis.

The even sublattice and its colattice are now coupled together because the sites (x,y,z, even) and (x,y,z, odd) have the same state.

If we now look at the 4-D Navier Stokes equation which our 4-D automaton is presumably simulating, then since density and momentum density are not functions of w, the projection πu of the velocity vector u into the section w = 0 together with the original density n, satisfy a Navier Stokes like system in 3-D.

Since the sections w = k remain the same throughout the evolution of the 4-D automaton, the dynamics can be fully described in the section w = 0. How do we do so? We need to describe the state at each 3-D site with a 24 bit vector, corresponding to presence or absence of a particle headed in direction $e^a; a = 1, 2, \ldots, 24$. Since we will be concerned only with the projection of the 4-D momentum into the section w = 0, we project the 24 vectors e^a to get:

$(\pm 1, \pm 1, 0)$	once each
$(\pm 1, 0, \pm 1)$	once each
$(0, \pm 1, \pm 1)$	once each
$(\pm 1, 0, 0)$	twice each
$(0, \pm 1, 0)$	twice each
$(0, 0, \pm 1)$	twice each.

The versions of the projected vectors which occur twice are to be labelled; we call one spin plus, the other spin minus, depending on the sign of the invisible 4th coordinate. The scattering laws such as (A), (B), (C), (D) described earlier are replaced by their versions with the last coordinate suppressed, but the vectors therein labelled spin plus or minus as required. The 3-D motion after scattering takes place along the 24 projection of the vectors e^a.

Here we have precisely what was described only as a possibility earlier, namely a 3-D Navier Stokes simulation using some vectors e^a repeatedly. The 3-D isotropy is obvious.

The 2-D simulation using 20 vectors, described earlier, can be simulated by projecting once more, into the section z = 0. It is relatively easy to see that the four rest particles arising from the projection of $(0,0,\pm 1)$, can be dispensed with altogether, but we do not go into details.

As matters currently stand, we can run our 3-D Navier Stokes simulation with a 24-bit vector to describe the state at each site. We want to argue now that 24 can be reduced to 18. To do so, we return to our 4-D simulation, with all sections w = k, k an integer, the same. Suppose for the moment that the state vector at each site in the section w = 0 is reflectively symmetric. By this we mean that if one of the particle directions such as (1, 0, 0, 1) occurs, so also does $(1,0,0,-1)$. This being the case, the state vector can be described by an 18 bit vector. Suppose also that the scattering rules we choose to use are reflectivity symmetric. By this we mean that for every scattering law, the law obtained by changing the signs of last coordinates is also a scattering law we use. (With some care, probabilistic scattering laws can be handled). All this being so, it is easy to see that in the evolution of the 4-D automata, the section w = 0 remains reflectively symmetric, and so the 3-D dynamics needs only 18 bits per site. At the same time, we must assure ourselves that we have enough applicable scattering laws to avoid undesirable conservation laws. It is clear that we will not succeed by using only those vectors which occur singly. But the purpose of our writing down scattering laws (C) and (D) earlier was to persuade the reader now that enough scattering survives, these being two instances of scattering laws in which the spin plus and spin minus directions occur in pairs. Actually, a binary law, such as (1, 0, 0, +1) + (1, 0, 0, -1) $\otimes$ (1, 1, 0, 0) + (1, -1, 0, 0), will eliminate unwanted conservations.

It is interesting to note that in running a 4-D reflectively symmetric cellular automaton with all sections w = k the same, the macroscopic momentum density has no component in the direction of the w-axis.

Now that we are done describing our 3-D simulation, it all seems quite trivial. We have simply reversed a standard device in fluid dynamics. If, for example, one wishes to describe a 3-D planar flow about a cylindrical obstacle, one reduces to a 2-D problem. We have, perversely, taken a 3-D problem and imbedded it in a 4-D hyperplane flow setting, to gain the advantage of the useful lattices existing in four dimensions.

There is another 4-D sublattice of the integral lattice which offers some promise. Essentially the dual of the sublattice considered earlier, it consists of points all of whose coordinates are even, or all of whose coordinates are odd. It is of degree 8 in the full integral lattice. The vectors to consider in this instance, 24 in number also, are

$$(\pm 2, 0, 0, 0) \text{ and its permutations and}$$

$$(\pm 1, \pm 1, \pm 1, \pm 1).$$

These vectors give the desired isotropy, and projecting into w = 0 gives two rest particles, 8 particle directions which occur twice, and 6 particle directions occurring once. The rest particles can be eliminated; running the remainder demanding reflective symmetry as before will give us a 3-D simulation requiring a 14 bit vector to describe the state at each site. Scattering laws such as

$$(1111) + (111-1) + (-2000) \leftrightarrow (-1111) + (-111-1) + (2000)$$

will give us some momentum scramble, but we do not appear to have an analogue of the very useful scattering law (D) used in the earlier lattice. Without such analogue, the total number of particles in all of the directions given by the 8 paired spin plus and spin minus directions is conserved, and this gives us a conservation law. The use of higher order scattering laws can eliminate conservation.

The fact that the 3-D dynamics takes place in 4 uncoupled colattices presents no difficulties; restrict one's attention to the sublattice: all coordinates even, or all coordinates odd.

It is appropriate to describe at this point what we feel is the right way to search for conservation laws. Recall that there is an assumption underlying the whole discussion of cellular automata, namely that particle number and momentum are the only conserved quantities.

Now we suppose that the automaton dynamics is broken into two parts, the applications of the scattering laws followed by the lattice motions. At an instant in time we have a state vector $A(x,t)$ which after scattering becomes $A'(x,t)$, followed by motion to neighboring sites, so that $A_a(x, t+1) = A'_a(x - e^a, t)$. Now suppose there is a vector v such that

$$\sum_a v_a A_a(x,t) = \sum_a v_a A'_a(x,t),$$

no matter what x and t, and such that also $\sum_a v_a A_a(x,t)$ is not identically zero. Then we have a particle number conservation law, since now

$$\sum_a v_a A_a(x,t+1) = \sum v_a A'_a(x - e_a, t).$$

Integrating over all space and taking expectations yields

$$\int v \cdot f(x,t) d\vec{x} = \int v \cdot f(x,t+1) d\vec{x}$$

which is not the usual conservation law if $v \cdot f(x,t)$ is not a scalar multiple of $n(x,t)$.

A law different from the usual momentum conservation law would arise from a vector v such that

$$\sum_a v_a e^a A_a(x,t) = \sum_a v_a e^a A'_a(x,t).$$

On taking the dot product of the last with a fixed arbitrary vector, we would find a particle conservation law of the sort considered just above, so it suffices to search for particle number conservation laws. As noted above, these arise from vectors v orthogonal to $A(x,t) - A'(x,t)$ for all possible x and t.

2.1 3-D Cellular Automata

We will now discuss in greater detail some of the technical issues involved in efficient use of 3-D cellular automata for simulating fluid flow, confining our attention to the two kinds of automata described in Section 2. The first of these is associated with the 4-D lattice of integral points whose coordinate sum is even and for which we pick lattice motions corresponding to the 24 directions $(\pm 1, \pm 1, 0, 0)$ and its permutations, this choice assuring sufficient isotropy of the flow to mimic the Navier-Stokes equations. The second choice is the lattice of integral points for which all coordinates are even or all are odd. In this case we pick 24 lattice motions associated to the vectors $(\pm 2, 0, 0, 0)$ and its permutations and $(\pm 1, \pm 1, \pm 1, \pm 1)$, enough again to insure isotropy.

In both cases we are going to use the 4-D automaton to simulate 3-D problems. This is effected by choosing initial conditions in the section w = 0, and repeating them in all the other sections w = k. If the scattering law used at the site (a, b, c, k) is the same for all k at each fixed instant in time, then all sections remain the same in the evolution of the automaton, and it is enough to simply follow the evolution in the section w = 0.

In order to further reduce the computational burden, the following useful device, already pointed out in Section 2, if a particle is needed in direction (a, b, c, d), there is also one headed in the direction (a, b, c, -d).

For the first lattice mentioned above, this convention permits the state at a 3-D site to be described by an 18-bit vector. For the second, the state at a site requires 16 bits. But since it is easy to see that the effect on the evolution of the automaton arising from the pair of directions $(0,0,0,\pm 2)$, if these are not used in any scattering laws, is irrelevant and does not create any unwanted conservation in the 3-D section w = 0, the number of bits necessary to describe the state at a site can be reduced to 14.

For the second lattice the pair of directions $(0,0,0,\pm 2)$, which look like rest particles from the 3-D point of view, have simply been dropped. For either lattice, a pair of directions such as (a, b, c, 1) and (a, b, c, -1), both of which occur at a site, or neither, is called a "married pair."

For both lattices, a variety of scattering laws are available, much more so than in the simple 2-D hexagonal lattice, even with the restrictions created by the married pairs. The problem, as we see it, is to have a rich enough collection of scattering laws to eliminate any undesirable particle or momentum conservation laws, but not so many that the computational logic at a site becomes unwieldy or too slow.

Part of the difficulty in achieving this goal is a consequence of the observation that some randomization is needed in applying scattering laws. For example, if the pair (1, 1, 0, 0) and (-1, -1, 0, 0) are present at a site, they can be replaced by any other velocity vector pair which sums to zero. How shall the choice be made? Similarly, we may have scattering laws:

(A) $(1, 1, 0, 0) + (-1, -1, 0, 0) \leftrightarrow (1, 0, -1, 0) + (-1, 0, 1, 0)$

(B) $(1, 1, 0, 0) + (1, -1, 0, 0) \leftrightarrow (1, 0, 1, 0) + (1, 0, -1, 0)$.

If all three velocity vectors on the left are present at the site, and none of those on the right, which of the two laws should be applied?

Recall that the ultimate macroscopic features of the flow are to be obtained by averaging over suitable sub-regions of the 3-D grid. If the scattering is very limited, or not always applied when available, then mean-free-paths will be quite long; consequently the suitable sub-regions of the grid will have to be undesirably large. On the other hand, attempting to put in all the scattering available will require intricate combinatorial decisions, and a fair amount of randomization.

For these and other reasons, it is desirable to keep the scattering laws as simple as possible. The simplest scattering laws are, of course, binary scatters, and physically these are the most likely to occur. It is fortunate that for the first of the two lattices described earlier, with 18 bits per site for the state vector, the use of binary scattering only suffice.

We describe the situation here in a little more detail now. We can interchange any of the four following pairs

$$(I)\quad \begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & -1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 1 & 0 \\ -1 & 0 & -1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & -1 & 0 \\ -1 & 0 & 1 & 0 \end{pmatrix},$$

or any of the following three pairs

$$(II)\quad \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 \end{pmatrix},$$

and analogously,

$$(III)\quad \begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{pmatrix},$$

$$(IV)\quad \begin{pmatrix} 1 & 0 & 1 & 0 \\ -1 & 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}, \text{ and}$$

(I) gives us 6 binary scatters, (II) to (VII) give us 3 binary scatters each, for a grand total of 24 scatters. Notice that these scattering laws meet the restriction imposed by the married pairs. It is also possible to verify that these binary laws are sufficient to avoid undesirable conservation laws (as described in Section 2); it is the exchanges offered by (II) through (VII) which mix up the "married pairs" with the "single" velocity vectors.

We have not investigated the question of how many of the binary scatters can be dropped while still avoiding conservation laws, though it is clear that some can be. We could, for example, rather than permit the interchanging of all the pairs of (I), interchange only pair 1 with pair 2, 2 with 3, ..., pair 4 with pair 1. Even with limitations such as this, the question of which interchange is to be made has to be decided with a randomization; if the second pair is present, and not the first or the third, which of the two interchanges should be effected?

It is problematic whether using a thin set of the binary scattering laws, big enough to avoid conservation, will give us results as favorable as using them all. We do not see any particular advantage to limited use, and are going to proceed on the basis of using all the binary laws, and on something like an equal footing.

Now we describe one scheme for using all of the 24 laws of the first lattice. The laws are labeled from one to twenty-four. A random number generator, somewhere on the side, picks a number between one and twenty-four, one for each site, and applies the selected scattering law if it can be applied; otherwise no scattering takes place. The random choice of scatter at each site is followed by motion of particles to neighboring sites, and then repetition of the procedure.

In practice it may be desirable to have the random selection of the integer between one and twenty-four other than uniform. Since there are a very large number of fast methods for generating random numbers with frequencies, we will not go into these questions here. There is, nonetheless, quite a large bit of randomization—one generator for each site. Possibly the randomization can be carried out intrinsically as follows: use the bits, and their complements, in the far field of a site to generate the random number at the site. There are obvious risks and deficiencies in this procedure, however.

For some purposes, the method outlined above will not effect scattering with sufficient frequency, for an available scattering law at a site has roughly a probability of 1/24 of taking place. At the cost of slowing the procedure down, a straightforward alternative is available as follows. After the first choice of scattering law is made and applied if possible, and before the particle motion to neighboring sites, a second independent choice is made of binary scatter and applied if possible. This may be repeated as many times as necessary before motion. An attractive feature of this scheme is that the number of repetitions may be settable by the program. Whether the scheme is any better than running the simulation faster with only one randomization per motion depends on the time trade-off for scattering steps compared to motion steps.

There is still another way of proceeding which in a sense eliminates the need for randomization. One may simply choose, once and for all, at each site in the field one of the 24 binary scattering laws, the choice to be made as randomly as possible. This is probably the cheapest and fastest way to simulate, though it might be useful for some purposes if the fixed random allocation of scattering laws to sites could be easily changed.

So much for the first lattice. The second lattice has a smaller state vector at the site, but is considerably poorer in scattering laws. With the restriction imposed by the "married pairs," there are only three binary scatters—interchange any of the following three pairs

$$(I) \quad \begin{pmatrix} 2 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 2 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 2 & 0 \\ 0 & 0 & -2 & 0 \end{pmatrix},$$

none of which mix up married and singles. There are no ternary laws; there are, however, 24 special quaternary laws of which a general example is

$$\begin{aligned} &(1,1,1,1) + (1,1,1,-1) + (-1,-1,-1-1) \leftrightarrow \\ &(2,0,0,0) + (-2,0,0,0) + (0,2,0,0) + (0,-2,0,0) \end{aligned}$$

and these 24 together with the three of (I) eliminate unwanted conservation.

This lattice with the 27 scattering laws above can be used along any of the lines described for the first lattice, though it is not at all clear what the relative frequency of scatters coming from (I) with those coming from (II) should be. In general, especially in instances in which overall density is fairly large or fairly small, it is going to be difficult to use (II) and its ilk. Since without (II) the marrieds and singles are never mixed, it seems clear that somewhat greater emphasis must be put on finding applications of (II).

In the final analysis, the performance in practice of either lattice, along any of the lines described above, must be studied by actual computer simulations. A great deal will depend on the size of the region over which averages are taken in order to obtain macroscopic estimates, on details of the scattering laws used, and on particle density.

3 A VLSI CHIP FOR A CELLULAR AUTOMATA MACHINE

3.1 Introduction

The 4-D scheme discussed in Section 2 above turns out to be very complex. To explain a VLSI circuit that can implement it, we begin by explaining a 2-D square-lattice system first. The scheme conserves particles and momentum. (However, it is not isotropic, and is hence unsuitable for simulating real physics.) The collision rules are a function of the incoming particles from the North (N), South (S), East (E), and West (W) directions, a random collision bit (C) that causes a collision to occur if it is possible to have one, and a random bit (R_N), but we will include this function for possible later use.

The most straightforward method is a simple table look up. Table 3-1 illustrates this. Given a set of particles, a collision bit, and a random bit, the output particles are determined.

Table 3-1

INPUT NSEW	C	R_N	OUTPUT NSEW
0000	-	-	0000
0001	-	-	0001
0010	-	-	0010
0011	0	-	0011
0011	1	-	1100
0100	-	-	0100
0101	-	-	0101
0110	-	-	0110
0111	-	-	0111
1000	-	-	1000
1001	-	-	1001
1010	-	-	1010
1011	-	-	1011
1100	0	-	1100
1100	1	-	0011
1101	-	-	1101
1110	-	-	1110
1111	-	-	1111

Now consider a 'factored' solution to the same problem. Only head-on collisions can result in scattering and if scattering is to occur, then there must be an output channel for each particle to occupy. Thus there are two 'scattering' rules:

1) $N \cdot S \cdot \bar{E} \cdot \bar{W} \cdot C \rightarrow E \cdot W$

2) $\bar{N} \cdot \bar{S} \cdot E \cdot W \cdot C \rightarrow N \cdot S.$

The first rule can be read as

"If there is a particle from the North, and a particle from the South,
and no particle from the East, and no particle from the West, and a collision is to occur, then send a particle out to the East and send a particle out to the West."

The enabling condition parts of these rules are simply implemented by 'and' gates.

Next we must decide which rule to employ if both should be enabled. (Of course, in our example this can never actually occur, but it will occur in more complex systems.) We will employ a priority encoder to choose a rule, with a dynamic priority assignment by the random bit.

Finally, the chosen rule will select which output channels to block and which output channels to inject particles into.

The complete circuit is illustrated in Figure 3-1. It is, of course, more complex than a simple table look-up approach for this very simple system. However, it illustrates the method we intend to use to implement the 4-D cellular automata scheme.

3.2 Computational Cell for 4-D Rules

The 4-D case has a total set of 24 incoming particles but some (6) are paired, so only 18 bits are needed to specify the incoming and outgoing particles. In addition there are 66 rules, so that seven input priority specification bits are needed to randomize rule selection. These will be decoded to the one in 66 priority selection bits. The 'C' bit (C) suffices to determine if an allowed collision will occur.

The straight-forward table look-up scheme will no longer work efficiently as a table of $18 * 2 ** (18 + 7 + 1) \cong 10^9$ bits is needed. So we will employ the 'factored' approach illustrated above. The computational node to accomplish this is shown in Figure 3-2.

The rule enabling conditions will be calculated in the 'AND-plane' of a PLA (programmed logic array). Thus we input the collision enabling bit and all of our particle signals, as well as the logical complement of each. This is some 38 total input signals. The result is 66 rule enable requests (R). Figure 3-3 illustrates the connections for the 'AND-plane.' From Figure 3-3, it is easy to count that there are 282 transistors needed to implement the connections. A few more ($\sim$ 100) will also be needed to function as inverters, drivers, etc., for a total of about 500 transistors.

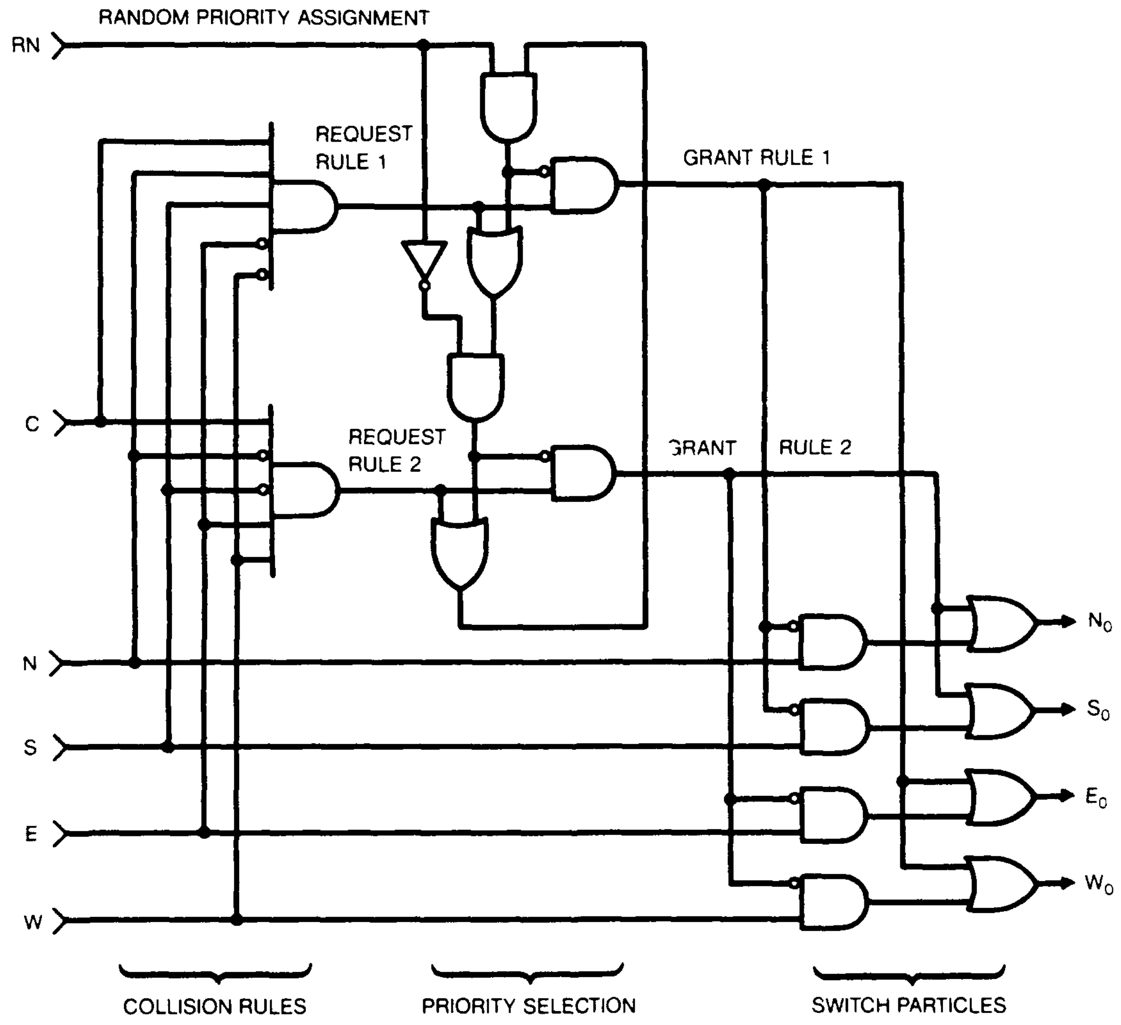


Figure 3-1. "FACTORED" Computation Cell.

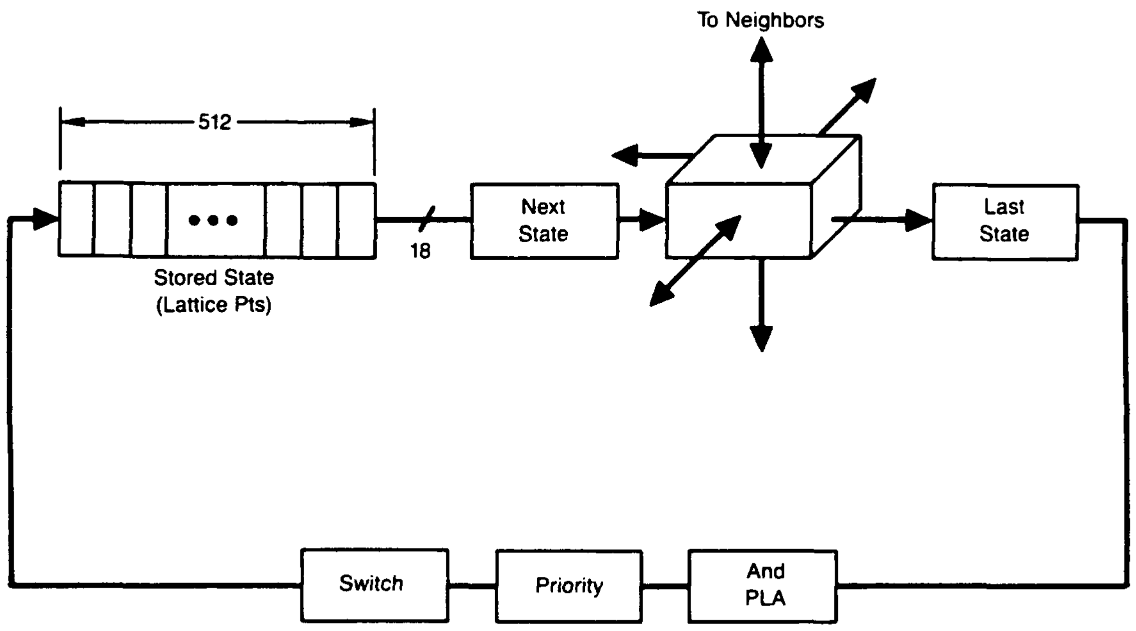


Figure 3-2. A computational node for the VLSI chip.

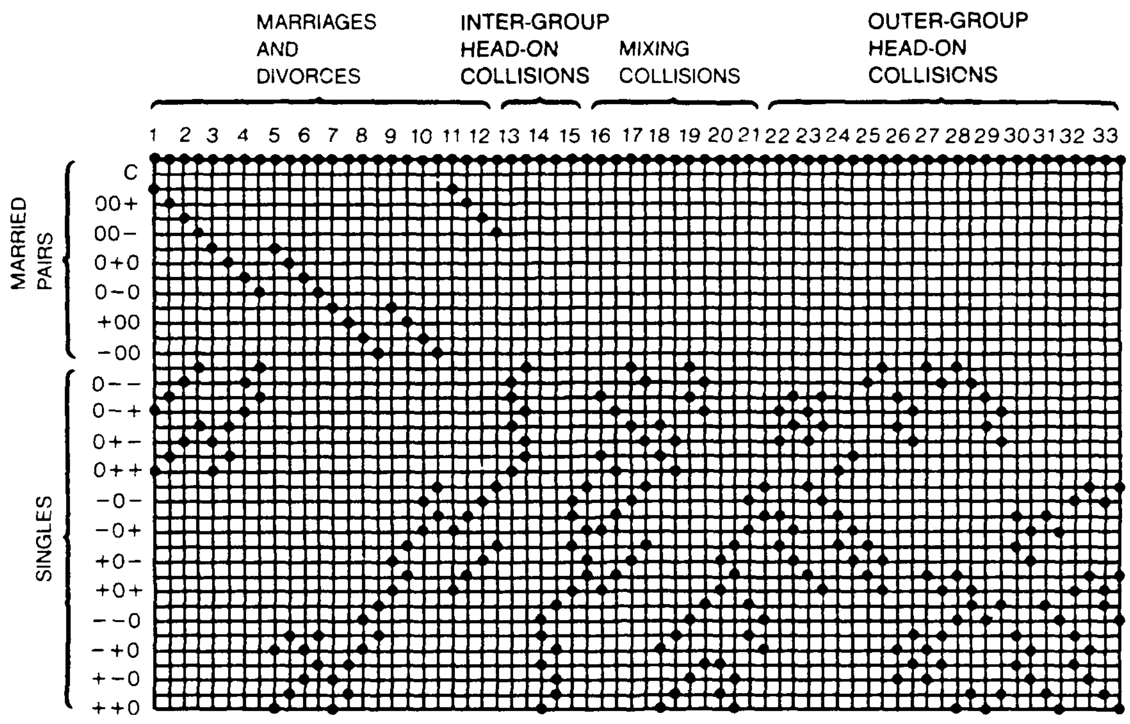


Figure 3-3. 'AND-plane' of Rule Enable Signal Generation Circuit. (The complement of each input signal appears in the row next to the signal.)

The next step is to select (randomly) a rule to fire. We will decode the seven random input bits into 66 signals to control the 'AND-gates' of the priority circuit. This will require about 500 transistors.

If a simple extension of the priority circuit of Figure 3-1 is employed, it would require three gates and two inverters per stage, or about sixteen transistors per stage. This would be about 1000 transistors for the simple priority circuit. Such a priority circuit would be far too slow, and if pipelined would have too much latency. So a 'carry-look-ahead' scheme will be employed. It will require about 1200 transistors total if full look-ahead is employed using a gate fan of about 4. This combined with modest pipelining would require about 1500 transistors total. So we will assume this number.

The output switch circuitry will require 36 gates or about 150 transistors.

Thus we see in summing up that the factored circuitry will require about 3000 transistors per computation node.

If one-half the chip is dedicated to computation circuitry and one-half to memory (to store the virtual node state) then about 8 × 8 (64) real computation nodes/chip is about the limit that could be expected for near-term VLSI technology (see Table 3-2)

Table 3-2

VLSI TECHNOLOGY (CMOS)

Today	in about 5 years
~ cm^2 active area	~ cm^2
~ 200 pins	~ 400 pins
~ 1 Mbit DRAM	~ 4 Mbit DRAM
~ 50K 'Random' transistors	~ 200K Tran.
~ 10 nsec internal clock	~ nsec
~ 80 nsec external drive	~ nsec

These nodes are laid out as illustrated in Figure 3-4.

The performance of the proposed 'lattice gas computer' is now easily estimated (see Table 3-3).

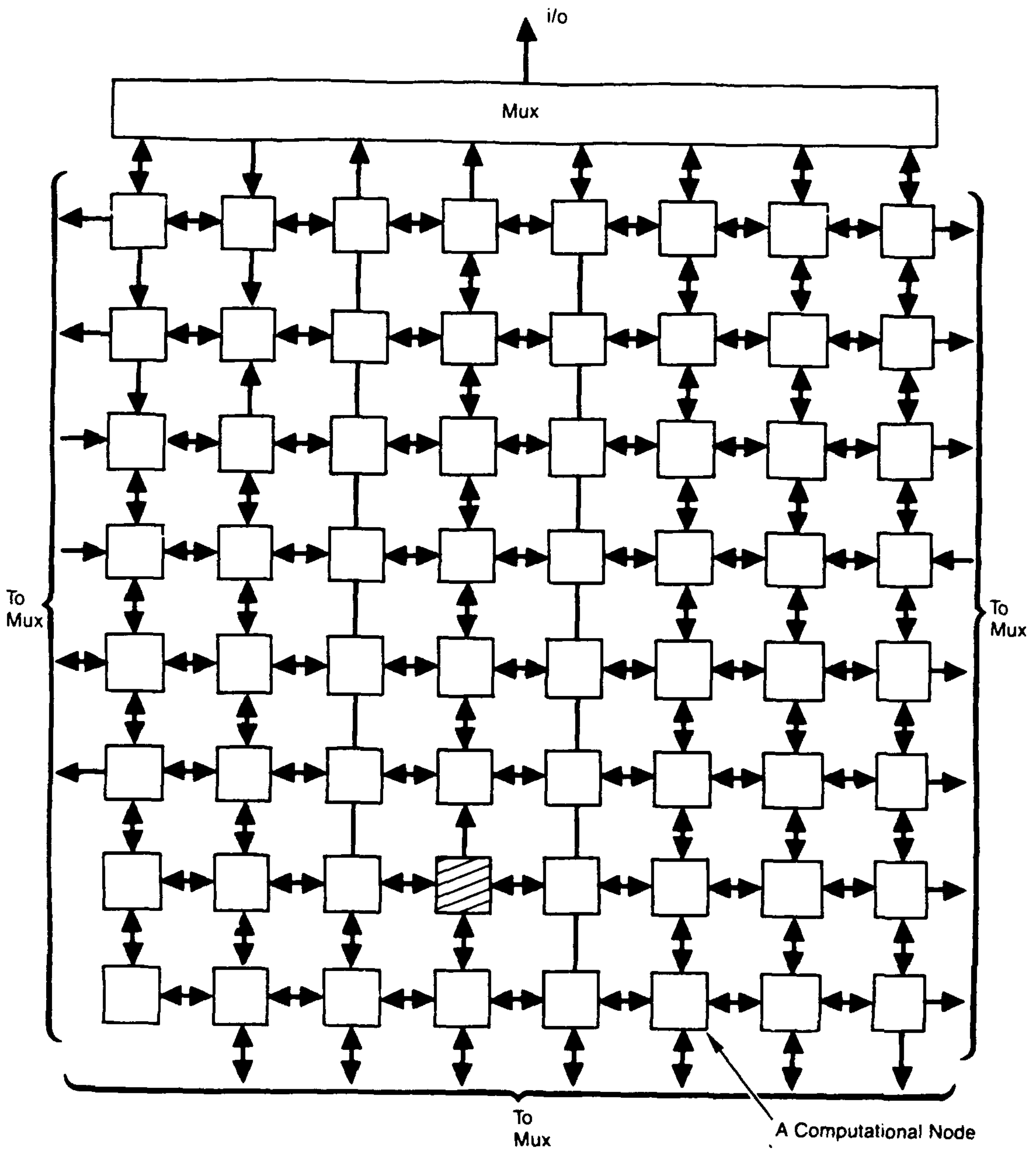


Figure 3-4. Layout of nodes on the VLSI chip.

Table 3-3

PERFORMANCE ESTIMATES

Present Day	Future (5 years)
512 lattice points	
64 nodes/chip	
10^6 chips	
32K lattice points/chip	
3×10^{10} lattice points TOTAL	$\approx 10^{11}$
10 nsec update rate	
64×10^8 updates/chip/sec	
10^{16} updates/sec	$\approx 10^{17}$

About 10^6 VLSI chips is the limit for a single computer system (modern large scale supercomputers have about 3×10^5 chips). At 512 lattice points per node, 64 nodes/chip and 10^6 chips, up to 3×10^{10} lattice points may be feasible. With a 10 nanosecond update rate, 64×10^8 updates per chip and 10^6 chips yields an update rate of $\sim 10^{16}$ per second.

Advances in VLSI technology over the next five years will allow ≈ 3 times the number of lattice points and ≈ 10 times the internal clock rate to yield $\approx 10^{11}$ lattice points updated at the rate of $\approx 10^{17}$ updates per second.

4 GENERAL COMPARISON OF CELLULAR AUTOMATA AND CONVENTIONAL FLUID-DYNAMICAL METHODS

Before looking in detail at the computational requirements for cellular automata and conventional fluid dynamics, we first attempt to summarize some of their relative advantages and disadvantages in a more general way.

4.1 Cellular Automata: Potential Advantages

Cellular automata have the nice properties of elegance, in the sense that their microphysics is immediately transparent, and of local simplicity. They are more immediately suitable for parallelization than conventional fluid methods, and they lend themselves to the design of custom computer architectures which may give very large increases in speed relative to conventional multipurpose machines. In addition, because of the "modular" nature of their geometry, they may be able to handle some types of complicated boundary conditions more readily than conventional fluid techniques.

4.2 Cellular Automata: Disadvantages

On the other hand, the macroscopic physics which is being described by the cellular automata model is not always clear. This is the case, for example, for flows with Mach numbers approaching unity or for lattices lacking the appropriate isotropy properties. Additionally, cellular automata cannot be used for hypersonic flows, and they scale more poorly to high Reynolds numbers than conventional methods, as shown in Reference 2.

C.A. calculations suffer with respect to Navier-Stokes equations because of the need to calculate much more than required by the application to fluid flow. Cellular automata provide a "few state" representation of configuration and velocity space, and are basically very simple versions of kinetic theory. Thus one must average over large numbers of individual sites in $\underset{\sim}{v}$ space to construct the macroscopic density, $n\ (\underset{\sim}{x}\ t)$, and mean velocities, $\underset{\sim}{u}\ (\underset{\sim}{x}\ t)$, which enter the fluid equations directly. This problem is shared by all kinetic theories, of course. Indeed, the point of solving macroscopic equations such as those of Navier-Stokes is to first filter out the microscopic spatial and temporal scales, and then solve for n and $\underset{\sim}{u}$.

4.3 Conventional Methods: Advantages

For conventional solutions of the Navier-Stokes equations, the physical assumptions are more immediately apparent. The physical model is also more flexible, since for example additional terms may be added or the magnitude of the viscosity may be changed without altering the underlying spatial or grid structure. Conventional fluid dynamics seems better able to deal with a large dynamic range in spatial or velocity coordinates. This is partially because of the more favorable scaling to high Reynolds number discussed above, and partially due to the ability to use adaptive grid techniques in those locations where high spatial resolution is needed, without having to use the fine grid spacing in those locations where nothing much is going on. They also have the potential advantage that the assumption of incompressibility can be incorporated explicitly in the algorithm if the flow is highly subsonic, leading to a substantial savings in computer time. By contrast there is no comparable technique for taking advantage of incompressibility for cellular automata.

Again this is because C.A. share with kinetic theories or molecular dynamics the feature of calculating all time scales at once.

4.4 Conventional Methods: Disadvantages

Conventional fluid techniques have more complexity and more bits per cell than cellular automata. Per cell, they thus have higher memory requirements. The floating-point operations required for finite-difference solutions of the Navier-Stokes equations take much longer to perform, per operation, than the simple logical operations or table look-ups of the cellular automata case. Conventional fluid techniques are not as easy to parallelize as cellular automata, and they may in some cases be less able to deal with complex boundary conditions.

4.5 Comparison on the Basis of Computational Work

Here we attempt to quantify some of the explicitly computational pros and cons of the two methods. Consider solving the same fluid-dynamics problem two ways: using cellular automata and using conventional methods based on finite-difference solution of the Navier-Stokes partial differential equations. We compare the computational work needed in the two methods. We emphasize 3-D applications, since in our judgment these represent the next major step in computational fluid dynamics over the coming decade.

When the specifics of computer architecture enter our discussion, we shall compare a conventional Navier-Stokes fluid calculation performed on a Cray-2 class supercomputer with a cellular automata calculation performed on a special-purpose massively parallel machine that does not exist today. Some details of this hypothetical special-purpose machine were described in Section 3; others will emerge as the discussion proceeds.

Both numerical techniques involve subdividing the fluid volume of interest into a discrete grid or lattice. First we discuss how many grid or lattice points are needed for the two approaches (Reference 2).

4.5.1 Number of Grid or Lattice Points

Let L be a macroscopic scale length characterizing the fluid-dynamics problem, and let U be a macroscopic velocity. For example L might be the length of an obstruction in the flow, the width of an shear layer in a jet, or the thickness of a channel or pipe through which the fulid is flowing. The cell size for both the conventional and the cellular automata calculations must be clearly be much smaller than L, so as to resolve the macroscopic structure.

Figure 4-1 depicts the relations between the macroscopic scale L, the size of a cell ℓ_c in the fluid code, and the lattice spacing a in the cellular automata model. Because the equivalent of fluid quantities must be determined in the cellular automata model by averaging over the noisy data of many lattice points, it is clear that $L >> \ell_c >> a$.

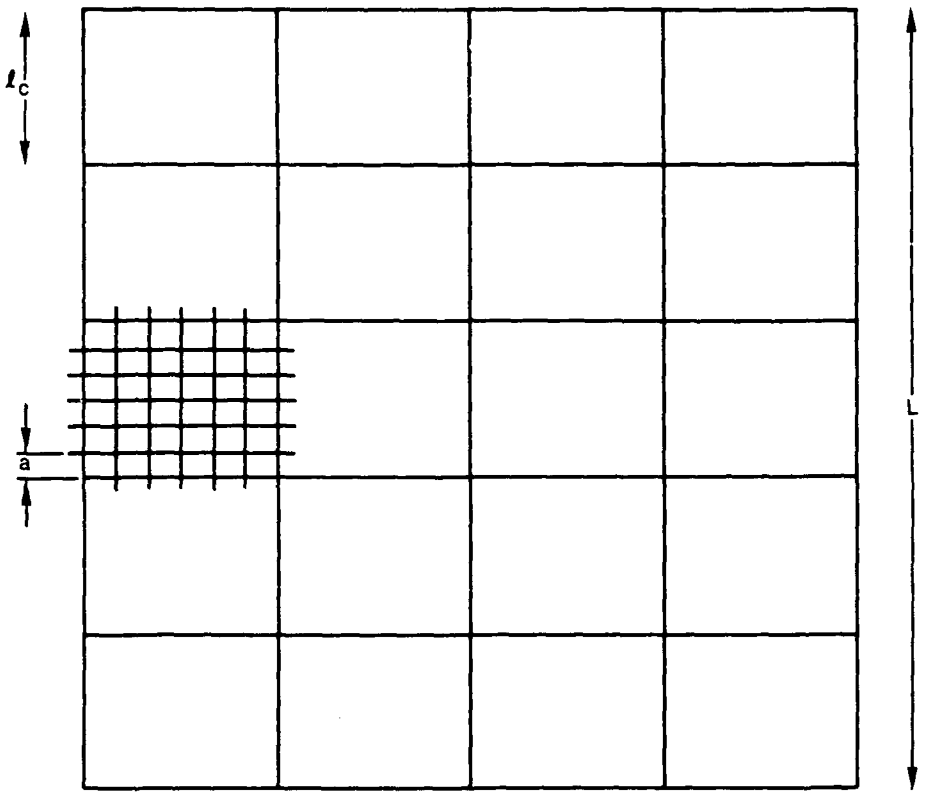


Figure 4-1. Schematic of relations between macroscopic scale ℓ_c for conventional hydrodynamical calculation, and lattice spacing a for cellular automata model.

In a conventional hydrodynamics calculation, the viscosity determines the required size of the cell, because a cell should resolve the Kolmogorov dissipation scale η:

$$\ell_c \approx \eta = (L\, \nu^3/U^3)^{1/4} = L\, Re^{-3/4}, \tag{4-1}$$

where ν is the viscosity and Re is the Reynolds number,

$$Re = U\ L/\nu. \tag{4-2}$$

Thus from (4-1), the number of fluid cells in a length L is

$$L/\mu \approx Re^{3/4}. \tag{4-3}$$

This quantity, $\underline{\text{Re}^{3/4}\text{, represents the dynamic range in scale sizes}}$ which the calculation must resolve, since in any case the finite viscosity will not permit shear flow to develop on scales smaller than η. The $\underline{\text{dynamic range in velocities required in the calculation is of order Re}^{1/4}}$.

From Equation (4-3) we see that in the conventional fluid-dynamic calculation, the number of cells required in three dimensions is approximately

$$N_{\text{cell}}(\text{3D fluid}) = (L/\eta)^3 \cong Re^{9/4}. \tag{4-4}$$

It is well known that this strong scaling of the number of cells with Reynolds number makes it difficult to carry out 3-D fluid-dynamics calculations even at Reynolds numbers of a few hundred. For example if one used a three dimensional grid with 64 grid points on a side (2.6×10^5 total grid points in 3-D), one could by the above criterion do a good job with a Navier-Stokes fluid algorithm simulating Reynolds numbers up to about $(64)^{4/3} = 256$. To simulate a Reynolds number of 1000 would require 178 grid points on a side, or 5.6×10^6 fluid cells in all.

For the celluar automata case, there are several different criteria which must be met in order to produce a physically meaningful simulation of a fluid. These are reviewed in Reference 2. For our purposes the most stringent condition turns out to concern the way in which collisions in the celluar automata model must represent the viscosity of the fluid, when averaged over many cellular automata nodes or lattice points.

In the cellular automata model, all particles move with one discrete velocity (or at most a few). We shall call this velocity v_s. If the mean free path of a cellular automata particles is λ, then the effective viscosity ν which the model will produce when averaged over distances much longer than the mean free path is

$$\nu \cong \lambda\, v_s. \qquad (4-5)$$

The relation between the lattice spacing a and the mean free path λ varies with the collision rules chosen for the cellular automata. In some models the mean free path will be considerably larger than a, if the rules state that a scattering event can only occur when the lattice sites which will be the final states of the collision are initially unoccupied. In other cases the mean free path will be comparable with, or even in some cases less than, a; this may be the situation when rules state that there can be more than one cellular automata particle occupying a given site.

In the discussion to follow, we shall assume that the mean free path is approximately comparable with the lattice spacing, a. When one averages over distances long compared to the mean free path, the macroscopic viscosity ν will be roughly

$$\nu \overset{\sim}{=} \lambda\, v_s \overset{\sim}{=} a\, v_s. \qquad (4-6)$$

Dividing this inequality by a and multiplying by the macroscopic scale length L, this implies that the number of lattice sites in a macroscopic scale length L is

$$L/a \cong\ L\, v_s/\nu = (\frac{UL}{\nu})(\frac{v_s}{U}) = Re/M, \qquad (4-7)$$

where M is the Mach number,

$$M = U/v_s. \qquad (4-8)$$

Equation (4-7) is the cellular automata counterpart of Equation (4-3) for the conventional fluid-dynamics case. The Reynolds number Re is considerably larger than unity in most cases of interest. Additionally, the Mach number M must be small because it is shown in Section 6 of this report that the cellular automata model does not reproduce Navier-Stokes fluid dynamics unless $M << 1$. As a consequence, the scaling for cellular automata given in Equation (4-7) is even more stringent than that for conventional fluid calculations given in Equation (4-3). The number of lattice points required in three dimensions is approximately

$$N_{\text{cell}}\ (3\text{D } C.A.) \cong (L/a)^3 \cong (Re/M)^3. \qquad (4-9)$$

To continue our numerical example, using Re = 256 and Mach number M = 0.2 the number of cells required is 2.1×10^9. A Reynolds number of 1000 would require 1.3×10^{11} cells for a Mach number of 0.2.

Figure 4-2 illustrates the number of cells required for the equivalent Navier-Stokes and cellular automata calculations, at several Mach numbers.

In three dimensions, the ratio of the number of grid points needed for cellular automata and conventional fluid-dynamical calculations of the same problem is

$$(\eta/a)^3 = \left[\frac{L/a}{L/\eta}\right]^3 \cong \frac{Re^{3/4}}{M^3} >> 1. \qquad (4-10)$$

Consider a numerical example just discussed, where the fluid dynamics calculation has 64^3 grid points. The cellular automata model needs a factor of

$$Re^{3/4}/M^3 = 8000$$

more grid points than the fluid dynamics calculation would need. For a Reynolds number of 1000, the cellular automata model would need a factor of 2.2×10^4 more grid points than the fluid model.

4.5.2 Number of Bits per Lattice or Grid Point

The computational effort needed to solve a given problem depends on a number of factors besides the number of lattice or grid points. In the next few subsections we consider these in turn. First, we discuss the number of bits needed at each grid or lattice position, to describe the state of the fluid or cellular automata model. This is ultimately related to the memory requirements. In general cellular automata models will require considerably less memory per cell, but will have many more cells than the conventional models.

For the simplest triangular-lattice cellular automata models in two dimensions, the magnitude of a particle's velocity is always v_s, and after a scattering event each particle goes in one of 6 discrete directions. Thus in the simplest case there are 6 bits per node to keep track of.

However the situation is more complex in three dimensions, since the required lattices may not have simple symmetries (see Sections 2 and 5 of

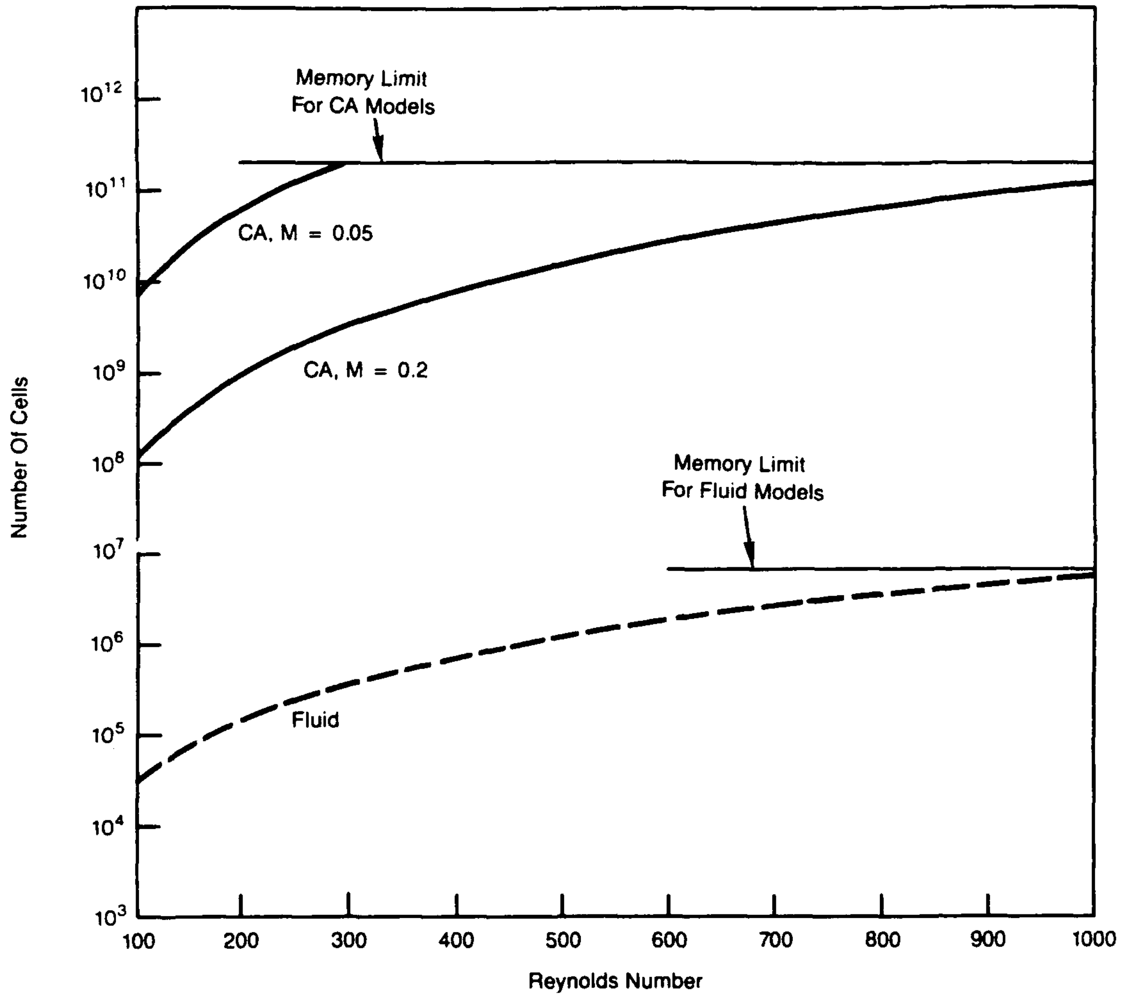


Figure 4-2. Number of fluid cells, N_{cell} (3D fluid), required in three dimensional Navier-Stokes computation, compared with number of cellular automata lattice points, N_{cell} (C.A.) required to perform the equivalent calculation. Memory limits are derived in Sections 4.5.2 and 4.5.3; they correspond in the cellular automata case to a 10^6-chip parallel supercomputer with several megabits of local memory per node, and in the fluid case to a Cray-2 with 64 million words of shared memory.

the present report). The number of bits required per node ranges from 14 at the low end to 30 or 40 at the high end, depending on the lattice geometry and collision rules chosen. For the purposes of this discussion we will estimate that about 20 bits per node are required in three dimensions.

Next we consider the number of bits required at each cell or grid point in a conventional fluid-dynamics calculation, recalling that each fluid-dynamic variable is now typically described by a 64-bit word. Since the Mach number for cellular automata models must be small, it is appropriate to compare these models with the computational requirements of incompressible hydrodynamics. In this case the only independent variables are two components of the velocity, since the third component is determined from div $\underset{\sim}{v} = 0$. The pressure is found from solving a Poisson equation of the form

$$p = Q(v_x,\ v_y,\ v_z). \qquad (4-11)$$

However computationally one must in fact utilize more than just two 64-bit words. For example since one must find the pressure by solving Equation (4-11), the pressure and all three velocity components must be known or calculated at each cell or grid point. A previous JASON report (Reference 3) analyzed one particular 3-D finite-difference technique for solving Equation (4-11), and found that of order 10 words per cell were needed, or approximately 640 bits (if the words were 64 bits each).

Thus very roughly, the ratio of the number of bits per cell or node required for cellular automata and fluid calculations is

$$\frac{\mathrm{N_{bits}(Cellular\ Automata)}}{\mathrm{N_{bits}(Fluid\ Dynamics)}} \cong \frac{20}{640} = 3.1 \times 10^{-2}. \qquad (4-12)$$

As anticipated, there are considerably fewer bits per node required for the cellular automata case. However the <u>total</u> memory requirements for cellular automata may still be more than for the fluid-dynamics calculation, since the number of cells is larger:

$$\frac{\text{Total Memory (Cellular Automata)}}{\text{Total Memory (Fluid Dynamics)}} \cong \frac{(3.1 \times 10^{-2}) Re^{3/4}}{M^3} \qquad (4-13)$$

For our previous numerical examples, $Re = 256$, $M = 0.2$, the cellular automata model requires 250 times more <u>total</u> memory than the fluid dynamics calculation, but only 1/32nd the amount of memory per node. A calculation at a Reynolds number of 1000 require 690 times more total memory for a cellular automata model than for a fluid one.

4.5.3 Number and Speed of Operations per Timestep

Another way to compare cellular automata techniques with conventional fluid dynamics is to look at the required number of operations per timestep, and the speed with which they can be executed.

First we consider the number of operations required by conventional fluid dynamics. These will of course vary with the choice of algorithm. Here we choose one example, which we hope is typical. Reference 3 studied one 3-D incompressible fluid dynamics algorithm in detail, and found the following number of total operations per time step, for a computational grid of N_x, N_y, and N_z zones in the x, y, and z directions:

Additions:

$$N_x N_y N_z (95 + 3 \log_2 N_x N_y)$$

Multiplications:

$$N_x N_y N_z (81 + 2 \log_2 N_x N_y)$$

Memory Transfers:

$$28\, N_x\, N_y\, N_z \qquad (4-14)$$

Continuing our numerical example, if $N_x = N_y = N_z = 64$, corresponding to a Reynolds number of 256, there would be 3.4×10^7 additions, 2.8×10^7 multiplications, and 7.3×10^6 memory transfers per timestep for the fluid calculation. Per fluid cell, there would be 131 additions, 105 multiplications, and 28 memory transfers per timestep. Of course each of the additions and multiplications is a floating-point operation.

Conventional fluid calculations running on a Cray-2 computer in vector mode, with pipelining of memory fetches, would require a clock cycle of about 4 nsec for each of the required adds, multiplies, and memory fetches.

For a 3-D cellular automata model, the number of required operations per timestep is considerably more uncertain, since it depends on the particular choice of lattice and collision rules, neither of which has yet been well explored. Here we consider for the sake of definiteness the "4-D" lattice described in Section 2 of this report. For this lattice, 18 bits at each node are required to describe the "state vector" of the system.

First we ask how complex the collision rules are, and in particular how much space it would take to realize them on a VLSI chip.

The rule function accepts 18 bits to represent incoming particles, utilizes 2 to 4 random bits, and produces 18 bits to represent the outgoing particles after each collision. A straightforward table look-up for the collision rules would thus require a table of about 10^6 entries. Symmetry, however, greatly reduces the functional complexity. A rough estimate based on collision rules designed by Rothaus (Section 2 of the present report) indicates that there are 24 "parallel collision sites" per lattice point. If the rule functions were to be hard-wired, as in a special-purpose computer devoted exclusively to cellular automata, each "parallel collision site" would require about 8 inputs and 4 outputs or about 500 transistors each in a PLA (Programmed Logic Array). Thus the 24 "parallel collision sites" needed for each rule function would require about 10^4 transistors or 1% of the area of a VLSI chip, exclusive of wiring. There could therefore be about 10^2 collision rule calculation engines per chip, and the collision rule calculation could be parallelized by a factor of up to 100.

The limiting number of lattice points per chip is easily calculated by assuming that approximately 20 bits are needed to represent one lattice point. Random access memory chips can presently contain up to 4 x 10^6 bits. So up to 200,000 lattice points can reside in a single chip. Since today about 10^6 chips can be practically assembled into a super computer-scale system, the number of cellular automata lattice points is limited to about 2×10^{11}.

A given chip will have a fixed area, so there is a design tradeoff betwen the number of lattice point states that can be put on a chip. If equal chip area were allocated to each of these functions, we would have 10^5 lattice point states per chip (or 10^{11} total lattice points in our hypothetical 10^6-chip supercomputer), and 50 parallel lattice point engines per chip. According to Equation (4-9), the Reynolds number accessible to cellular automata calculations on this special-purpose computer would then be less than or equal to 1000 at a Mach number of 0.2, and less than or equal to 300 at a Mach number of 0.05. To access Reynolds numbers larger than these values, lattice points could be stored on disc rather than in local memory, and then shuttled in and out of the computational nodes as needed. This use of "virtual memory" would be very costly in execution time, however.

In principle one could also increase the accessible Reynolds numbers somewhat by allocating a larger fraction of the chip area to lattice points, at the expense of parallelism in the rule engines. But since the accessible Reynolds number only increases as the 1/3 power of the number of lattice points, there would not be a great deal to be gained by allocating more than 50% of the chip area to lattice points.

This discussion leads us to a preliminary view of the required characteristics for the special-purpose cellular automata computer which we are postulating. The desire to simulate fluid behavior in three spatial dimensions and at large Reynolds numbers requires a great many lattice points, and implies a massively parallel architecture which nevertheless has a great many (of order 10^5) lattice points at each computational node or chip. This is in contrast to the prevalent practice to date, in which for two spatial dimensions and moderate Reynolds numbers one can devote one computational node to a single lattice point. The cost of placing many lattice points at each computational node is some increased complexity, due to the requirement to communicate to neighboring lattice points which are sometimes off-chip and sometimes on-chip. However, with appropriate lattice layout the bulk of these nearest-neighbor communications will occur within a single chip, and thus the considerably slower off-chip communication times will not be as much of a penalty as they are today.

A second contrast between this type of computer and today's parallel machines is in the sophistication and cost of each computational node. Today, devices such as the "connection machine" endeavor to use computational nodes which are relatively simple, inexpensive, and which have only a modest amount of local memory. The computer which we have envisioned here would require several megabits of local memory at each computational node, and would in addition use custom designed VLSI rule engines at each node to achieve adequate parallelism. These requirements are due to the desire to perform 3-D simulations at Reynolds numbers of 10^2 - 10^3.

4.5.4 Number of Timesteps

The different grid sizes and algorithms will impose different requirements on the number of timesteps required to perform the same physical calculation using cellular automata or conventional fluid-dynamics models.

For cellular automata, there is a genuine Courant condition since "sound waves" can propagate at the velocity v_s. Thus the timestep is limited to the time it takes a sound wave to cross from one lattice point to another:

$$\Delta t_{ca} \cong a/v_s. \qquad (4-15)$$

From Equation (4-7), this is equivalent to

$$\Delta t_{ca} \cong (L/U)(M^2/Re) \qquad (4-16)$$

For incompressible fluid dynamics, there is not a Courant condition because sound waves effectively move instantaneously around the grid. Rather, the timestep is determined by the time for the smallest scale eddies to evolve. Since the smallest spatial scale is the grid spacing $\eta \cong L\ Re^{-3/4}$ and the smallest velocity is the corresponding eddy velocity $v_\eta \cong U\ Re^{-1/4}$, the evolution time for the smallest eddies is η/v_η. Thus the timestep is approximately

$$\Delta t_{\text{fluid}} \cong (L/U)\ Re^{-1/2}. \qquad (4-17)$$

The ratio of the minimum timestep size for the cellular automata and fluid dynamics models is

$$\Delta t_{ca}/\Delta t_{\text{fluid}} \cong M^2\, Re^{-1/2}. \qquad (4-18)$$

Continuing with our numerical example of $Re = 256$ and $M = 0.2$, this would imply that the cellular automata model would need $Re^{1/2}/M^2$=400 times more timesteps to calculate the same physical problem as the fluid dynamics model. For a Reynolds number of 1000, the cellular automata calculation would require about 790 times more timesteps than the equivalent Navier-Stokes calculation.

4.5.5 Ease of Adapting to a Parallel Computation Environment

A final factor influencing the relative promise of the cellular automata and conventional fluid dynamics approaches is the ease with which each can be adapted to a computational environment which is highly parallel. By design, the cellular automata algorithms are well suited to massively parallel computer architectures; one can assign a node or group of nodes to a specific microprocessor and its associated memory. Since we have seen that the memory requirements per lattice point or cell are much less stringent for cellular automata than for conventional techniques, and since only logical operations are required, each microprocessor can be relatively unsophisticated. In addition, the collision rules are designed so that each lattice point need only communicate with its nearest neighbors. Thus the communications overhead is not severe, provided that lattice points which are logically adjacent are also connected by short physical paths (see Section 7 for a more complete discussion of the latter issue).

In the discussion of cellular automata which follows, we shall continue to hypothesize a special-purpose parallel machine which does not exist today but which we believe to be within today's state-of-the-art: a computer of order 10^6 computational nodes, a 1 nsec clock, and about 4 megabits of fast local memory per node. Thus before any additional parallelization at the chip level, one gains a factor of 10^6 over a one-processor machine.

If at the chip level one uses the type of hard-wired rule engine which we discussed in Section 4.5.3, there is an additional gain of a factor of 50 due to the ability to compute collision rules for 50 lattice points at once.

Conventional Navier-Stokes hydrodynamics can also be adapted for paralel computing, although the individual processors must be considerably more powerful than those needed for cellular automata. Typical requirements are 32- or 64-bit words and fast floating-point arithmetic. Many algorithms also need substantial shared memory accessible by all the processors.

For the purpose of the present discussion, we contrast massively parallel cellular automata calculations with Navier-Stokes hydrodynamics computed on a 4-processor Cray-2 scale machine. Experience to date suggests that multiprocessing on such a machine can yield speed-ups in elapsed time of about a factor of 3.6, relative to the same calculation performed on one processor alone.

4.5.6 Overall Comparison of Computational Effort

One can attempt to combine the criteria developed in the preceeding sections into an overall figure of merit for the cellular automata approach, relative to conventional fluid dynamical methods. This figure of merit is in some ways a narrow one, in the sense that it does not take into account practical issues such as mesh tangling, numerical stability, accuracy and noise properties, flexibility, and so forth. What our overall figure of merit does measure is the relative amount of elapsed computer time needed to perform the same physical calculation, using the two methods.

What we describe as the computational effort is an estimate of the overall computer time needed to complete a fluid-dynamics calculation of physical duration L/U, the timescale for macroscopic evolution of the flow. Of course most actual computations will be carried out for physical durations longer than this. But since the number of L/U times required will vary with the specific problem being solved, we shall use L/U as a convenient scaling parameter.

The computational effort thus defined is made up of a series of factors:

$$\text{Effort} = N_{\text{cell}} \times N_{\text{timesteps}} \times (\text{parallelization speed-up})^{-1} \times \sum_{\text{operations}} [\text{operation speed} \times N_{\text{operations}}/\ \text{cell/timestep}]. \qquad (4\text{-}19)$$

The comparison we *shall present is unfair in one important way: it compares* an as yet unbuilt special-purpose parallel supercomputer for cellular automata calculations with a four or five year-old Cray-2 machine for conventional hydrodynamics. In a sense one is comparing computers which are a generation apart, and the reader should be cautioned that this inserts a bias favoring cellular automata algorithms on massively parallel machines.

Table 4-1 summarizes the properties needed for the evaluation of Equation (4-19). We have assumed that Cray-2 memory fetches are pipelined and are thus executable in one clock cycle, and that adds and multiplies are in vector mode.

Inserting the values from Table 4-1 into Equation (4-19), we obtain for the Navier-Stokes calculation on a Cray-2 class machine:

$$\text{Effort (Navier-Stokes)} = 10^{-7} Re^{11/4}(1 + 0.16 \log_{10} Re)\text{sec}, \qquad (4-20)$$

where we have assumed that the adds give the dominant timing. For the special-purpose cellular automata computer, the effort is approximately

$$\text{Effort (C.A.)} \cong 2 \times 10^{-17} Re^4 M^{-5} \text{sec}. \qquad (4-21)$$

Table 4-1

Comparison of Cellular Automata with Incompressible Fluid Algorithmns in Three Dimensions

Property	Cellular Automata Hypothetical Parallel Computer	Imcompressible Fluid, Cray-2 Computer
N_{cell}	(Re/M^3)	$Re^{9/4}$
$N_{timesteps}$	Re/M^2	$Re^{1/2}$
Speedup due to parallelization	10^6 chips $\times$ 50 rule engines/chip $= 5 \times 10^7$	3.6
$N_{operations}$ per cell pet timestop	Rule engine: 1	Memory fetch: 28 Add: $95 + 4.5 \log_2 Re$ Multiply: $81 + 3 \log_2 Re$
Operation speed	1 nsec	4 nsec

The resulting effort values are tabluated in Table 4-2 for several Reynolds numbers and two Mach numbers, assuming that the cellular automata rule-engine parallelism is a factor of 50. The parentheses for higher Reynolds numbers and low Mach numbers indicate that the cellular automata calculation would not fit on a special-purpose computer with 10^6 total chips, if the size of a typical chip is limited to 4 megabits. Similarly, for the fluid case the parentheses indicate that the calculation would not fit in the 64 million word memory of a Cray-2. Under both of these circumstances, virtual disc-based memory could be utilized, but at a substantial degradation in execution speed.

Figure 4-3 illustrates the relation between computational effort and Reynolds number graphically. One sees that for a Mach number of 0.2 and Reynolds numbers between 100 and 1000, the cellular automata model is predicted to execute about three orders of magnitude faster than the conventional fluid calculation. To be sure, we have made several quite idealized assumptions concerning the charactersitics of our hypothetical cellular automata computer. But even if one therefore gives the fluid calculation credit for an additional factor of 10 in speed, due for example to an improved next generation of general-purpose supercomputers, the cellular automata approach appears to be quite promising.

The same cannot be said, however, for the case of Mach number equal to 0.05. Here the two approaches show much more comparable execution speeds, and the cellular automata technique requires so much more memory that it cannot plausibly fit in a 10^6-chip parallel supercomputer for Reynolds numbers larger than about 300. Furthermore, if as before one gives the conventional fluid approach credit for an additional factor of 10 in speed, the fluid algorithms emerge as somewhat superior. The cellular automata approach retains interest primarily in those situations where it has a unique advantage; this might be the case, for example, in studies of fluid behavior within a boundary layer, where the Reynolds numbers are not large and the boundary properties may be complex.

Thus our conclusions regarding the relative execution speeds of the two approaches depend on the range of Mach numbers in which one is interested. If, as suggested in the discussion of the Chapman-Enskog expansion in Section 6, one must limit Mach numbers to very small values in order to assure that the cellular automata model reduces to the Navier-Stokes equations,

Table 4-2

Computational Effort for Navier-Stokes Fluid and Cellular Automata (C.A.) Methods in Three Dimensions

Reynolds Number	Fluid Effort: Cray-2 (sec)	C.A. Effort: Special-Purpose Parallel Computer (sec)	Ratio, Fluid to C.A. Effort
100	4.2×10^{-2}	6.3×10^{-6}	6.7×10^{3}
256	0.58	2.7×10^{-4}	2.2×10^{3}
1000	26	6.3×10^{-2}	4.2×10^{2}
5000	(2.4×10^{3})	(39)	(60)

4-2a. Mach number equal to 0.2

Reynolds Number	Fluid Effort: Cray-2 (sec)	C.A. Effort: Special-Purpose Parallel Computer (sec)	Ratio, Fluid to C.A. Effort
100	4.2×10^{-2}	6.4×10^{-3}	6.5
256	0.58	2.8×10^{-1}	2.1
1000	26	(64.5)	(4.0×10^{-1})
5000	(2.4×10^{3})	(39.9×10^{3})	(60×10^{-2})

4-2b. Mach number equal to 0.05

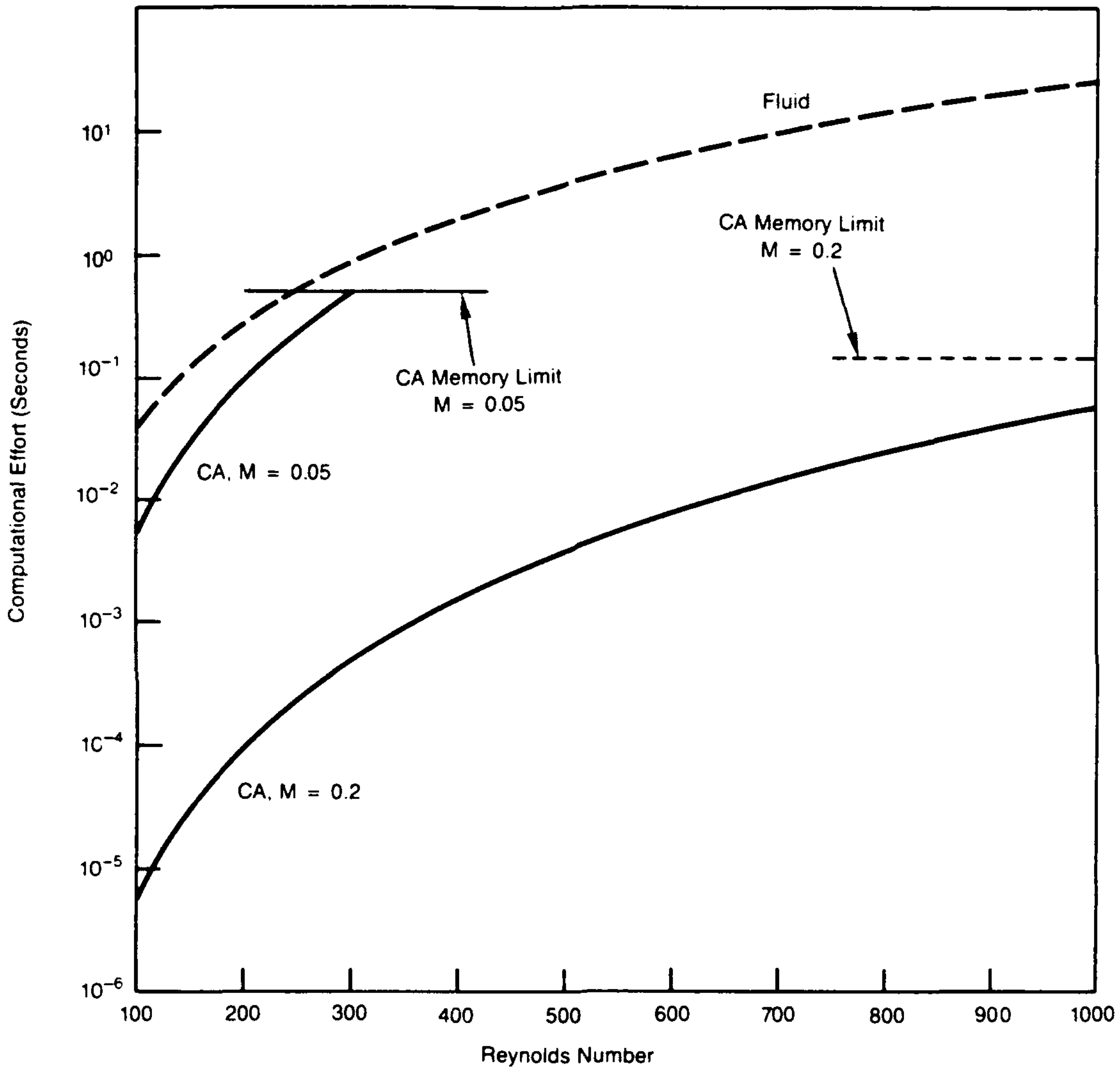


Figure 4-3. Overall computational effort for fluid and cellular automata models, as a function of Reynolds number. In our definition, computational effort is a measure of the elapsed time required to perform a given calculation, exclusive of I/O, averaging in the cellular automata case, and diagnostics. Memory limits for cellular automata are derived in Sections 4.5.2 and 4.5.3, and correspond to a 10^6-chip parallel supercomputer with several megabits of local memory per node.

then for 3-D problems the cellular automata technique does not appear to have notable advantages over conventional fluid techniques. Under these circumstances we do not imagine that there will be interest in developing or purchasing an expensive special-purpose parallel computer for cellular automata work.

It may turn out, on the other hand, that Mach numbers as high as one or two tenths are adequate for assuring the reduction of the cellular automata results to the Navier-Stokes model. Only considerably more experience in doing 3-D calculations will suggest whether this is the case. If it is, then the factors of 100 - 1000 improvement in execution speed suggested in Figure 4-3 for $M = 0.2$ would be a powerful incentive for further pursuing the design and execution of a dedicated massively parallel supercomputer for cellular automata work.

5 QUASILATTICES FOR CELLULAR AUTOMATON FLUID CALCULATIONS

5.1 Introduction

It is possible that lattice models of hydrodynamics with discrete velocities, or "cellular automata fluids," may prove to be an efficient way to simulate complex fluid mechanics problems, at least for virtually incompressible flows at low Mach number. Lattice models were introduced into the physics literature in 1968 by Kadanoff and Swift[4], who were interested in dynamic critical phenomena, and later by Hardy and Pomeau[5], who were interested in fluid properties in general. The recent attention focused on these models was stimulated by the observation of Frisch, Hasslacher and Pomeau[6] that cellular automata on a triangular lattice can provide good approximations to the two-dimensional Navier-Stokes equations of an isotropic fluid. Cellular automata lend themselves to highly efficient parallel processing on digital computers. A discussion of different models and methods of obtaining approximately isotropic equations in the continuum limit has been given by Wolfram.[1]

Cellular automata fluid models usually allow a collection of particles to travel with one of a few, discrete velocities along a specified set of directions or "channels" in two or three dimensions. The directions are specified by a set of unit vectors $\{\vec{e}_a\}$. On the triangular lattice, for example, particles typically travel with unit velocity along one of the six directions connecting a given site to its neighbors. The models determine the time evolution of a set of probabilities $f_a(\vec{x}, t)$, which give the fraction of particles in velocity channel a at position $\vec{x}$ and time t. The number density $n(\vec{x}, t)$, momentum density $g_i(\vec{x}, t)$, and stress tensor $\sigma_{ij}(\vec{x}, t)$ of the fluid are determined from the f_a's via the relations

$$n = \sum_a f_a \tag{5-1}$$

$$\vec{g} = \sum_a f_a \vec{e}_a \tag{5-2}$$

$$\sigma_{ij} = \sum_a e_a^i e_a^j f_a . \tag{5-3}$$

The Navier-Stokes equations may be written

$$\partial_t g_i + \partial_j \sigma_{ij} = 0 \qquad (5-4)$$

where the momentum density is related to the fluid velocity $\vec{u}(\vec{x},t)$ by $\vec{g} = n\vec{u}$. A closed set of equations for $\vec{u}$ results if we can express σ_{ij} in terms of $\vec{u}$. As shown, for example, by Wolfram[1], the Chapman-Enskog expansion in small velocities and gradients for σ_{ij} takes the form

$$\begin{aligned}
\sigma_{ij} &= \frac{n}{b}\left[\sum_a e_a^i e_a^j + e_1 \sum_a e_a^i e_a^j e_a^k u_k\right] \qquad (5\text{-}5)\\
&+ e_2\left(\sum_a e_a^i e_a^j e_a^k e_a^1 - 1/2 \sum_a e_a^i e_a^j \delta_{kl}\right) u_k u_1\\
& c_2'\left(\sum_a e_a^i e_a^j e_a^k e_a^1 - 1/2 \sum_a e_a^i e_a^j \delta_{kl}\right) u_k u_1\\
&+ [\ldots\ldots\ldots\ldots\ldots]
\end{aligned}$$

If the $\{\vec{e}_a\}$ connect nearest neighbors on a triangular lattice, the sums over a in Equation (5-5) are guaranteed to be isotropic up to quantities fourth order in $\{\vec{e}_a\}$[6]:

$$\sum_a e_a^i e_a^j \propto \delta_{ij} \qquad (5-6)$$

$$\sum_a e_a^i e_a^j e_a^k e_a^1 \propto \delta_{ij}\delta_{kl} + \delta_{ij}\delta_{jl} + \delta_{il}\delta_{jk}. \qquad (5-7)$$

Although Equations (5-6) and (5-7) are enough to ensure that the usual truncation of the gradient expansion leading to the Navier-Stokes equations is isotropic, higher order corrections involving sixth rank tensors will be anisotropic. There is, moreover, no simple analogue in three dimensions of the triangular lattice which ensures that even fourth rank tensors will transform properly under rotations[7]. One solution to this problem is to introduce additional directions (beyond nearest neighbors) of propagation on lattices with low symmetries (the (11) direction on square lattices and the (110) and (111) directions on cubic lattices, for example) and adjust the collision rules to eliminate higher order anisotropies. A new variant of this solution was described in Section 2. Another approach, mentioned for example in Reference 1, is to have particles moving on a quasilattice, which can have symmetries which are forbidden on conventional crystallographic lattices.

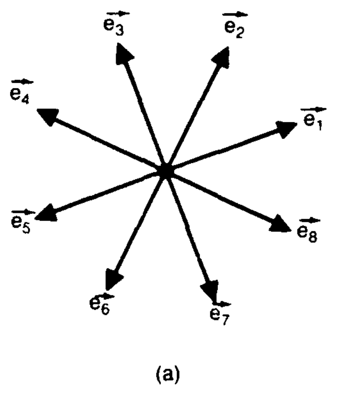
In this section we discuss the latter approach, and show in particular how to construct an octagonal quasilattice in two dimensions which ensures isotropy of all tensors up to eighth order. Particles occupy the faces of a four-dimensional hypercubic lattice, and hop from face to face along a two-dimensional hyperplane which cuts this lattice at a set of irrational angles. When projected into two dimensions, particles flow along eight channels in a way which, despite its apparent complexity, can be simply described using four dimensional integer arithmetic. Momentum is exchanged among the eight channels by allowing special collisions for particles intersecting at right angles, and two distinct velocities. We also discuss a model in three dimensions which allows particles to flow along one of thirty directions corresponding to the midpoints of the bonds of an icosahedron. Among these directions can be found ten groups of six which are coplanar, and point to the vertices of a regular hexagon. This feature allows three-particle collisions and momentum transfer between channels. The model requires only one velocity, and is in many ways reminiscent of a triangular lattice cellular automaton in two dimensions. The icosahedral quasilattice is an irrational projection of a six dimensional hypercubic lattice, and particle positions can be specified using 6d integer arithmetic.

5.2 Octagonal Quasilattices

In Figure 5-1a we show a set of eight basis vectors pointing to the vertices of a regular octagon. One might naively hope to generate a lattice where particles move along one of these eight directions by taking integer linear combinations,

$$\vec{r} = \sum_{i=1}^{4} n_i \vec{e}_i, \qquad (5-8)$$

of the basis vectors $\vec{e}_1$ through $\vec{e}_4$ (the vectors $\vec{e}_5$ through $\vec{e}_8$ are redundant, since they are the negatives of $\vec{e}_1$ through $\vec{e}_4$). This clearly will not work, however, because this set is overcomplete, and the basis vectors have irrational projections on each other: If arbitrary quartets of integers $\{n_1, n_2, n_3, n_4\}$ are allowed in (5-8), we will fill space very inefficiently with an irregular array of points, many of which are almost on top of one another. These difficulties are the basis of a theorem of classical crystallography which states that regular lattices with an octagonal symmetry are impossible. We need to find a way to restrict the integers $\{n_i\}$ so that the "lattice" points are approximately equally spaced.



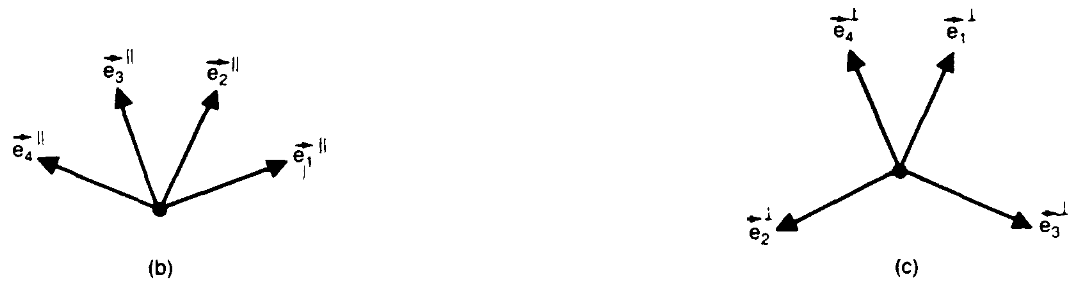


Figure 5-1. Eightfold basis vectors for a two dimensional octagonal lattice. The basis set indicated in (a) can be regarded as the projection of a four dimensional basis set, as shown in (b). The projection of the 4D basis set onto a perpendicular "shadow" plane is shown in (c).

One solution of this problem is embodied in Figure 5-1b, where we choose to regard the basis vectors $\{\vec{e}_1\}$ as a two dimensional projection of an orthonormal basis set in <u>four</u> dimensions. A projection matrix P which acts on unit vectors along the coordinate axes in four dimensions to give the vectors $\{\vec{e}_i^{\parallel}\}$ is

$$P = \frac{1}{\sqrt{8}} \begin{pmatrix} \sqrt{2} & 1 & 0 & -1 \\ 1 & \sqrt{2} & 1 & 0 \\ 0 & 1 & \sqrt{2} & 1 \\ -1 & 0 & 1 & \sqrt{2} \end{pmatrix}. \qquad (5-9)$$

The components of the $\left\{\vec{e}_1^{\parallel}\right\}$ are given by the columns of this matrix. They occupy a common 2-D plane, and it is easily checked by taking dot products that the angles between them are consistent with the octagonal geometry of Figure 5-1a. Let us call this subspace the parallel or "physical" plane. This will be the space occupied by the quasilattice. One can also choose to project normal to this plane into a perpendicular or "shadow" subspace via the matrix $Q = 1 - P$,

$$Q = \frac{1}{\sqrt{8}} \begin{pmatrix} \sqrt{2} & -1 & 0 & 1 \\ -1 & \sqrt{2} & -1 & 0 \\ 0 & -1 & \sqrt{2} & -1 \\ 1 & 0 & -1 & \sqrt{2} \end{pmatrix} \qquad (5-10)$$

and obtain the four alternative projections of the four dimensions basis set labelled $\left\{\vec{e}_1^{\perp}\right\}$ in Figure 5-1c.

The idea behind the projection method for generating quasilattices[8] is illustrated for a 2-to-1 projection in Figure 5-2. Here, a 2-D lattice point (n_1, n_2) is projected if it is at the lower left hand corner of a square which intersects the line ℓ. The sites are projected onto the axis labelled $\vec{e}_{\parallel}$. A useful reformulation of this projection criterion has been given by Elser[8] in terms of the projection of a potential quasilattice point onto the axis labelled $\vec{x}_{\perp}$: A potential site $\vec{n} \equiv (n_1, n_2)$ will be accepted into the quasilattice if and only if its projection $Q\vec{n}$ contained in the shadowed region of the $x_{\perp}$ axis $C_{\perp}(\vec{x}_o)$,

$$C_{\perp}(x_o) = Q\left[-C(\vec{o})\right] + \vec{x}_o, \qquad (5-11)$$

where $Q\left[-C(\vec{o})\right]$ is the set formed by applying Q to the inversion of a cube centered at the origin, and $\vec{x}_o$ a vector marking the intersection of the line ℓ with the $x_{\perp}$ axis (see Figure 5-2). By varying the parameter x_o, we obtain

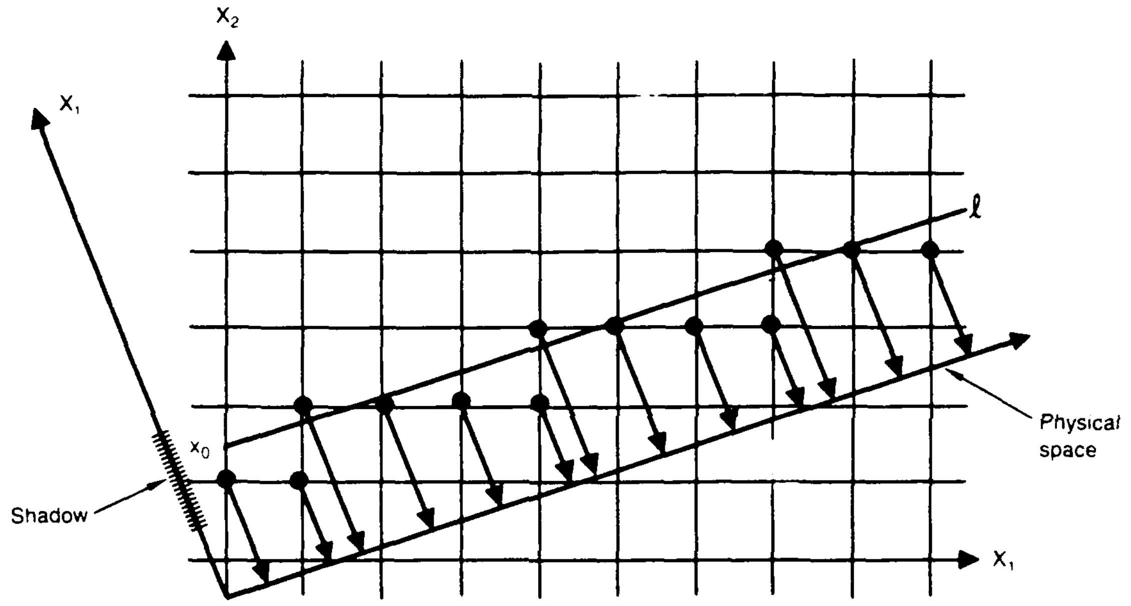


Figure 5-2. Projection ideas illustrated for a 2 to 1 projection.

a continuous family of possible projections. A more formed definition of this set of lattice points (call it $S(x_o)$) is

$$S(\vec{x}_o) = \left\{ (n_1, n_2) \epsilon Z^2 | \overset{Q}{\approx} \vec{n} \epsilon C_\perp(\vec{x}_o) \right\}. \qquad (5-12)$$

This set is indicated by the darkened portion of the $\vec{x}_\perp$ axis labelled "shadow" in Figure 5-2. Its bounded extent along the $x_\perp$ axis ensures that the projected points will not be too close together.

The generalization of these ideas to the octagonal $4 \rightarrow 2$ projection is straightforward. The analogue of the "shadow" $C_\perp(\vec{x}_o)$ in Figure 5-2 is the two dimensional shadow of a 4-D hypercube, which is the interior of a regular octagon. If the 4-D lattice has unit spacing, it can be shown straightforwardly that the distance between parallel faces of the shadow octagon is (see Figure 5-3)

$$d = 1 + \sqrt{2}/2. \qquad (5-13)$$

There is now a 2-D family of possible projections obtained by translating the shadow octagon within the perpendicular plane. The translation vector $\vec{x}_o$ may be conveniently written in terms of the orthogonal vectors

$$|a> \; = \; 1/2\left(1, 0, -1, \sqrt{2}\right) \qquad (5\text{-}14)$$

$$|b> \; = \; 1/2\left(-1, \sqrt{2}, -1, 0\right) \qquad (5\text{-}15)$$

as

$$\vec{x}_o = s|a> + t|b> \qquad (5-16)$$

where s and t are arbitrary real parameters. Note that the transverse projection operator Q may be written

$$Q = |a>< a| + |b>< b| \qquad (5-17)$$

and that $P|a> = P|b> = 0$. Allowed quasilattice positions $\vec{n} \equiv \{n_1, n_2, n_3, n_4\}$ are now determined by the set

$$S(\vec{x}_o) = \left\{ \vec{n} \varepsilon Z^4 | \overset{Q}{\approx} \vec{n} \varepsilon C_\perp(\vec{x}_o) \right\}. \qquad (5-18)$$

It is easy to use these ideas to "grow" an octagonal quasilattice on a computer. A program written in Pascal which runs on an IBM PC is included in the Appendix. The program tests all points of Z^4 inside a large hypercube to see if their projection falls into a particular shadow octagon. If the test

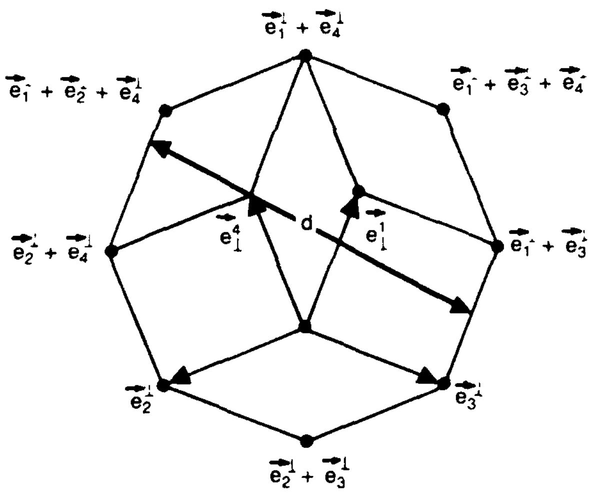

$\vec{e}_1^{\perp} + \vec{e}_4^{\perp}$
$\vec{e}_1^{\perp} + \vec{e}_2^{\perp} + \vec{e}_4^{\perp}$
$\vec{e}_1^{\perp} + \vec{e}_3^{\perp} + \vec{e}_4^{\perp}$
$\vec{e}_2^{\perp} + \vec{e}_4^{\perp}$
$\vec{e}_\perp^4$
d
$\vec{e}_\perp^1$
$\vec{e}_1^{\perp} + \vec{e}_3^{\perp}$
$\vec{e}_2^{\perp}$
$\vec{e}_3^{\perp}$
$\vec{e}_2^{\perp} + \vec{e}_3^{\perp}$

Figure 5-3. Shadow of a four dimensional cube, obtained by applying the transverse projection operator Q to the unit cube consisting of points of the form $\vec{r} = \sum_{i=1}^{4} x_i \vec{e_2}$, where $0 < x_i < 1$.

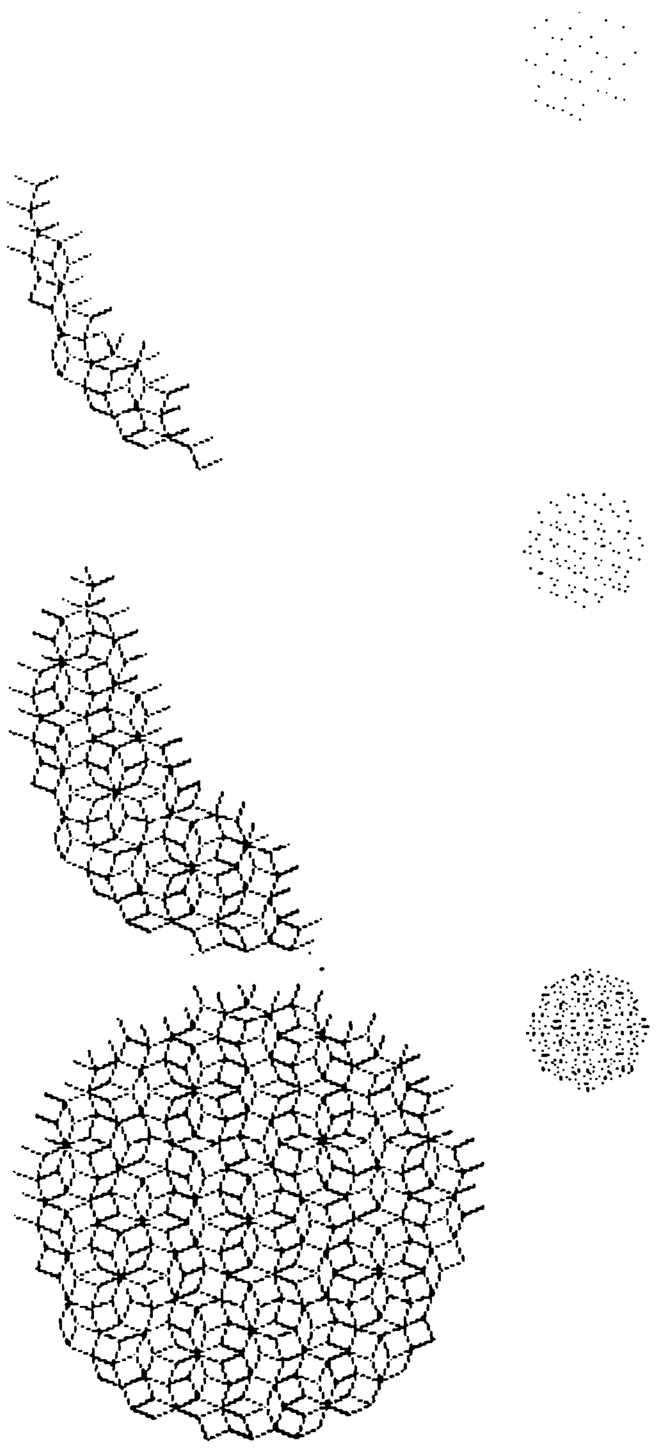


Figure 5-4. Growth of an octagonal quasilattice. There are 321 vertices in the bottom picture. The image of these vertices in shadow space is shown in the upper right.

is succsssful, it then draws the projection into physical space of the lines connecting this point to any nearest neighbors which also happen to be in the set $S(\vec{x}_o)$. A growing octagonal quasilattice is shown in Figure 5-4. The square and rhombic cells in this projection are the images of faces of 4-D hypercubes which intersect the projection plane. Also shown, in the upper right hand corner, are the projections onto the perpendicular shadow space of the quasilattice vertices. It can be shown using the methods of Reference 8 that the shadow octagon is filled with a uniform density of points in the limit of an infinite quasilattice. This feature can be used to determine the frequency and location of various patterns in the quasilattice[8].

One way an octoganal quasilattice could be used in a cellular automaton simulation is illustrated in Figure 5-5. In the absence of collisions, particles in one time step hop between adjacent cells connected by a common set of parallel bonds. Several such jagged particle trajectories are shown in the upper portion of Figure 5-5. Particles moving in this way are defined to have unit velocity in a direction normal to the bonds they are traversing. Binary collisions occur whenever two particles occupy the same cell, as illustrated in the lower portion of the figure. Wolfram has suggested using octagonal quasilattices to improve isotropy[1], but instead of Figure 5-5 draws a picture like that shown in Figure 5-6. The straight lines shown here may be viewed as approximations to the jagged trajectories of Figure 5-5. We find our implementation of the quasilattice idea preferable because particle positions may be indexed using simple 4-D integer arithmetic, rather than solving for the intersections of a set of incommensurate parallel lines. Note also that there are a number of near "coincidences" in Figure 5-6, where three or more lines almost intersect in a point. Such potential ambiguities are resolved automatically in Figure 5-5.

All cellular automaton models with only binary collisions have additional, unwanted conservation laws, in addition to conservation of particle number, energy, and momentum[1]. On an octagonal quasilattice, for example, the momentum contained in particles flowing along each of four directions (each direction corresponding to a pair of channels) is conserved. This is because a binary collision, like that shown in Figure 5-5, extracts a net zero momentum from one direction, and adds a net of zero momentum to another. On a triangular lattice, equilibration of momentum between three different channels is achieved by allowing triple collisions.

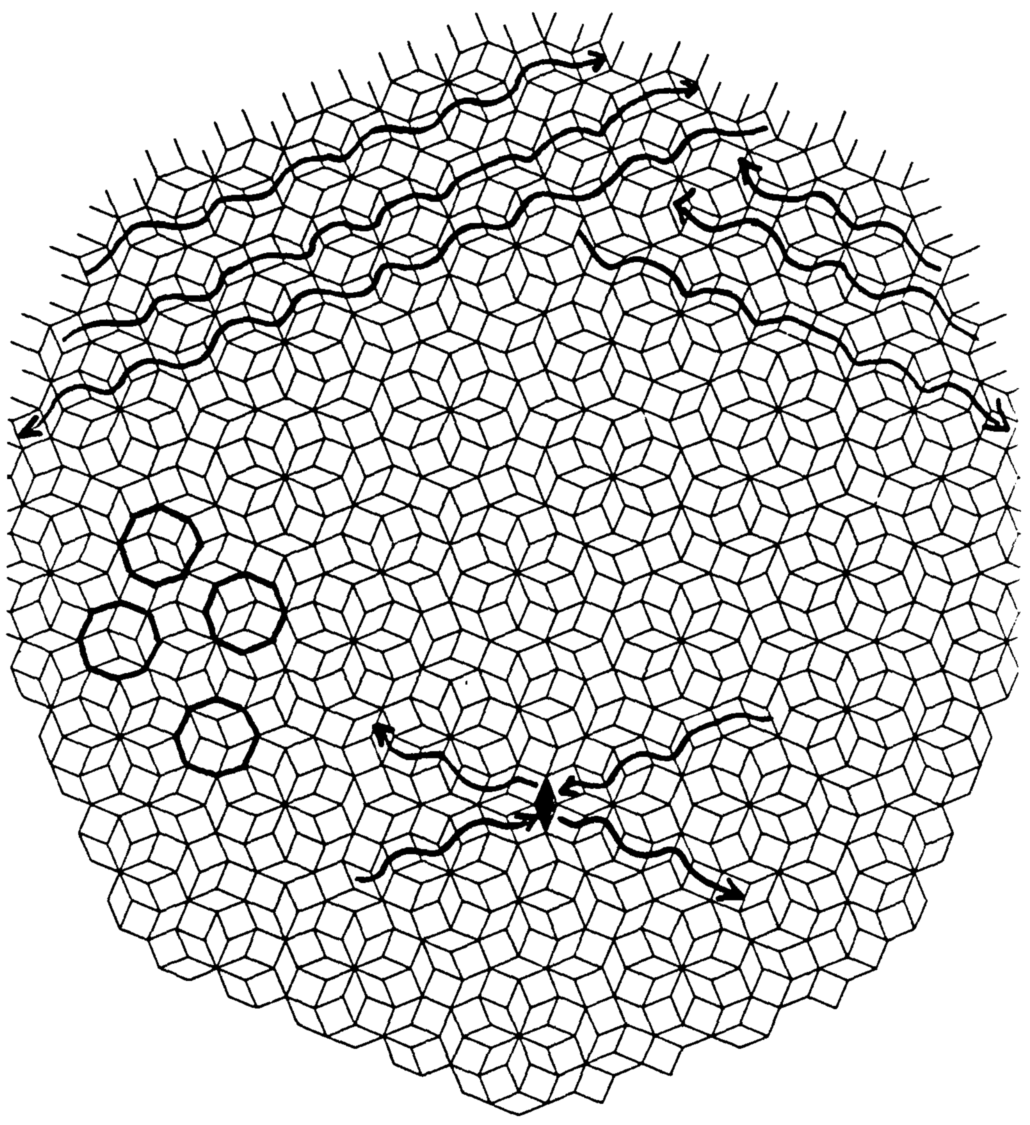


Figure 5-5. Particle trajectories on an octagonal quasilattice. Particles travel in one of eight directions. A binary collision is shown at the bottom. Octagonal figures such as those highlighted on the left side are sites of collisions that transfer momentum between channels.

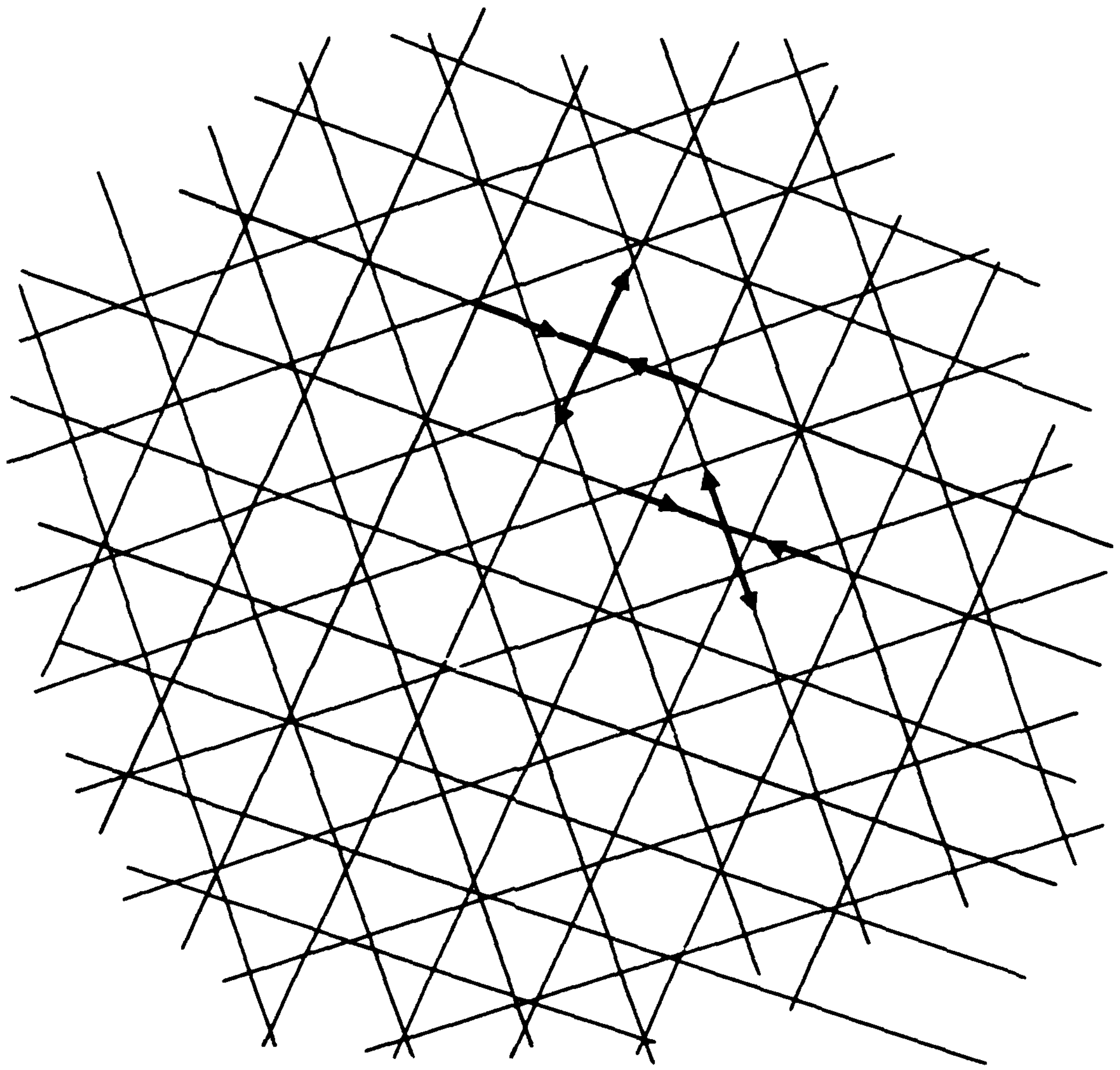


Figure 5-6. Alternative representation of the momentum channels shown in Fig. 5-5.

Figure 5-7 illustrates one way to achieve a similar effect on an octagonal lattice. Particles are now lalowed to have velocities 0, 1, or $\sqrt{2}$. Collisions which transfer momentum between directions occur whenever two particles moving with unit velocity at right angles to each other meet inside an octagon consisting of two squares and four rhombuses. Several such octagons are highlighted in Figure 5-5. Because these octagons have eight symmetrically distributed edges, a representative trajectory from all four possible directions will pass through every octagon. The result of such a collision is defined to be a particle at rest somewhere inside the octagon, as well as a particle moving with velocity $\sqrt{2}$ in the direction given by the vector sum of the two incident velocities. The inverse process occurs when a particle with velocity $\sqrt{2}$ enters an octagon containing a rest particle and produces a pair of particles with velocity 1. Note that energy (velocity squared) is conserved in both cases. Both $\sqrt{2}$ -velocity and unit-velocity particles also have the usual binary collisions indicated in Figure 5-7 as well. The eightfold symmetry of the octagonal quasilattice cellular automaton model insures the isotropy of all tensors with rank six or lower in a Chapman-Enskog expansion of the Navier-Stokes equations.

The image of the bonds encompassing one of the octagons discussed above is an eight pointed star when projected into the shadow octagon (see Figure 5-8). By translating this star around so that it remains completely inside the shadow octagon, one obtains the set of all octagons which could be used as potential collision sites in physical space.

Computer implementations of these are conveniently viewed as shuffling particles around the sites of a regular four-dimensional lattice. The test (5-14) can be used to confine the particles to the "physical" subset of 4-D lattice points. Particles travel on jagged paths in the 4-D space whose projections correspond to the trajectories in Figure 5-5.

5.3 Icosahedral Quasilattices

It is straightforward to apply these ideas in three dimensions and obtain cellular automata with an overall icosahedral symmetry, thus insuring the isotropy of all tensors with rank four and lower. In analogy with Figure 5-6, Wolfram has considered cellular automata models obtained from the

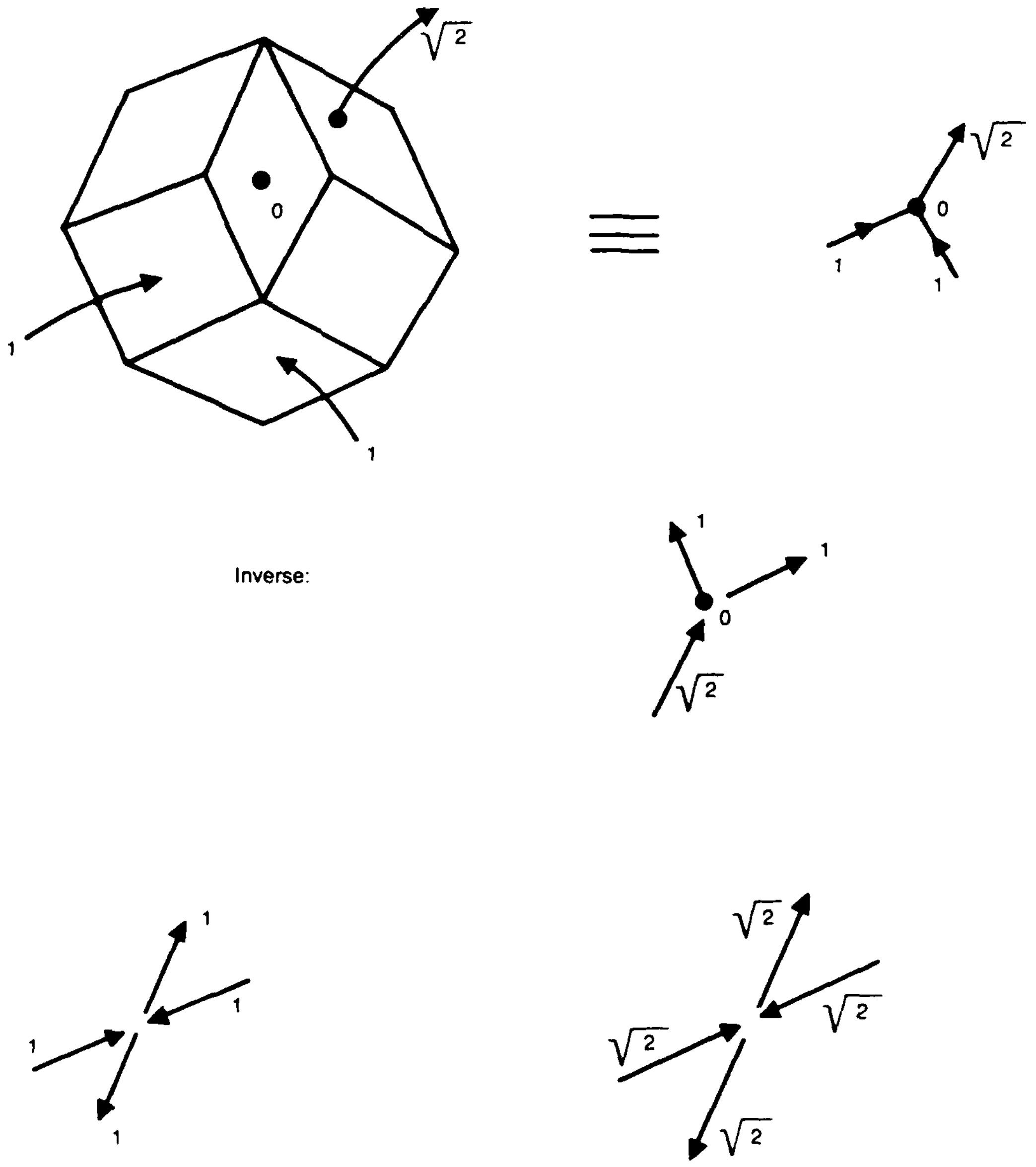


Figure 5-7. Different kinds of binary collisions on an octagonal quasilattice.

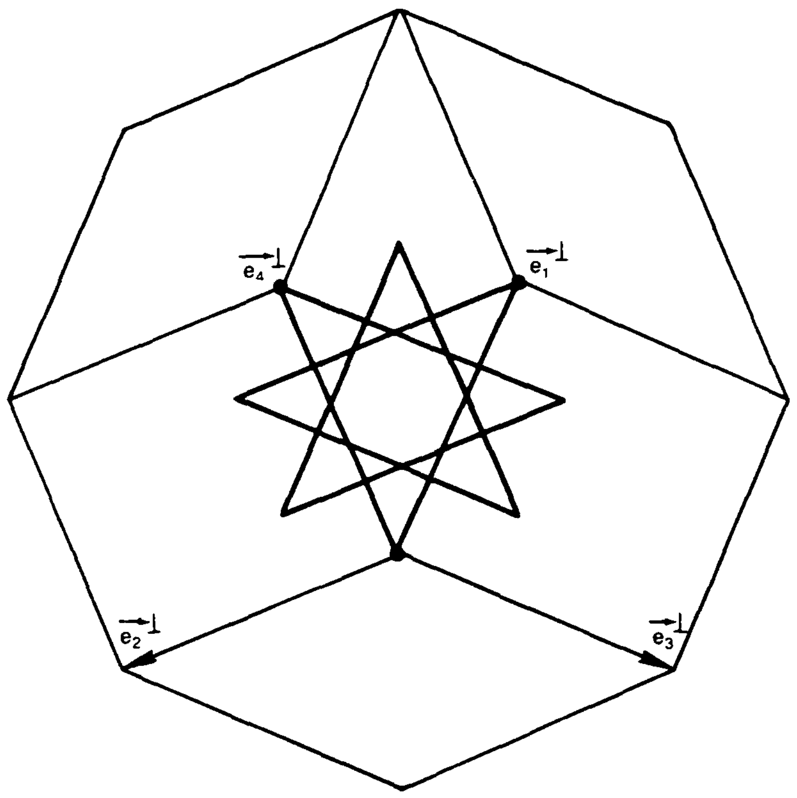


Figure 5-8. Image of an octagon, like those highlighted in Fig. 5-5, in shadow space. The star shaped object is the image of an octagon whose edges are traced out by starting at the origin and applying $e_1, e_2, \ldots, e_8$ in succession.

intersections of sets of equally spaced planes such that particles move in directions given either by the vertices of a regular icosahedron, or by its dual, the dodecahedron.[1] Particles can then flow in one of twelve or twenty symmetrically arranged channels. In our view, it would be preferable to have particles move in directions piercing the midpoints of the bonds of an icosahedron, thus obtaining a thirty channel model. These 30 channels can be grouped into ten coplanar sets of six, which point to the vertices of a regular hexagon. It is then possible to equilibrate momenta across channels using triple collisions in much the same way as for the triangular lattice. Such simple triple collisions are not possible with the other two arrangements.[1]

In any event, we think it may be better to obtain icosahedral quasilattices using the projection technique, rather than from the intersections of incommensurate planes, for the reasons discussed in the previous subsection. The projection technology is well-developed in this case, because it has been used to model recently discovered metallic alloys with an icosahedral symmetry.[9] One now tries to obtain a lattice of points of the form

$$\vec{r} = \sum_{i=1}^{6} n_a \vec{e}_i \qquad (5-19)$$

where six $\{\vec{e}_i\}$ now point to six of the twelve vertices of an icosahedron (the remaining six vertices are obtained by inversion). The allowed sextets of integers $\{n_1, \ldots, n_6\}$ are now obtained by projecting a six dimensional hypercubic lattice into three dimensions.[8,9,10] The projection matrix into physical space is now given by[8]

$$P = \frac{1}{\sqrt{20}} \left| \begin{array}{cccccc} \sqrt{5} & 1 & 1 & 1 & 1 & 1 \\ 1 & \sqrt{5} & 1 & -1 & -1 & 1 \\ 1 & 1 & \sqrt{5} & 1 & -1 & 1 \\ 1 & -1 & 1 & \sqrt{5} & 1 & -1 \\ 1 & -1 & -1 & 1 & \sqrt{5} & 1 \\ 1 & 1 & -1 & -1 & 1 & \sqrt{5} \end{array} \right| \qquad (5-20)$$

and points are projected only if their perpendicular projection (obtained from $Q = 1 - P$) falls within the three dimensional shadow of a 6D cube, which is the interior of a rhombic triacontahedron.

In physical space, one obtains a tiling of space by the two rhombahedra shown in Figure 5-9. These two shapes are analogous to the square and rhombus which appear in Figure 5-5. To obtain a cellular automaton, one

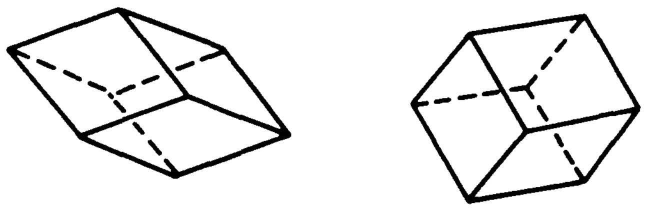


Figure 5-9. Prolate and oblate rhombahedral tiles that comprise the icosahedral quasilattice.

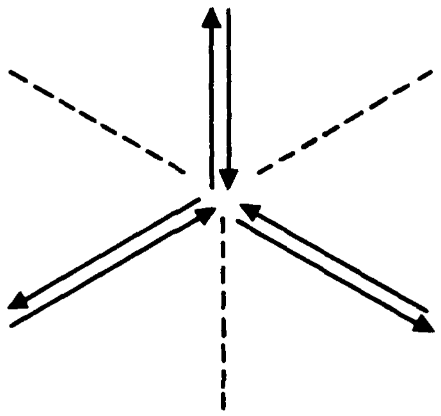


Figure 5-10. Schematic representation of a triple collision in a plane normal to a three fold symmetry axis of triacontahedron.

allows particles to hop between centers of rhombahedra, across a set of rhombic faces with identical orientation. The set of jagged paths obtained in this way leads to thirty possible momentum channels. Particles suffering head on binary collisions within a particular cell exit via one of two sets of parallel faces different from those by which they entered.

Just as one finds regular octagons in the 2-D quasilattice, we can find many regular rhombic triacontahedra in 3-D. These triacontahedra have thirty rhombic faces, corresponding to the thirty channels discussed above; the centers of the faces may be put into a one to one correspondence with the centers of the bonds of a regular icosahedron. Each triacontahedron is composed of ten large (prolate) and ten smaller (oblate) rhombahedra. A stereo view of a triacontahedron filled in this way is given in the paper by MacKay[11].

Momentum conserving triple collisions can be implemented using these triacontahedra as discussed above: Normal to each of the ten three-fold symmetry axes of the tricontrahedron there are six coplanar momentum channels pointing to the vertices of a hexagon. Whenever three particles enter a triacontahedron as sketched in Figure 5-10, we have them backscatter into the directions from whence they came.

Implementations are possible along the lines sketched for the octagonal quasilattice in Section 5.2.

APPENDIX TO SECTION 5
OCTAGONAL QUASILATTICE PROGRAM

The follwoing program was written and run on an IBM personal computer equipped with a high resolution graphics card and an 8087 chip to speed up arithmetic operations. Less than seven minutes were required to draw the 321 vertex lattice shown in Figure 5-4. The program was written using the TURBO Pascal compiler developed for the IBM PC by Borland International, Scotts Valley, California.

```
type octagon.pas
Appendix: Octagonal Quasilattice Program

     The following program was written and run on an IBM personal
computer equipped with a high resolution graphics card and an 8087
chip to speed up arithmetic operations.  Less than seven minutes were
required to draw the 321 vertex lattice shown in Fig. 4.  The program
was written using the TURBO pascal compiler developed by the IBM PC
by Borland International, Scotts Valley, California.

program octagon;

var
   e: array[1..2,1..4] of real; ep: array[1..2,1..4] of real;
   p: array[1..4,1..4] of real; q: array[1..4,1..4] of real;
   rs : array[1..4] of real;
   x0: array[1..4] of real;
   s,t,rx,ry,sx,sy,d1,d2,d3,d4: real;
   n: array[1..4] of real; m: array[1..4] of real;
   i,j,k,l,u,v,w,rxi,ryi,sxi,syi, count: integer;

const
   phi = 0.785398163; r8 = 2.828427125; r2 = 1.414213562; dm = 0.603553391;

begin
     HiRes; HiResColor(15);
     {initialize basis vectors in 2d physical space and 2d shadow space}
     for j := 1 to 4 do begin
          e[1,j] := cos((j - 0.5)*phi); e[2,j] := sin((j - 0.5)*phi);
          ep[1,j] := cos(3.0*(j - 0.5)*phi); ep[2,j] := sin(3.0*(j - 0.5)*phi);
     end;

     {initialize projection matrices}
     q[1,1] := 0.5; q[1,2] := -1.0/r8; q[1,3] := 0.0; q[1,4] := 1.0/r8;
     q[2,1] := -1.0/r8; q[2,2] := 0.5; q[2,3] := -1.0/r8; q[2,4] := 0.0;
     q[3,1] := 0.0; q[3,2] := -1.0/r8; q[3,3] := 0.5; q[3,4] := -1.0/r8;
     q[4,1] := 1.0/r8; q[4,2] := 0.0; q[4,3] := -1.0/r8 ; q[4,4] := 0.5;

     {compute displacement vector}
     s := 0.1; t := 0.05;
     x0[1] := s*0.5 - t*0.5; x0[2] := s*0.0 + t*r2/2.0;
     x0[3] := -s*0.5 - t*0.5; x0[4] := s*r2/2.0 + t*0.0;

     {test all lattice sites near origin of Z4 for inclusion in 2d
      octagonal latttice}
     count := 0;
     for i := -4 to 4 do begin
     for j := -4 to 4 do begin
     for k := -4 to 4 do begin
     for l := -4 to 4 do begin

     n[1] := i; n[2] := j; n[3] := k; n[4] := l;

         for u := 1 to 4 do begin
         rs[u] := 0.0;
         for v := 1 to 4 do begin
              rs[u] := rs[u] + q[u,v]*n[v];
         end;
         rs[u] := rs[u] + x0[u];
         end;
```

```
        d1 := rs[1]*q[1,1] + rs[2]*q[2,1] + rs[3]*q[3,1] + rs[4]*q[4,1];
        d2 := rs[1]*q[1,2] + rs[2]*q[2,2] + rs[3]*q[3,2] + rs[4]*q[4,2];
        d3 := rs[1]*q[1,3] + rs[2]*q[2,3] + rs[3]*q[3,3] + rs[4]*q[4,3];
        d4 := rs[1]*q[1,4] + rs[2]*q[2,4] + rs[3]*q[3,4] + rs[4]*q[4,4];
        if ((abs(d1)<dm) and (abs(d2)<dm) and (abs(d3)<dm)
           and (abs(d4)<dm)) then begin
        {if site ok then check its four neighbors and draw line joining them
         if these also check out}
          rx := e[1,1]*n[1] + e[1,2]*n[2] + e[1,3]*n[3] + e[1,4]*n[4];
          ry := e[2,1]*n[1] + e[2,2]*n[2] + e[2,3]*n[3] + e[2,4]*n[4];
                 sx := ep[1,1]*m[1] + ep[1,2]*m[2]
                       + ep[1,3]*m[3] + ep[1,4]*m[4];
                 sy := ep[2,1]*m[1] + ep[2,2]*m[2]
                       + ep[2,3]*m[3] + ep[2,4]*m[4];
                 sxi := 480 + round(45.0*sx); syi := 50 - round(18.0*sy);
                 plot(sxi,syi,1);
          count := count + 1;
          for w := 1 to 4 do begin
            m[1] := i; m[2] := j; m[3] := k; m[4] := l;
            m[w] := m[w] + 1;
            for u := 1 to 4 do begin
            rs[u] := 0.0;
            for v := 1 to 4 do begin
                rs[u] := rs[u] + q[u,v]*m[v];
            end;
            rs[u] := rs[u] + x0[u];
            end;
            d1 := rs[1]*q[1,1] + rs[2]*q[2,1] + rs[3]*q[3,1] + rs[4]*q[4,1];
            d2 := rs[1]*q[1,2] + rs[2]*q[2,2] + rs[3]*q[3,2] + rs[4]*q[4,2];
            d3 := rs[1]*q[1,3] + rs[2]*q[2,3] + rs[3]*q[3,3] + rs[4]*q[4,3];
            d4 := rs[1]*q[1,4] + rs[2]*q[2,4] + rs[3]*q[3,4] + rs[4]*q[4,4];
            if ((abs(d1)<dm) and (abs(d2)<dm) and (abs(d3)<dm)
             and (abs(d4)<dm)) then begin
                 sx := e[1,1]*m[1] + e[1,2]*m[2]
                       + e[1,3]*m[3] + e[1,4]*m[4];
                 sy := e[2,1]*m[1] + e[2,2]*m[2]
                       + e[2,3]*m[3] + e[2,4]*m[4];
                 rxi := 240 + round(25.0*rx); ryi := 110 - round(10*ry);
                 sxi := 240 + round(25.0*sx); syi := 110 - round(10*sy);
                 draw(rxi,ryi,sxi,syi,1);

                 end;
           end;
          end;
   end; end; end; end;
   writeln('total number of vertices is   ',count:6);
end.
E
```

6 COMPARISON BETWEEN CONVENTIONAL KINETIC THEORY AND CELLULAR AUTOMATA DERIVATIONS OF HYDRODYNAMICS

6.1 Introduction

There has been much work done on trying to solve partial differential equations by computing the evolution of properly chosen cellular automata (CA[1,4,5]). Most attention has been focused on the Navier-Stokes (N-S) equation to which we restrict ourselves here.

The essential idea of CA is the following. At any instant of time the state of the automata are described by giving the numbers of "particles" at each point of a fairly regular space-filling lattice. Each of the particles has one of a discrete set of velocities. At the succeeding time step the state is changed so that:

1. A particle with a given velocity moves to the nearest lattice point in the direction of the velocity.
2. Two or more particles can have "collided" and become particles moving with other of the discrete velocities.

The claim has been made that with appropriate choices of lattices and collision rules, the behavior of the CA when sufficiently averaged represents solutions of the N-S equations. Here we wish to investigate this. Discussions of the process have been given elsewhere.(1,4–6) However, we thought it to be useful to give a derivation completely in parallel with a conventional derivation of the N-S equations. In particular we try to be as general as possible - for example not specifying lattice or dimension whenever possible. Some (important) technical points are ignored. Thus the question of how to choose parameters so that the isotropy of the N-S equations is obtained is not considered. We drop the restriction frequently made that at most one particle with a given velocity can be at a site. This restriction seems mostly to simplify numerical computation. It is rather artificial and could be an additional barrier to obtaining a desired macroscopic equation.

In the following sections we give a step by step derivation of the N-S equation (denoted as "molecules" in the text headings to follow) taken directly from reference(12). At each step we then obtain the corresponding equation for CA. Discussion of similarities and differences are given when appropriate.

6.2 Equations for One Particle Distribution Function

6.2.1 Molecules

In principle we start with a Hamiltonian describing all particles of the system. Then a distribution function $f\,(\underset{\sim}{r}_1, \underset{\sim}{v}_1, ..., \underset{\sim}{r}_m, \underset{\sim}{v}_m, t)$ is introduced. Integrating the Liouville equation over all particle coordinates but one, we obtain an equation for the one particle distribution function $(f(\underset{\sim}{r}_1, \underset{\sim}{v}_1, t))$,

$$\frac{\partial f}{\partial t} + V \cdot \nabla f = J_v. \qquad (6-1)$$

J_v the collision term, contains all reference to two and many body collisions. We have for simplicity omitted any external potentials. These are readily included.

This equation is really no particular simplification because J_v involves the two particle distribution function. The equation for this involves the three particle distribution for which we have an equation involving the four particle function and so on to N. The usual assumption is, following Boltzmann, that of molecular chaos. This is to the effect that the two particle function can be approximated as the product of one particle functions. This means that particles before and after collision are uncorrelated - which is reasonable if the correlation function falls off rapidly in space and time (for example, exponentially). Recently(13) this has been found not to be necessarily the case. Power law behavior has been found. Consequently, the molecular chaos assumption which we will make can only be assumed safe for sufficiently low density. This does not mean that the N-S equations do not hold for dense fluids (e.g. water). Rather our derivation is then suspect.

With the assumption we have

$$J_v = \int d^3v_1 \int d\Omega \; g \; I(g,\Theta) \, [\, f' f_1' - f f_1 \,]. \qquad (6-2)$$

The integrand describes the collision of particles of velocity $\underset{\sim}{v}$ and $\underset{\sim}{v}_1$ producing or being produced by particles with velocity $\underset{\sim}{v}', \underset{\sim}{v}'_1$. We assume momentum conservation

$$\underset{\sim}{v} + \underset{\sim}{v}_1 = \underset{\sim}{v}' + \underset{\sim}{v}_1' \qquad (6-3)$$

and energy conservation

$$1/2 \, \underset{\sim}{v}^2 + 1/2 \, \underset{\sim}{v}_1^2 = 1/2(\underset{\sim}{v}'^2 + 1/2(\underset{\sim}{v}_1')^2. \qquad (6-4)$$

(Here all molecules are assumed to have unit mass.) I(g, θ) is essentially the differential scattering cross-section and

$$g = | \, \underset{\sim}{v} - \underset{\sim}{v}_1 \, | = | \, \underset{\sim}{v}' - \underset{\sim}{v}_1' \, |. \qquad (6-5)$$

The conservation laws tell us that

$$\int J_v \, d^3v = \int \underset{\sim}{v} \, J_v \, d^3v = \int \frac{v^2}{2} \, J_v \, d^3v = 0. \qquad (6-6)$$

6.2.2 C.A

Describing the state of the C.A. by a distribution function $f_a(\underset{\sim}{r}, t)$ giving the number of particles at $\underset{\sim}{r}, t$ with velocity $\underset{\sim}{e}_a$, and replacing small time and space differences by derivatives, gives the Boltzmann-like equation

$$\frac{\partial}{\partial t} f_a + \underset{\sim}{e}_a \cdot \nabla f_a = J_a. \qquad (6-7)$$

The $\underset{\sim}{e}_a$ are the allowed velocities which we here require all to be of unit magnitude. The J_a can involve two, three, four, ... products of the f's depending on the collision law chosen. To have an analogy with Equation (6-6) we must require that the particle number and momentum are conserved in the collison. Then we have

$$\underset{a}{\Sigma} \; J_a = O = \underset{a}{\Sigma} \, \underset{\sim}{e}_a \; J_a. \qquad (6-8)$$

We always have one less such equation than for the true Boltzmann equation. In the C.A. models "energy" is trivially conserved.

6.3 Macroscopic Conservation Laws

6.3.1 Molecules

Define for any $\psi(\underset{\sim}{v})$

$$\overline{\psi}(\underset{\sim}{v}) = \frac{\int \psi(\underset{\sim}{v})\, f(v)\, d^3v}{\int f(v)\, d^3v}$$

Multiplying Equation (6-1) by 1, $\underset{\sim}{v}$, and $\frac{v^2}{2}$ respectively and integrating over $\underset{\sim}{v}$ gives in view of Equation (6-6)

$$\frac{\partial n}{\partial t} + \frac{\partial}{\partial x_i}(nu_i) = O, \tag{6-9}$$

$$\frac{\partial}{\partial t}(nu_i) + \frac{\partial}{\partial x_j} n\, \overline{v_i v_j} = O, \tag{6-10}$$

and

$$\frac{partial}{\partial t}\frac{n}{2}\, \overline{v}^2 + \frac{\partial}{\partial x_i}\frac{n\overline{v_i v^2}}{2} = 0. \tag{6-11}$$

Here the number density $n = \int f d^3v$ and $\underset{\sim}{u}$ is the average velocity,

$$\text{i.e. } \underset{\sim}{u}\, (\underset{\sim}{x}, t) = \overline{\underset{\sim}{v}}. \tag{6-12}$$

The Equations (6-10) and (6-11) can be put in a more useful form by introducing the deviation of $\underset{\sim}{v}$ from its mean

$$\underset{\sim}{U} = \underset{\sim}{v} - \underset{\sim}{u}\, (\underset{\sim}{x}, t). \tag{6-13}$$

Then, for example,

$$\overline{v_i v_j} = u_i u_j + \overline{U_i U_j}.$$

Equation (6-2) becomes

$$\frac{\partial}{\partial t}(nu_i) + \frac{\partial}{\partial x_j} nu_i u_j = -\frac{\partial}{\partial x_j} P_{ij} \tag{6-14}$$

with

$$P_{ij} = n\, \overline{U_i U_j}. \tag{6-15}$$

Using mass conservation Equation (6-10) simplifies to

$$n[\frac{\partial u_i}{\partial t} + \underset{\sim}{u} \cdot \nabla u_i\,] = -\frac{\partial}{\partial x_j} P_{ij}. \qquad (6-16)$$

Similiarly introducing U_i Equation (6-3) and simplifying using Equations (6-3) and (6-14) we get

$$n\{\,\frac{\partial}{\partial t} + \underset{\sim}{u} \cdot \nabla\,\}\,\{\frac{Q}{n}\,\} + \frac{\partial}{\partial x_i} q_i = -P_{ij} D_{ij}, \qquad (6-17)$$

with

$$Q = \frac{n}{2}\overline{\underset{\sim}{U}}^2, q_i = \frac{n}{2}\overline{U_i\,\underset{\sim}{U}^2} \qquad (6-18)$$

and

$$D_{ij} = 1/2\,(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}). \qquad (6-19)$$

6.3.2 C.A.

Following the above procedure we multiply Equation (6-7) by 1 and $\underset{\sim}{e}_a$ and sum over a. We obtain

$$\frac{\partial n}{\partial t} + \frac{\partial}{\partial x_i} n u_i = 0 \qquad (6-20)$$

and

$$\frac{\partial}{\partial t}(n u_i) + \frac{\partial}{\partial x_j}\,\overline{n(\underset{\sim}{e}_a)_i(\underset{\sim}{e}_a)_j} = 0. \qquad (6-21)$$

Here

$$n = \sum_a f_a \qquad (6-22)$$

and

$$\underset{\sim}{u} = \overline{\underset{\sim}{e}_a}. \qquad (6-23)$$

Again introducing velocities relative to the mean

$$\underset{\sim}{U}_a = \underset{\sim}{e}_a - \underset{\sim}{u} \qquad (6-24)$$

and using Equation (6-18) to simplify Equation (6-19) we obtain

$$n\,\{\,\frac{\partial u_i}{\partial t} + \underset{\sim}{u} \cdot \nabla\,\underset{\sim}{u}_i\,\} = -\frac{\partial}{\partial x_j} P_{ij} \qquad (6-25)$$

with

$$P_{ij} = n\,T_{ij}\,, T_{ij} = \overline{(\underset{\sim}{U}_a)_i(\underset{\sim}{U}_a)_j}. \qquad (6-26)$$

6.4 The Euler Equations

6.4.1 Molecules

We are interested in solutions slightly away from equilibrium. Then as a zeroth approximation we should use an f° such that the collision integral is zero. This will be so if $\ln f$ is a linear function of the conserved quantities. This leads to

$$f^\circ = \frac{n}{(2\pi kT)^{3/2}}\, e\left[-\frac{(\underset{\sim}{v} - \underset{\sim}{u})^2}{2kT}\right], \tag{6-27}$$

the Maxwell distribution. (Note: At this point one usually invokes the H-theorem to further justify our choice of f°. However, to our knowledge there is no general H-theorem for CA's). Notice that if $\underset{\sim}{n}, \underset{\sim}{u}, T$ are all space-time independent Equation (6-25) is indeed an exact solution of the Boltzmann equation. However, even if $\underset{\sim}{n}, \underset{\sim}{u}, T$ are space time functions we still have $J_v[f^\circ] = 0$. A reasonable lowest order approximation for our macroscopic equations is to use f° allowing $T, \underset{\sim}{n}, \underset{\sim}{u}$, to vary and then calculate the quantities occurring in Equations (6-13) and (6-16). One obtains the set

$$\begin{aligned} q_i = 0,\ Q &= \frac{3}{2}p \\ P_{ij} &= p\delta_{ij},\ p = nkT \end{aligned}$$

$$\frac{\partial n}{\partial t} + \frac{\partial}{\partial x_i} nu_i = 0 \tag{6-28}$$

$$n\,\{\frac{\partial u_i}{\partial t} + \underset{\sim}{u}\cdot\nabla u_i\} = -\nabla p \tag{6-29}$$

$$(\frac{\partial}{\partial t} + \underset{\sim}{u}\cdot\nabla)(nT^{-3/2}) = 0 \tag{6-30}$$

which are the Euler equations with an adiabatic temperature law and the ideal gas equation of state. These are true for any distribution function f° which is even in the variable $\underset{\sim}{v} - \underset{\sim}{u}\,(\underset{\sim}{x}, t)$.

6.4.2 C.A.

Here again it is true that if *ln f* is a linear function of the conserved constants, then

$$J_a\,[f] = 0 \qquad (6-31)$$

even if the coefficients of the linear function depend on $\underset{\sim}{r}_1, t$.

We then assume as a lowest order approximation

$$f_a \approx f_a^{\circ} = \frac{n e^{-\underset{\sim}{\omega}\cdot\underset{\sim}{e}_a}}{\sum\limits_b e^{-\underset{\sim}{\omega}\cdot\underset{\sim}{e}_a}}. \qquad (6-32)$$

Here then indeed n is the number density. ω is a vector to be related to $\underset{\sim}{u}$. Specifically:

$$\underset{\sim}{u} = \frac{\sum\limits_a \underset{\sim}{e}_a\, e^{-\underset{\sim}{\omega}\cdot\underset{\sim}{e}_a}}{\sum\limits_a e^{-\underset{\sim}{\omega}\cdot\underset{\sim}{e}_a}} \qquad (6-33)$$

Using Equation (6-30) for the distribution function we obtain for the T_{ij} in Equation (6-26)

$$T_{ij} \approx T_{ij}^{\circ} = \frac{\sum\limits_a (\underset{\sim}{e}_a - \underset{\sim}{u})_i\,(\underset{\sim}{e}_a - u)_j}{\sum\limits_b e^{-\underset{\sim}{\omega}\cdot\underset{\sim}{e}_b}}\; e^{-\underset{\sim}{\omega}\cdot\underset{\sim}{e}_a}. \qquad (6-34)$$

These "Euler" equations then differ significantly from those for molecules:

1. The parameter $\underset{\sim}{\omega}$ in the distribution is <u>not</u> the average velocity.
2. The pressure tensor now has a rather strange dependence on $\underset{\sim}{u}$.

If however, $\underset{\sim}{u}$ is small (i.e.$|\omega| \ll 1$) we can expand the exponentials

$$u_i = \frac{-\sum\limits_a (\underset{\sim}{e}_a)_i\, \underset{\sim}{\omega}\cdot\underset{\sim}{e}_a}{\sum\limits_b 1} \qquad (6-35)$$

and

$$T_{ij}^{(\circ)} = \frac{\sum\limits_a (\underset{\sim}{e}_a)_i\,(\underset{\sim}{e}_a)_j}{\sum\limits_b 1} \qquad (6-36)$$

Assuming enough symmetry that these sums over α are isotropic (see Sections 2 and 5) we find

$$\underset{\sim}{u} = -\frac{\underset{\sim}{w}}{d} \tag{6-37}$$

and

$$P_{ij} = p\delta_{ij} \tag{6-38}$$

with

$$p = \frac{n}{d}. \tag{6-39}$$

(Here d is the number of spatial dimensions) i.e. we have the Euler equations at constant temperature kT = 1/d.

The lesson to be learned at this stage is that the C.A. are describing Euler hydrodynamics at constant temperature provided $| \underset{\sim}{u} | \ll 1$. This restricts C.A. models to small Mach numbers.

6.5 The Chapman-Enskog Expansion

6.5.1 Molecules

We have noted that the Maxwell-Boltzmann distribution function satisfies

$$J_v[f^\circ] = 0, \tag{6-40}$$

even if $\underset{\sim}{n}, \underset{\sim}{u}, T$ are functions of $\underset{\sim}{r}, t$. However, in this case the left hand side of Equation (6-1) will not be zero. Consider the case where $n, \underset{\sim}{u}, T$ are slowly varying on the spatial scale of the mean free path and time scale of the mean time between collisions. We proceed so.

Let

$$f = f^\circ[1 + \phi]. \tag{6-41}$$

Insert this in Equation (6-1). On the left hand side we keep only f° while on the right we keep only terms linear in ϕ. The result is

$$\frac{\partial f^\circ}{\partial t} + \underset{\sim}{v} \cdot \nabla f_o = L_v[\phi].$$

This is a <u>linear</u> integral equation for ϕ. Since we have the 5 conserved quantities, there are 5 zero eigenvalues of L_v corresponding to eigenfunctions

$$1, \underset{\sim}{v}, \frac{\underset{\sim}{v}^2}{2}. \qquad (6-42)$$

For Equation (6-37) to have a solution the inhomogeneous terms on the left must be orthogonal to these. But these conditions are just our "Euler" equations. Hence to evaluate the left hand side of Equation (6-37) where $n, T, \underset{\sim}{u}$ depend on $\underset{\sim}{r}$ and t we use the Euler equations to eliminate $\frac{\partial n}{\partial n}, \frac{\partial T}{\partial t}$ and $\frac{\partial u}{\partial t}$. This results in the integral equation

$$\begin{aligned} \{\frac{D_{ij}}{kT} & \left[U_i Uj - 1/3\, \delta_{ij} \underset{\sim}{U}^2 \right] + \frac{1}{T} \frac{\partial T}{\partial x_i} \qquad (6\text{-}43) \\ U_i & \left[\frac{\underset{\sim}{U}^2}{2kT} - \frac{5}{2} \right] \} f^\circ = L_v(\phi) \end{aligned}$$

The solution will be unique if we demand that

$$\int \begin{matrix} 1 \\ \underset{\sim}{v} \\ \frac{v^2}{2} \end{matrix} \; f^\circ(v)\, \phi\, d_i^3 = 0$$

For a general scattering cross-section we have no possibility of solving this analytically. However, we can see qualitatively what will result. Let ψ_i, λ_i be the eigenfunction of L_v. It can be readily shown that the ψ_i are orthogonal to each other with weight functions f_o, i.e. we can take the ψ_i to satisfy

$$\int \psi_i \psi_j f^\circ d^3 v = \delta_{ij}. \qquad (6-44)$$

The first 5 of the ψ_i correspond to the eigenvalue zero. The Equation (6-39) is satisfied if we expand so that

$$\phi = \Sigma_{i=6}^{\infty} a_i \psi_i. \qquad (6-45)$$

Further we note that the ψ_i will be functions only of $\underset{\sim}{U} = \underset{\sim}{v} - \underset{\sim}{u}$ and can be chosen to be even or odd in $\underset{\sim}{U}$. Then the coefficients of the even f› nctions are determined by the terms $\sim D_{ij}$ in Equation (6-38) and the coefficients of the odd functions are determined by the terms $\sim \frac{\partial T}{\partial x_j}$. The net result is that

$$\begin{aligned} P_{ij} &= nkT\, \delta_{ij} + c_1(T)\, (D_{ij} - \frac{1}{3} D_{kk} \delta_{ij}), \qquad (6\text{-}46) \\ q_i &= c_2(T) \frac{\partial T}{\partial x_i}. \end{aligned}$$

Note: Here c_1 and c_2 are independent of $\underset{\sim}{u}$. They are also independent of n. This last property results from the fact that we are considering only two body collisions. In Equation (6-38) there is one factor of n on the left and two on the right, therefore

$$\phi \sim \frac{1}{n} \qquad (6-47)$$

The portions of P_{ij} and q_i which depend on ϕ are then n independent.

6.5.2 C.A.

To complete the macroscopic Equations (6-18) and (6-23) we need to compute

$$T_{ij} = \overline{(\underset{\sim}{U}_a)_i\,(U_a)_j} \qquad (6-48)$$

$$\equiv \frac{\sum_a (U_a)_i (U_a)_j f_a}{\sum_a f_a}$$

For this we need f_a to be obtained from Equation (6-7). Because of the conservation laws

$$J_a[f^\circ] = O \qquad (6-49)$$

where

$$f_a^{(o)} = \frac{n e^{-\underset{\sim}{w}\cdot\underset{\sim}{c}_a}}{\sum_a e^{-\underset{\sim}{w}\cdot\underset{\sim}{c}_a}} \qquad (6-50)$$

even when n and $\underset{\sim}{w}$ (i.e. $\underset{\sim}{u}$) are functions of space and time.

We assume these are slowly varying functions and try

$$f_a = f_a^\circ[1+\phi_a]. \qquad (6-51)$$

Inserting this in Equation (6-7), keeping only the first term on the left and terms linear in ϕ on the right gives

$$\frac{\partial f_a^\circ}{\partial t} + \underset{\sim}{c}_a \cdot \nabla f_a^\circ = L_a[\phi] \qquad (6-52)$$

which is an inhomogeneous linear equation for ϕ. The inhomogeneous terms are given by the derivatives of f_a°. The linear operator L_a has zero eigenvalues corresponding to the number and momentum conservation laws. Thus for Equation (6-46) to have solutions we must require that

$$\sum_a \underset{\sim}{c}_a^1 \left(\frac{\partial f_a^\circ}{\partial t} = \underset{\sim}{c} \cdot \nabla f_a^\circ\right) = O. \qquad (6-53)$$

For the solution to be unique we require ϕ to be orthogonal to the eigenfunctions of zero eigenvalue.

The Equation (6-47) are just our Euler equations. In Section 6.1 we used these to eliminate $\frac{\partial n}{\partial t}$ and $\frac{\partial \underset{\sim}{u}}{\partial t}$ from the analog of Equation (6-48). Unfortunately it is not the derivatives of $\underset{\sim}{u}$which occur here but rather those of $\underset{\sim}{\omega}$. We thus obtain

$$\begin{aligned}(\frac{\partial}{\partial t} \;&+\; \underset{\sim}{e}_a \cdot\nabla)\, f_a^\circ = \{\, -\nabla\cdot u + \frac{(\underset{\sim}{e}_a - \underset{\sim}{u})_i}{n}\,\frac{\partial n}{\partial x_i} \\ &-\; (\underset{\sim}{e}_a - \underset{\sim}{u})_j\,(\underset{\sim}{e}_a - \underset{\sim}{u})_k\,\frac{\partial \omega_j}{\partial u_i}\,\frac{\partial u_i}{\partial x_k} \\ &+\; (\underset{\sim}{e}_a - \underset{\sim}{u})_j\,\frac{\partial \omega_j}{\partial u_i}\,\frac{1}{n}\,\frac{\partial}{\partial x_k}\,[\,nT^\circ_{ik}]\;\}f_a^\circ\end{aligned} \tag{6-54}$$

where $T_{ik}^{(O)}$ is as given in Equation (6-31).

However, this can be simplified. Thus, since

$$u_i = \frac{\sum_a (\underset{\sim}{e}_a)_i\, e^{-\underset{\sim}{\omega}\cdot\underset{\sim}{e}_a}}{\sum_a e^{-\underset{\sim}{\omega}\cdot\underset{\sim}{e}_a}} \tag{6-55}$$

$$\begin{aligned}\frac{\partial u_i}{\partial u_k} &= \delta_{ik} = -\frac{\partial \omega_j}{\partial u_k}\,\overline{(\underset{\sim}{e}_a - \underset{\sim}{u})_i - (\underset{\sim}{e}_a - \underset{\sim}{u})_j} \\ &= -T^\circ_{ij}\,\frac{\partial \omega_j}{\partial u_k} \text{ or } \frac{\partial \omega_j}{\partial u_i} = -(T^\circ)^{-1}_{ij}.\end{aligned} \tag{6-56}$$

Then Equation (6-48) becomes

$$\begin{aligned}(\frac{\partial}{\partial t} \;&+\; \underset{\sim}{e}_a \cdot\nabla)\, f_a^\circ = \{-\nabla\cdot \underset{\sim}{u} + (\underset{\sim}{e}_a - \underset{\sim}{u})_k (T^\circ)^{-1}_{ji}\,\frac{\partial u_i}{\partial x_k} \\ &-\; (\underset{\sim}{e}_a - \underset{\sim}{u})_j (T^\circ)^{-1}_{ji}\,\frac{\partial}{\partial x_k}\,T^{(0)}_{ik}\}\, f_a^\circ.\end{aligned} \tag{6-57}$$

Finally we note that after some calculation we can show that

$$\frac{\partial}{\partial x_k}\,T^{(0)}_{ik} = T^0_{ikj}\,(T^0)^{-1}_{jm}\,\frac{\partial u_m}{partialx_k} \tag{6 – 58}$$

where

$$T^\circ_{ikj} = \overline{(\underset{\sim}{e}_a - \underset{\sim}{u})_i\,(\underset{\sim}{e}_a - \underset{\sim}{u})_j\,(\underset{\sim}{e}_a - \underset{\sim}{u})_k}. \tag{6 – 59}$$

Thus our linear equation for ϕ is

$$\begin{aligned} \{ \quad - \quad & \nabla\cdot \underset{\sim}{u} + [(\underset{\sim}{e}_a - \underset{\sim}{u})_j(\underset{\sim}{e}_a - \underset{\sim}{u})_k - (\underset{\sim}{e}_a - \underset{\sim}{u})_m (T^\circ)^{-1}_{nm} \\ T^\circ_{mkj}] \quad & (T^\circ)^{-1}_{ji} \frac{\partial u_i}{\partial x_k}\} \, f^\circ_a \end{aligned} \tag{6-60}$$

The complicated dependence on $\underset{\sim}{u}$ of this equation should be compared with the simple form of Equation (6-38). Indeed the T° depend on $\underset{\sim}{u}$ and this is an implicit function of $\underset{\sim}{u}$.

To see what is happening we look at this equation for small $\underset{\sim}{u}$. The lowest order terms in the { } of Equation (6-53) are:

$$\{ \; \} = d\,\{\, (\underset{\sim}{e}_a)_i(\underset{\sim}{e}_a)_k - \frac{1}{d}\delta_{ik} \,\}\, d_{ik} \tag{6 - 61}$$

This should be compared to the terms in Equation (6-38) left when $\frac{\partial T}{\partial x_i} = 0$. We see this will lead to a pressure tensor of the form of Equation (6-42) (As explained above the c_1 will not now necessarily be independent of n.)

If we look at second order terms we obtain

$$\begin{aligned} \{ \; \}_2 \; = \; & \frac{2d}{d+2} \underset{\sim}{e}_a \cdot \underset{\sim}{u} \; \nabla\cdot \underset{\sim}{u} & \text{(6-62)} \\ - \; & \frac{d^2}{d+2}[\, (\underset{\sim}{e}_a)_k u_i \frac{\partial u_i}{\partial x_k} & \\ + \; & (\underset{\sim}{e}_a)_i \; \underset{\sim}{u}_k \frac{\partial u_i}{\partial x_k} \,] \,.. & \text{(6-63)} \end{aligned}$$

The first two terms of this expression will give rise to terms which have no analogue in the N-S equation. The last term gives rise to a modification of the convection term in the N-S equation.

6.6 Conclusion

The class of Cellular Automata considered here can model low velocity (i.e. incompressible) Navier-Stokes flows, for which the Mach number is << 1. Sound waves can be modeled, though they require a sound velocity that is specified arbitrarily. When calculations are made for Mach numbers approaching unity, the results are questionable. Some non-linear partial differential equations are being solved but they are not the Navier-Stokes equations.

In addition, we note that for both the Navier-Stokes and cellular automata systems, the Chapman-Enskog expansion assumes that typical scales for spatial variation are long compared with the collision mean free path. For fluid models this condition is easily satisfied since the typical cell size is usually kept large compared to a mean free path.

The situation is more difficult for cellular automata. Here the effective mean free path is generally a few times the distance between adjacent lattice points. Thus care must be taken that external sructures and boundaries introduced into the flow be already smoothed or rounded to the required spatial scales, since the physics of the lattice model alone will not guarantee slow enough variation of fluid quantities.

This requirement for smooth variation of boundaries and other structures in the flow has some interesting consequences. Although cellular automata models may be useful in the study of boundary layers for which the boundary geometry is complex, as suggested in Section 4, the Chapman-Enskog expansion requires that boundary surfaces of a cellular automata model must vary slowly relative to the lattice spacing. They cannot be too jagged or bumpy. This raises interesting questions about recent two-dimensional simulations of flow past a plate, for example, because at the abrupt ends of the plate there is a region where these assumptions are violated, as illustrated in Figure 6-1. In a fan-like region emanating from the corners of the plate, the Navier-Stokes equations are not being well modeled. It would therefore be interesting to repeat the calculation using a finite thickness plate with rounded ends, to see whether the previous sharp edges affected the downstream vortex structure or the long-time behavior of the flow.

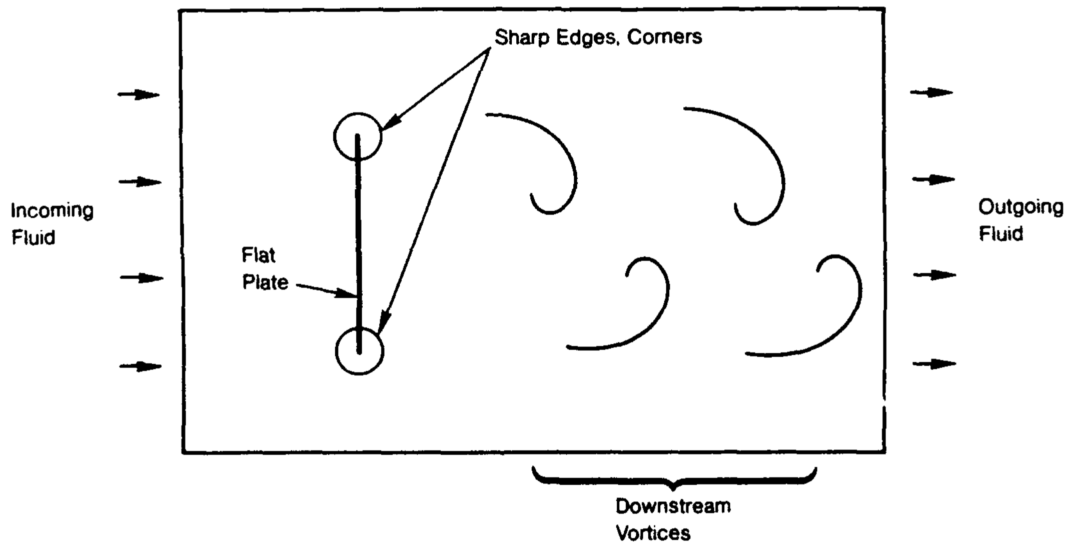


Figure 6-1. Sharp edges in cellular automata computation of flow past a plate raise questions about validity of Chapman-Enskog expansion near corners.

7 PACKING A PLANAR LATTICE

Cellular automata have been proposed for problems whose initial data have a natural parameterization by a metrical surface, usually the Euclidian plane E^2. Such applications include the study of: global weather, vision, two-dimensional fluid flow. The physical realization of such C.A.'s suggests an optimization problem in discrete geometry.

Let S_x be the square of side $= x$ in the usual planar lattice $Z + Z$ and C_y the cube of side y in the cubical lattice Z^3. Our problem is to represent the model S_x in the computer C_y so that: (1) model-adjacent nodes can exchange information quickly, and (2) a high density $\Delta = y^3/x^2$ is achieved.

On S_x we take the "city street" metric $1 1 \times 1 1 = 1 x_o 1 + 1 x_1 1$. We imagine C_y to be composed of horizontal layers or leaves within which communication time is essentially distance (e.g., as defined on S_x). Communication between distinct layers only occurs between vertically adjacent nodes <u>at the surface</u> of the cube. (We fix a constant c to be the "distance" between such pairs.) This determines a metric on C_y, dist (p,q) = $\overset{\min}{\gamma}$ (# (horizontal edges ofγ)+ c# (vertical edges of γ)) where γ is an edge path from p to q whose only vertical edges are on the cube's surface. Fabrication and heat dissipation problems make this model more realistic than a homogeneous cube.

It can be proved [14] that any one to one map $f : S_x \rightarrow C_y$ which fills the cube with a fixed density $p > 0$ must produce a distortion of distance (= communication time) which grows as $c^{1/2}$ const. $(\rho) \times^{1/16}$. For ρ near zero the constant is at least $\frac{\rho^{2/3}}{8\sqrt{6}}$; for ρ near 1 the constant is > 1/10.) Furthermore, we explicitly define a map F: $S_x \rightarrow C_y$ with density ≈ 1, 1/6-power stretching, and small leading coefficient (stretch$_F \approx \sqrt{2}\, c^{1/2} \times^{1/6}$).

The map F describes the best way to cut a two-dimensional data set apart and reassemble it on a stack of trays. Dicing into squares and stacking is not optimal since it concentrates all the stretching in the vertical direction (stretch $\approx \times^{1/3}$ arises). The idea in constructing F is to share the necessary stretching equally (hence in the amount $\times^{1/6}$) between horizontal and vertical stretching; these add - not multiply - to give the total stretching. For convenience we fix $c = 2$ and assume $\times^{1/6}$ is an integer; here is the definition of F:

$$F^{-1}(\text{each z-level set}) \text{ is a } \times^{5/6} \text{ by } \times^{1/2} \text{ rectangle } R_{\times^{5/6}} \times^{1/2}. \qquad (7-1)$$

Define

$$h(p,q) = (\rho(p_o)[p_1], x^{1/6}q + p_o)$$

where

$$p = P_o x^{2/3} + P_1, 0 \leqq P_1 \leqq x^{2/3} - 1$$

and

$$\rho(p_o) \varepsilon Z_2,$$

the group of two elements. The nontrivial element is the permutation:

$$\begin{array}{ccccc} 0 & 1 & & x^{2/\underline{3}}\,1 \\ \downarrow & \downarrow & \cdots\cdots & \downarrow \\ x^{2/\underline{3}}\,1 & x^{2/\underline{3}}\,2 & & 0 \end{array}$$

and σ(odd) is nontrivial, σ (even) is trivial. The map h is a bijection which stretches by no more than a factor of $\times^{1/6}$. Now set:

$$F(u,v) = (h(\sigma'(u_o)[u_1], \sigma''(v_o)[v_1]), u_o + x^{1/6} v_o),$$

where

$$u = u_o x^{5/6} + u_1, 0 \leqq u_1 \leqq x^{5/6} - 1$$

and

$$v = v_o x^{1/2} + v_1, 0 \leqq v_1 \leqq x^{1/2} - 1$$

σ' (odd) and σ" (odd) are permutations which (resp.) reverse the ordered sets:

As a final note, if Cy is given the more homogeneous metric $|| \times || = | \times_1 | + |x_2| + | \times_2 | + |x_3|$ then packings with $\rho \approx 1$ and stretching $= 3$ (independent of $\times$) can be constructed. I thank J. Komlos for this sharp bound on stretching.

DISTRIBUTION LIST

Dr. Henry D.I. Abarbanel
Institute for Nonlinear Science
Mail Code R002/Building CMRR/Room 115
University of California/San Diego
La Jolla, CA 92093-0402

Dr. Donald M. Austin
Scientific Computing Staff
Office of Energy Research
U.S. Department of Energy
ER-7, GTN
Washington, DC 20545

The Honorable John A. Betti
Undersecretary of Defense for Acquisition
The Pentagon, Room 3E933
Washington, DC 20301-3000

Dr. Arthur E. Bisson
Technical Director of Submarine
and SSBN Security Program
Department of the Navy, OP-02T
The Pentagon, Room 4D534
Washington, DC 20350-2000

Mr. Edward C. Brady
Sr. Vice President and General Manager
The MITRE Corporation
Mail Stop Z605
7525 Colshire Drive
McLean, VA 22102

Mr. Edward Brown
Assistant Director on Nuclear Monitoring
DARPA/PM
1400 Wilson Boulevard
Arlington, VA 22209-2308

Dr. Herbert L. Buchanan, III
Director
DARPA/DSO
1400 Wilson Boulevard
Arlington, VA 22209-2308

Dr. Curtis G. Callan, Jr.
Physics Department
P.O. Box 708
Princeton University
Princeton, NJ 08544

Dr. Kenneth M. Case
Institute for Nonlinear Science
Mail Code R-002
University of California/San Diego
San Diego, CA 92093-0402

Dr. Ferdinand N. Cirillo, Jr.
Central Intelligence Agency
Washington, DC 20505

Ambassador Henry F. Cooper
Director/SDIO-D
Strategic Defense Initiative Organization
Room 1E1081
The Pentagon
Wasington, DC 20301-7100

Mr. John Darrah
Senior Scientist and Technical Advisor
HQAF SPACOM/CN
Peterson AFB, CO 80914-5001

DISTRIBUTION LIST

Dr. Alvin M. Despain
Electrical Engineering Systems
SAL-318
University of Southern California
Los Angeles, CA 90089-0781

DTIC [2]
Defense Technical Information Center
Cameron Station
Alexandria, VA 22314

Professor Freeman J. Dyson
Institute for Advanced Study
Olden Lane
Princeton, NJ 08540

Maj Gen Robert D. Eaglet
Assistant Deputy SAF/AQ
The Pentagon, Room 4E969
Washington, DC 20330-1000

Mr. John N. Entzminger
Director
DARPA/TTO
1400 Wilson Boulevard
Arlington, VA 22209-2308

Dr. Robert Foord [2]
Central Intelligence Agency
Washington, DC 20505

Dr. Michael H. Freedman
Department of Mathematics
C-012
University of California/San Diego
La Jolla, CA 92093-0112

Dr. Richard Gajewski
Director, Division of Advanced Energy Projects
ER-16
U.S. Department of Energy
Washington, DC 20545

Dr. Larry Gershwin
Central Intelligence Agency
Washington, DC 20505

Dr. Fred M. Glaser
Office of Technical Coordination
U.S. Department of Energy
FE-14/GTN
Washington, DC 20545

Dr. S. William Gouse
Sr. Vice President and General Manager
The MITRE Corporation
Mail Stop Z605
7525 Colshire Drive
McLean, VA 22102

Dr. David A. Hammer
Laboratory of Plasma Studies
369 Upson Hall
Cornell University
Ithaca, NY 14853

DISTRIBUTION LIST

LTGEN Robert D. Hammond
Commander and Program Executive Officer
U.S. Army / CSSD-ZA
Strategic Defense Command
P.O. Box 15280
Arlington, VA 22215-0150

Mr. Thomas H. Handel
Office of Naval Intelligence
The Pentagon
Room 5D662
Washington, DC 20350-2000

Mr. Joe Harrison
Central Intelligence Agency
P.O. Box 1925
Room GV 1710 NHB
Washington, DC 20505

Dr. Robert G. Henderson
Director
JASON Program Office
The MITRE Corporation
7525 Colshire Drive, Z561
McLean, VA 22102

JASON Library [5]
The MITRE Corporation
Mail Stop: W002
7525 Colshire Drive
McLean, VA 22102

Dr. O'Dean P. Judd
Los Alamos National Lab
Mail Stop A-110
Los Alamos, NM 87545

Mr. Alfred Lieberman
ACDA/OA
Room 5726 State
320 21st Street N.W.
Washington, DC 20451

Mr. Robert Madden [2]
Department of Defense
National Security Agency
ATTN: R-9 (Mr. Madden)
Ft. George G. Meade, MD 20755-6000

Mr. Charles R. Mandelbaum
U.S. Department of Energy
Code ER-32
Mail Stop: G-236
Washington, DC 20545

Mr. Arthur F. Manfredi, Jr. [10]
OSWR
Central Intelligence Agency
Washington, DC 20505

Dr. Oscar P. Manley
Office of Basic Energy Research
U.S. Department of Energy
Code ER-15/GTN
Washington, DC 20545

Mr. Joe Martin
Director
Naval Warfare and Mobility
Office of Tactical Warfare Programs
The Pentagon
Washington, DC 20301

DISTRIBUTION LIST

Dr. Claire E. Max
Inst. of Geophysics & Planetary Physics
Lawrence Livermore Natl Lab
L-413
P.O. Box 808
Livermore, CA 94550

MGEN Thomas S. Moorman, Jr.
Director of Space and SDI Programs
Code SAF/AQS
The Pentagon
Washington, DC 20330-1000

Dr. Julian C. Nall
Institute for Defense Analyses
1801 North Beauregard Street
Alexandria, VA 22311

Dr. David R. Nelson
Department of Physics
Harvard University
Cambridge, MA 02138

Dr. Robert L. Norwood [2]
Director for Space
and Strategic Systems
Office of the Assistant Secretary of the Army
The Pentagon, Room 3E474
Washington, DC 20310-0103

Mr. Gordon Oehler
Central Intelligence Agency
Washington, DC 20505

Dr. Peter G. Pappas
Chief Scientist
U.S. Army Strategic Defense Command
P.O. Box 15280
Arlington, VA 22215-0280

MAJ Donald R. Ponikvar
Deputy for Division Engineering Systems
ODDR&E/DS
The Pentagon
Room 3D136
Washington, DC 20301-3090

Mr. John Rausch [2]
NAVOPINTCEN Detachment, Suitland
4301 Suitland Road
Washington, DC 20390

Records Resources
The MITRE Corporation
Mailstop: W115
7525 Colshire Drive
McLean, VA 22102

Dr. Victor H. Reis
Acting Director
DARPA
1400 Wilson Boulevard
Arlington, VA 22209-2308

Dr. Oscar S. Rothaus
Math Department
Cornell University
Ithaca, NY 14853

DISTRIBUTION LIST

Dr. Fred E. Saalfeld
Director
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5000

Dr. Philip A. Selwyn [2]
Director
Office of Naval Technology
Room 907
800 North Quincy Street
Arlington, VA 22217-5000

Dr. Donald K. Stevens
Associate Director for Basic Energy
Sciences,ER-10
U.S. Department of Energy
Office of Energy Research, GTN/Room J304
Washington, DC 20545

Superintendent
Code 1424
Attn: Documents Librarian
Naval Postgraduate School
Monterey, CA 93943

Dr. George W. Ullrich [3]
Deputy Director
Defense Nuclear Agency
6801 Telegraph Road
Alexandria, VA 22310

Ms. Michelle Van Cleave
Assistant Director for National Security
Affairs
Office of Science and Technology Policy
New Executive Office Building
17th and Pennsylvania Avenue
Washington, DC 20506

Mr. Richard Vitali
Director of Corporate Laboratory
U.S. Army Laboratory Command
2800 Powder Mill Road
Adelphi, MD 20783-1145

Dr. Edward C. Whitman
Duputy Assistance Secretary of the Navy
C3I Electronic Warfare & Space
Department of the Navy
The Pentagon, 4D745
Washington, DC 20350-5000

RADM Ray Witter
Director – Undersea Warfare
Space and Naval Warfare Systems Command
Code: PD-80
Department of the Navy
Washington, DC 20363-5100

ADM Daniel J. Wolkensdorfer
Director
DASWD (OASN/RD&A)
The Pentagon
Room 5C676
Washington, DC 20350-1000

Dr. Linda Zall
Central Intelligence Agency
Washington, DC 20505

Mr. Charles A. Zraket
President and Chief Executive Officer
The MITRE Corporation
Mail Stop A265
Burlington Road
Bedford, MA 01730