Simulation of fluid-dynamic processes with a probabilistic lattice gas

Lin Zhang

ABSTRACT

In recent SEP reports, Muir has developed a sequence of new lattice gas models with M2 as representative. In this paper, based on a simple proof of the existence and uniqueness of a solution for M2, a Steepest Ascent method is developed to solve nonlinear problems and obtain numerical solutions. Several fluid-dynamic processes are simulated for sound wave modeling and an inconclusive result for a vortex problem.

INTRODUCTION

Lattice gases represent a new and interesting way of modeling fluid-dynamic processes. Several models, such as HPP (Hardy, 1976) and FHP (Frisch, 1986), have been proposed and tested. These models describe the individual particle motion, and in order to show macroscopic behavior, a very large number of particles must be considered, which in turn requires powerful computers such as the Connection Machine, even for 2-D problems.

Muir reported a new model, M2, which treats the lattice gas as an ensemble, and calculations are carried out on probability values rather than on individual particles. This gives a tremendous saving of space and time in the computer and make it possible to do simulations on an ordinary machine. Muir and Sword have previously worked on a linear variant, M3, which models infinitesimal perturbations from a relaxed state. However, a complete solution of M2 is subject to a proof of uniqueness and existence of solution and the development of a numerical algorithm for this nonlinear problem.

M2 MODEL

A lattice gas describes how particles move along grids and defines collision rules. Contrary to some other lattices gases which make a point of reversibility, M2 requires that at each collision particles will move away all possible directions with equal probability. The objective function is the entropy function of state. The rule says that the next state will maximize the entropy under mass and momentum conservation constraints. Clearly this rule can be applied to any lattice structures including those in higher dimensions. For simplicity, I will focus on the two dimensional hexagonal grid as an example.

Let u_i be the probability that a particle moves towards a node in the ith direction. The evolution of the lattice gas involves two processes: a collision and a shift. Figure 1 shows how particles travel on the hexagonal lattice.

Conservation constraints

Let us define a probability vector

$$\mathbf{u} = (u_1 \quad u_2 \quad \dots \quad u_n)^T . \tag{1}$$

$\mathbf{u}$ describes the state of a vector. Three physical quantities: mass m, horizontal momentum p_x and vertical momentum p_y can be represented as

$$\mathbf{Cu} = \begin{pmatrix} m \\ p_x \\ p_y \end{pmatrix} \tag{2}$$

where $\mathbf{C}$ is a constant matrix. It is required that these three quantities are conserved during the collision process, so we have

$$\mathbf{Cu} = \mathbf{Cu}^0 \tag{3}$$

where $\mathbf{u}^0$ and $\mathbf{u}$ are probability vectors before and after collision.

Collision rule

Let $\mathbf{u}^0$ be a probability vector before collision

$$\mathbf{u}^0 = (u_1^0 \quad u_2^0 \quad \dots \quad u_n^0)^T \tag{4}$$

with $0 \leq u_i^0 \leq 1$. M2 states that after collision the new probability vector will be

$$\mathbf{u} = (u_1 \quad u_2 \quad \dots \quad u_n)^T \tag{5}$$

with $0 \leq u_i \leq 1$, such that the entropy of the state

$$H(\mathbf{u}) = \sum_i h(u_i) \tag{6}$$

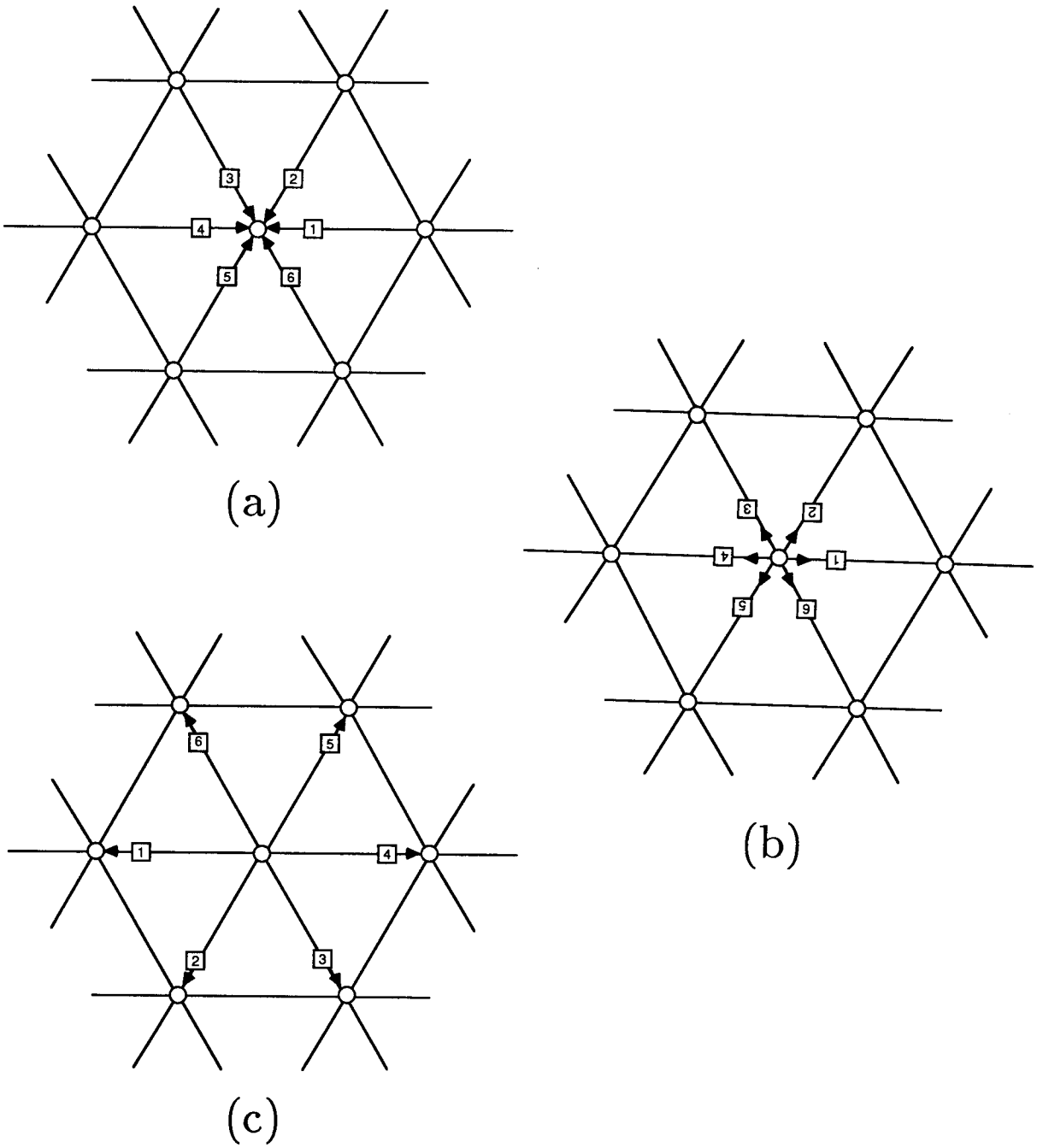


FIG. 1. Hexagonal lattice notation: (a) previous state, (b) collision process, (c) shift process.

is maximized under the conservation constraints

$$\mathbf{Cu} = \mathbf{Cu}^0 \tag{7}$$

where $h(x)$ is the entropy function

$$h(x) = xlog(\frac{1}{x}) + (1-x)log(\frac{1}{1-x}). \tag{8}$$

Meaning of objective function

Since u_i's are independent, the state probability (joint probability) is their product. So the entropy of the states is

$$H(\mathbf{u}) = \mathbf{E}[log_2(\frac{1}{\mathbf{P(u)}})] = \mathbf{E}[\sum_i log_2(\frac{1}{\mathbf{p}(u_i)})] = \sum_i h(u_i). \tag{9}$$

UNIQUENESS AND EXISTENCE OF SOLUTION

Before trying to solve this nonlinear problem, it is necessary to examine the validity of the solution. It is known that any strict convex downward function has unique maximum value. We will show that our objective function is strictly convex downward in the constrained domain.

Convexity of objective function

Define a domain

$$D_1 = \{\mathbf{u} \mid 0 \leq u_i \leq 1 \; i = 1, \ldots, n\}. \tag{10}$$

We know $h(u_i)$ is a strict convex downward function of u_i, but not strict in D_1. For any u_i^1 , u_i^2 and any scalar a, b with $0 \leq a, b \leq 1$ and $a + b = 1$

$$ah(u_i^1) + bh(u_i^2) \leq h(au_i^1 + bu_i^2), \tag{11}$$

the equality holds only when $u_i^1 = u_i^2$, therefore

$$a\sum_i h(u_i^1) + b\sum_i h(u_i^2) < \sum_i h(au_i^1 + bu_i^2) \tag{12}$$

for any $\mathbf{u}^1 \neq \mathbf{u}^2$. So $H(\mathbf{u})$ is a strict convex downward function in D_1. Now let us look at the constraints

$$\mathbf{Cu} = \mathbf{Cu}^0. \tag{13}$$

Let

$$\mathbf{u} = \mathbf{u}^0 + \partial\mathbf{u} \tag{14}$$

Here $\partial\mathbf{u}$ represents the change of probability vector. Now we have

$$\mathbf{C}\partial\mathbf{u} = \mathbf{0} \tag{15}$$

that is, $\partial\mathbf{u}$ is in the null space of $\mathbf{C}$. Consider $\mathbf{E}$, the generating matrix of this null space,

$$\mathbf{CE} = \mathbf{0}, \tag{16}$$

then we can write

$$\mathbf{u} = \mathbf{u}^0 + \mathbf{Ex} \tag{17}$$

here $\mathbf{x}$ is a vector.

Define another domain

$$D_2 = \{\mathbf{u} \mid 0 \leq u_i \leq 1; \mathbf{u} = \mathbf{u}^0 + \mathbf{Ex}\} \tag{18}.$$

Obviously D_2 is a subdomain of D_1. For any two points $\mathbf{u}^1$, $\mathbf{u}^2$ in D_2, $\hat{\mathbf{u}}$ is any point on the line segment connecting these two points,

$$\hat{\mathbf{u}} = a\mathbf{u}^1 + b\mathbf{u}^2 \tag{19}$$

where a, b are scalars with $0 \leq a, b \leq 1$ and $a + b = 1$. It is easy to show $\hat{\mathbf{u}}$ is in D_2. Then inequality (12) holds for any $\mathbf{u}^1 \neq \mathbf{u}^2$ in D_2. Thus $\mathrm{H}(\mathbf{u})$ is a strict convex downward function in convex domain D_2. Therefore we have proved that there always exists a unique solution for M2.

Two equivalent conditions

We can obtain this solution in two ways

- Find $\max\{\mathrm{H}(\mathbf{u})\}$ point in D_2.
- Find zero $\mathbf{grad}_{\mathbf{x}}\{H(\mathbf{u})\}$ point in D_2.

NUMERICAL ALGORITHM

There are many algorithms for solving nonlinear problems. I prefer considering stability first. In fact, five numerical algorithms are programmed and compared. They are

- Picard's method
- Muir's method
- M3 method (linear method)
- Steepest Ascent method

- Improved method

Here I will only describe Steepest Ascent method in detail and let the reader refer to the references for other methods. Again I will use hexagonal lattice as an example.

Steepest ascent method

The steepest ascent method is a standard method to solve maximization problems. It is quite easy and safe to apply to our problem because of the nice convexity of the objective function and domain. Since the solution path follows a gradient direction, it is expected to converge quickly.

We try to maximize $H(\mathbf{u})$ with $\mathbf{u} = \mathbf{u}^0 + \mathbf{E}\mathbf{x}$, where

$$\mathbf{x} = (x_1 \quad x_2 \quad \dots \quad x_{n-3})^T. \tag{20}$$

Let

$$\mathbf{g_u} = \mathbf{grad_u}\{H(\mathbf{u})\} = (log(\tfrac{1-u_1}{u_1}) \quad \dots \quad log(\tfrac{1-u_n}{u_n})), \tag{21}$$

$$\mathbf{g_x} = \mathbf{grad_x}\{H(\mathbf{u})\}, \tag{22}$$

then

$$\mathbf{g_x} = \mathbf{g_u}\mathbf{E}. \tag{23}$$

Starting from an initial state, we iterate

$$\mathbf{x}^{l+1} = \mathbf{x}^l + K\mathbf{g}_{\mathbf{x}}^{l\,T}, \tag{24}$$

and equivalently

$$\mathbf{u}^{l+1} = \mathbf{u}^l + K\mathbf{E}\underline{\mathbf{E}}^T\mathbf{g}_{\mathbf{u}}^{l\,T}. \tag{25}$$

K is an important factor which controls the speed and stability of the process. We can choose K to maximize $H(\mathbf{u}^{l+1})$ locally along gradient direction.

Expand $H(\mathbf{u}^{l+1}) = H(\mathbf{u}^l + K\mathbf{g}_{\mathbf{x}}^{l\,T})$ as a polynomial of K in the neighborhood of $K = 0$, and ignore the third and higher order terms.

$$H(\mathbf{u}^l + \mathbf{g}_{\mathbf{x}}^{l\,T}) \approx H(\mathbf{u}^l) + \frac{\partial H(\mathbf{u}^l)}{\partial K}K + \frac{1}{2}\frac{\partial^2 H(\mathbf{u}^l)}{\partial K^2}K^2. \tag{26}$$

We can also derive

$$\frac{\partial H(\mathbf{u}^l)}{\partial K} = \mathbf{g}_{\mathbf{u}}^l\mathbf{E}\mathbf{E}^T\mathbf{g}_{\mathbf{u}}^{l\,T}, \tag{27}$$

$$\frac{\partial^2 H(\mathbf{u}^l)}{\partial K^2} = \mathbf{g}_{\mathbf{u}}^l\mathbf{E}\mathbf{E}^T\mathbf{B}^l\mathbf{E}\mathbf{E}^T\mathbf{g}_{\mathbf{u}}^{l\,T} \tag{28}$$

where $\mathbf{B}^l$ is a matrix with elements

$$\mathbf{b}_{ij}^l = \frac{\partial^2 H(\mathbf{u}^l)}{\partial u_i \partial u_j}. \tag{29}$$

B turns out to be a diagonal matrix,

$$\mathbf{b}_{ij}^{l} = -\frac{1}{u_i^l(1-u_j^l)}\delta_{ij}. \tag{30}$$

Set

$$\frac{\partial H(\mathbf{u}^{l+1})}{\partial K} = 0, \tag{31}$$

then we get

$$K = -\frac{\mathbf{g}_{\mathbf{u}}^{l}\mathbf{E}\mathbf{E}^T{\mathbf{g}_{\mathbf{u}}^{l}}^T}{\mathbf{g}_{\mathbf{u}}^{l}\mathbf{E}\mathbf{E}^T\mathbf{B}^l\mathbf{E}\mathbf{E}^T{\mathbf{g}_{\mathbf{u}}^{l}}^T}. \tag{32}$$

An example of C and E matrices

In the case of the hexagonal lattice gas,

$$\mathbf{C} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 & -1 & 1 \\ 0 & 1 & 1 & 0 & -1 & -1 \end{pmatrix}. \tag{33}$$

$$\mathbf{E} = \begin{pmatrix} 1 & 0 & 1 \\ -1 & 1 & -1 \\ 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 1 \\ 0 & -1 & -1 \end{pmatrix} \tag{34}$$

Generally, **E** is not unique for a given **C**. But this will not affect the solution and also it is not difficult to find a simple one.

Some protections

Notice two places in the algorithm that may cause problems: discontinuity of the objective function and poor choice of K. Both these situations could cause the algorithm to give a solution which violates condition $0 \leq u_i \leq 1$. The first problem can be solved by setting a bound for u_i, anything beyond the boundary values will be changed to the nearest boundary value. To solve the second problem, we need to constrain K. Whenever the current value of K causes a violation, reduce K to half its value and check again. Numerical experiments show that this situation rarely happens, probably because of the convex property.

Linear solution is given by

$$\mathbf{u} = \mathbf{A}\mathbf{u}^0, \tag{35}$$

where **A** is collision matrix. Let

$$\mathbf{A} = \mathbf{I} + \mathbf{D}, \tag{36}$$

then

$$\mathbf{D} = \mathbf{A} - \mathbf{I}, \tag{37}$$

$$\mathbf{u} = \mathbf{u}^0 + \mathbf{D}\mathbf{u}^0, \tag{38}$$

$\mathbf{I}$ is identity matrix. Clearly

$$\mathbf{C}\mathbf{D}\mathbf{u}^0 = \mathbf{0}, \tag{39}$$

then

$$\mathbf{u} = \mathbf{u}^0 + K\mathbf{D}\mathbf{u}^0 \tag{40}$$

satisfies contraints, it is not a optimum linear solution but is meaningful for proper constant K.

Numerical results

A interactive program written in C reads in a desired input or generates random input, and give a solution with bounded error. Figure 2 shows a set of typical convergence curves. Table 1 gives the iteration number of 6 samples for different methods.

Number of iterations for error < 0.0001					
Sample	Picard	Francis	Linear	Steep	Improve
1	35	37	*	11	6
2	41	41	*	4	8
3	29	29	*	7	3
4	78	90	*	10	15
5	41	41	*	10	5
6	43	43	*	9	7
Number of iterations for error < 0.01					
1	-	-	1	4	1
2	-	-	1	2	1
3	-	-	1	4	1
4	-	-	*	7	7
5	-	-	1	4	1
6	-	-	*	4	2

Table 1. Number of iterations

Clearly, Picard's and Muir's method have almost the same speed, while Steepest Ascent is much faster. The linear solution usually gives a good approximation for the first step but there it stops. This suggests using the linear solution as a first step and then improving it by Steepest Ascent. I call this the Improved Method. This often leads to a less iteration number, but not always. For safety, I choose Steepest Ascent to perform the simulation which forms the rest of this report.

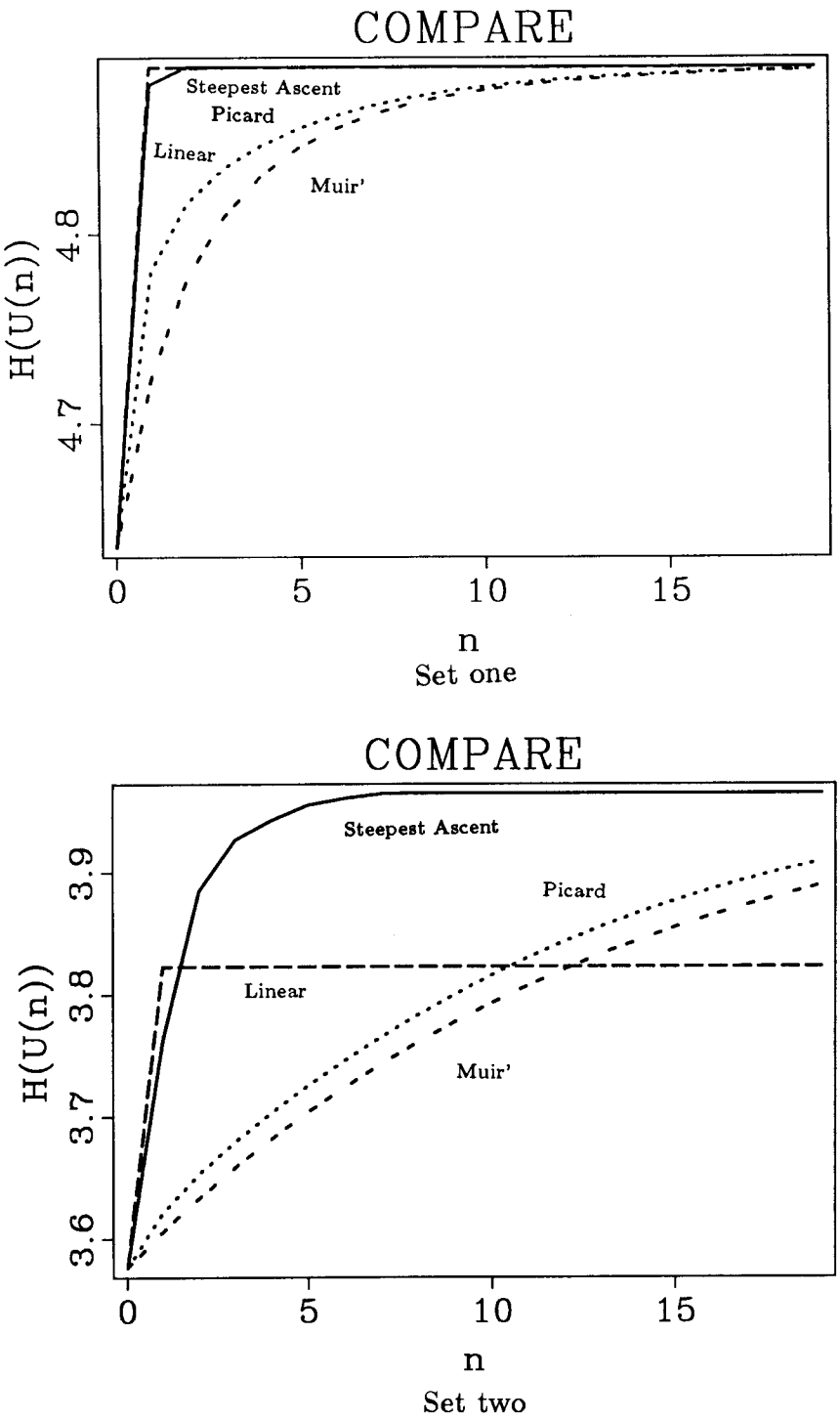


FIG. 2. Two sets of convergence curves. Horizontal axis shows the number of iterations, and vertical axis shows entropy

SIMULATION OF FLUID-DYNAMIC PROCESSES

An interesting observation is that Muir's method is constructed directly on the irreversibility ideal of M2, and has a lot of physical meaning. Now it transpires that it has the same solution as the other methods. This intuitively shows that our object function does work well. However, is the irreversibility what we want, in other words, does it really describe the physical phenomena? To answer this question, we need to do more experimenting.

Problems involved in simulation

A simple application of M2 is to simulate fluid-dynamics in a medium with fixed boundaries. As mentioned before, simulation includes two major processes: the collision and the shifting process. Both have special cases at boundaries. Several quantities should be considered.

1. dimension of data and computation time

   These two quantities are related. Large dimension implies a long computation time. The probabilistic approach of M2 greatly reduces the dimension, however in order to emerge certain details in a phenomena, a sufficient number of vertices should be considered. To me 256×256 is probably a upper bound (for computational reasons). To update all these vertices once requires about 2 minutes cpu time on our Convex.

2. boundary condition

   An important point is that unlike other models M2 describes particle motion with probability values. Thus the computation involved is completely deterministic. For example, to describe a equilibrium state conventional lattice gas models will have a random distributed moving particles with constant average value. But in M2 only average values are considered, so we have a deterministic value as a function of position and directions. Therefore at equilibrium state, the states of every vertex is unchanged, and boundary vertices should not create disturbance. Now let us look at several boundary conditions. First one is simply bouncing back along the same edge, as shown in Figure 3. In this case, rigid boundary is assumed but reflection angle is ignored. It is very easy to program so in the simulation it is always used. A better one will be reflecting specularly by following Snell's law, as shown in Figure 4. In both cases mass is conserved but momenta are not. Now suppose we want to simulate an infinite region, then periodic boundary conditions, as shown in Figure 5, are a good idea. Finally for a constant flow, add in a constant quantity of momentum at one side and take out same quantity at the other.

3. velocity of sound wave

   In lattice gas models, all particles move with the same velocity along edges. Let this velocity be unity, the wave traveling velocity is derived to be $\frac{1}{\sqrt{2}}$.

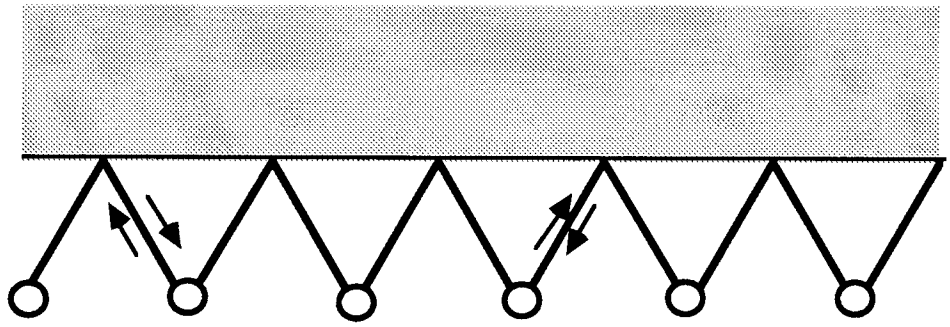


FIG. 3. Simple bouncing back boundary.

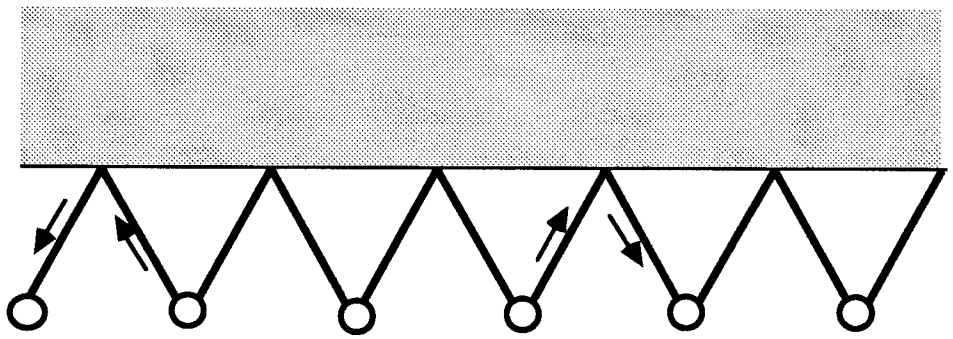


FIG. 4. Reflecting specularly boundary.

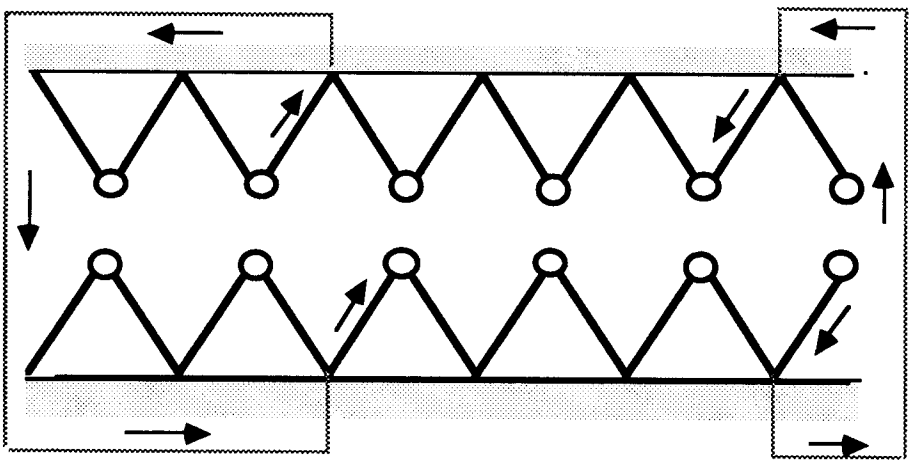


FIG. 5. Cyclic boundary.

Three phenomena are simulated with hexagonal lattice structure which is two dimensional.

Circular sound wave produced by a localized disturbance

A tightly-packed cloud of particles is placed in an equilibrium environment of less densely packed particles. A compressed circular wave expands outward. The boundary condition is set so that the equilibrium states are kept till the disturbance arrival. In this experiment, I use probability 0.9 for the disturbance and 0.1 for environment. 128×128 lattice gases form an array. The allowable error in a collision process is 0.0001. Figure 6 shows a propagating wave. The wavelet has a positive head and negative tail. The wave travel velocity is measured to be approximately 0.71, which is roughly equal to the theoretic value. The wavefront is a circle because of the isotropic medium and the proper choice of rules. Figure 7 shows the distribution of mass m, horizontal momentum p_x and vertical momentum p_y. Figure 8 shows particle motions along six different directions. Similar experiments using other lattice gas models are reported. In these other experiments the number of gas particles involved is very much larger but with less macro resolution.

Plane sound wave reflected from rigid boundary

A plane wave is generated by a tightly-packed belt of particles. The wave travels through less dense media and reflect from a rigid boundary. Here again I use probability 0.9 for the disturbance and 0.1 for the background. 64×128 vertices are considered. Boundary condition is simply totally reflecting. Allowable error in the collision process is 0.0001. Figure 9 shows the wave propagation and reflection. Velocity is measured to be 0.74, very close to the circular wave case. An interesting point is that unlike the circular wave just shown the wavelet of this plane wave does not have a negative tail. This is because the rigid boundary reflects all particles. Wavefront remains planar except at the two boundaries.

Hydrodynamics — flow past an obstacle

This test attempts to simulate a non-linear phenomenon, vortex-shedding in the wake of an obstacle to fluid flow. In order to show this, proper choice of obstacle size, velocity of fluid flow and continuity of flow are important. Different parameters may result in different phenomena. Figure 10 shows the distribution of particles moving in six different directions. Figure 11 shows the distributions of total mass m, horizontal momentum p_x and vertical momentum p_y of the same experiment. It turns out that vortex motion is not that obvious. Why is that? The dimension of the obstacle and density of the lattice gases are possible causes. But I think that symmetry and nonrandomness are important. As we mentioned before, the computation of M2 is completely deterministic, so if we set a uniform initial distribution and symmetric boundary the result will be a symmetric distribution which in turn can not give rise to a turbulent phenomenon.

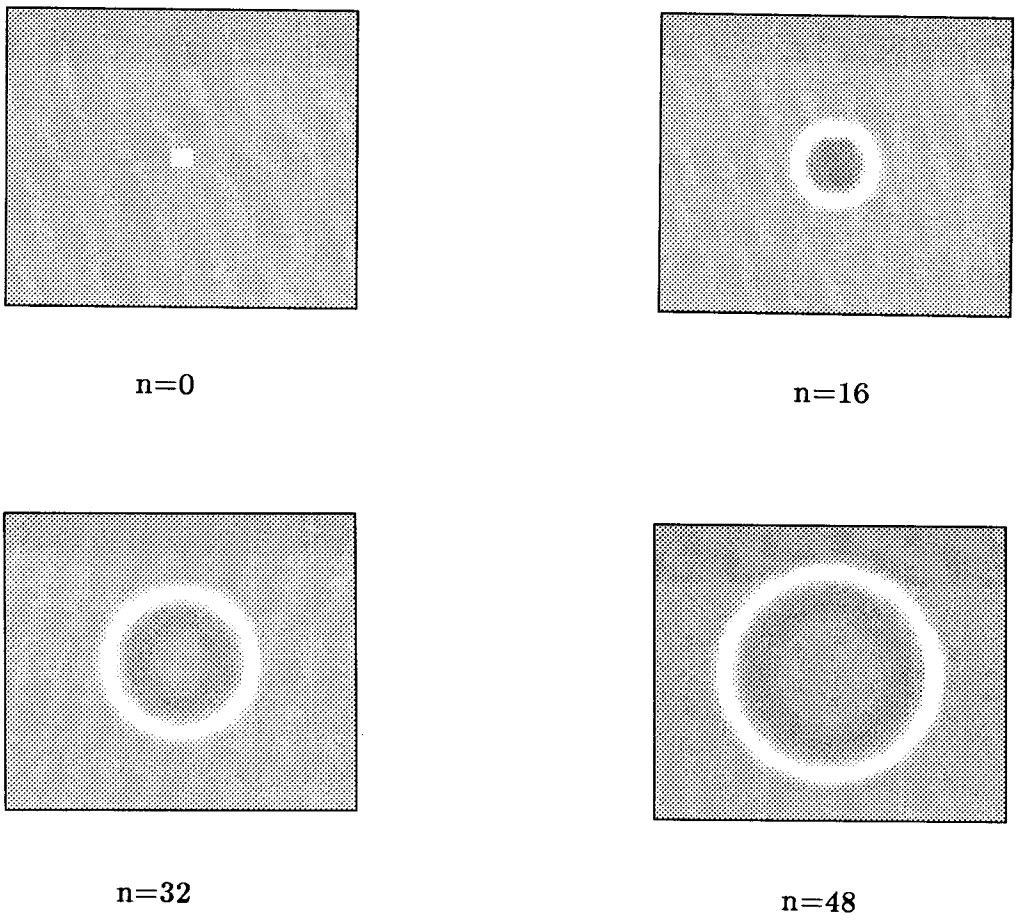


FIG. 6. Circular wave propagation.

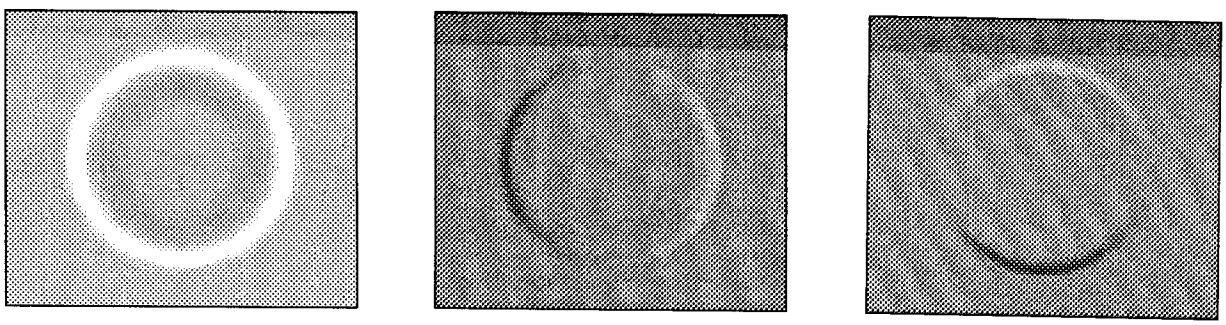
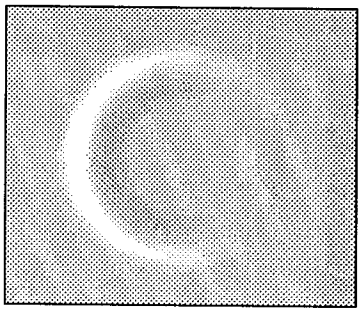
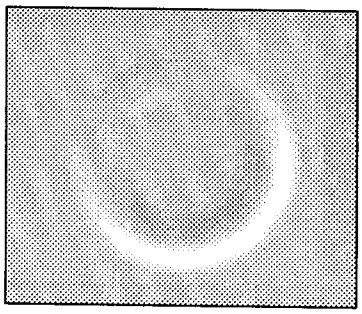


FIG. 7. Distributions of mass and momentums of circular wave.

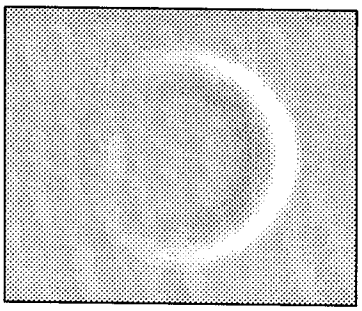


direction 1

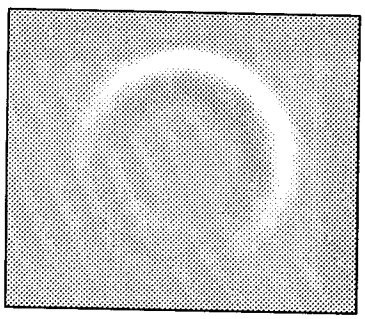
direction 2



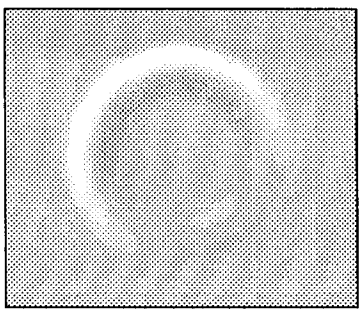
direction 3



direction 4



direction 5



direction 6

FIG. 8. Distributions of particle motions in six directions.

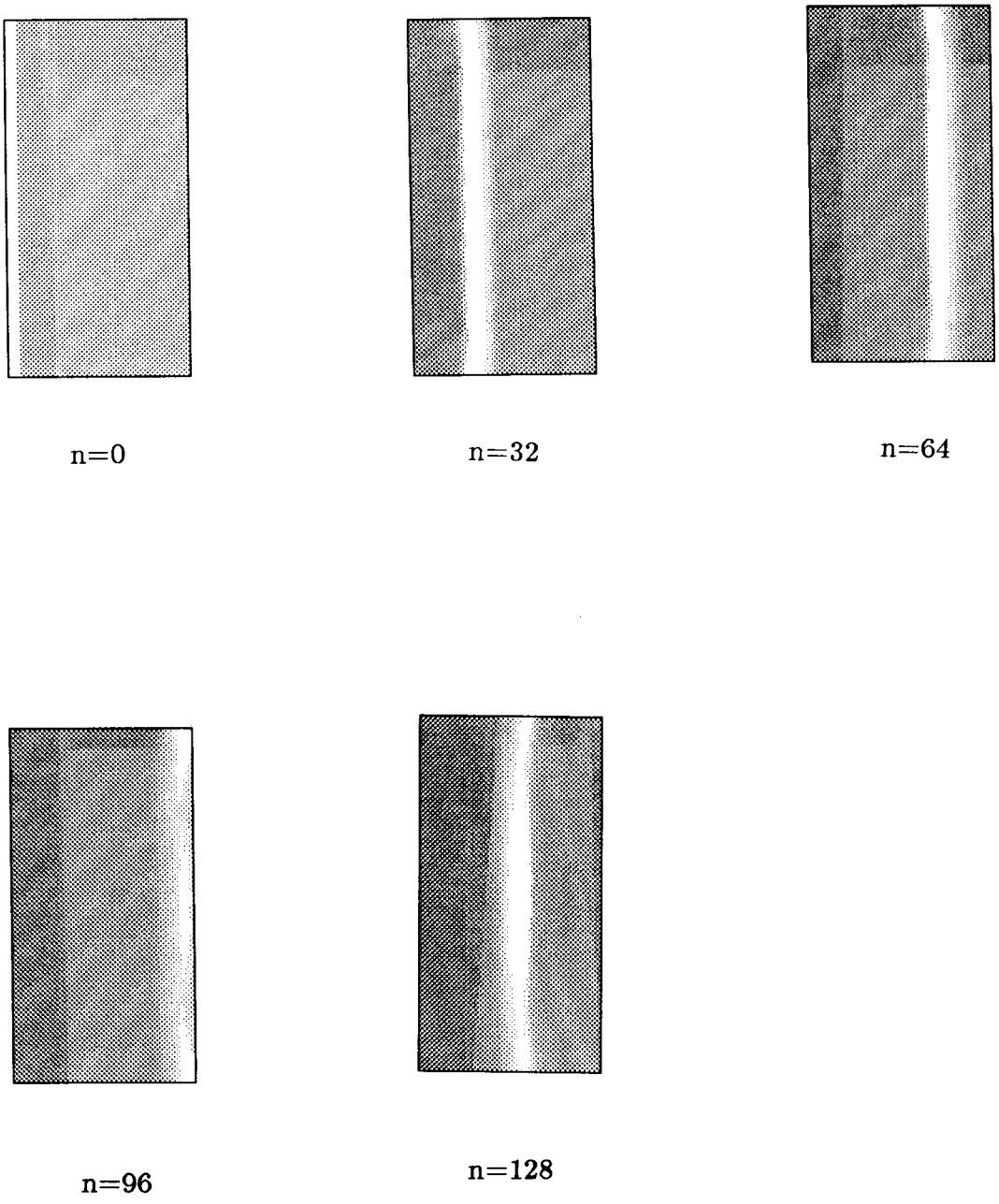
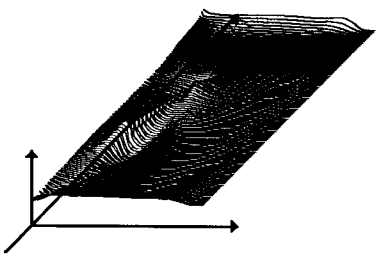
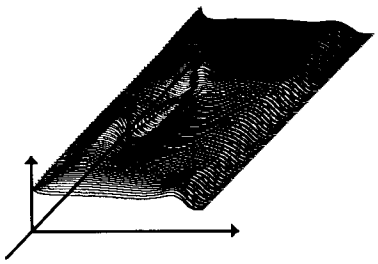


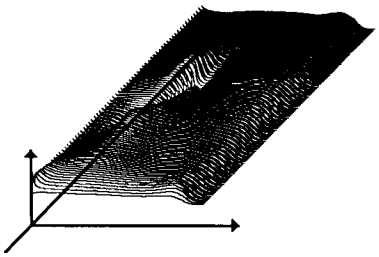
FIG. 9. Propagation and reflection of plane wave.



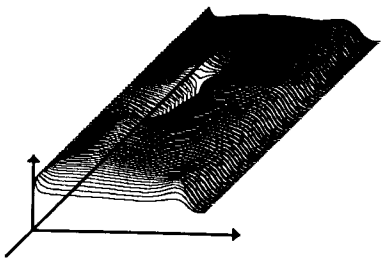
direction 1



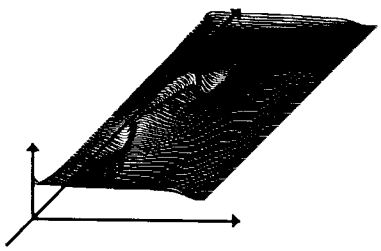
direction 2



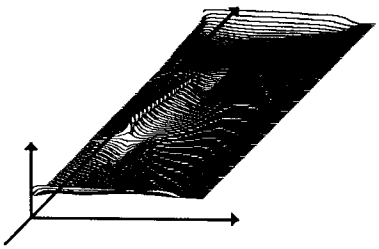
direction 3



direction 4

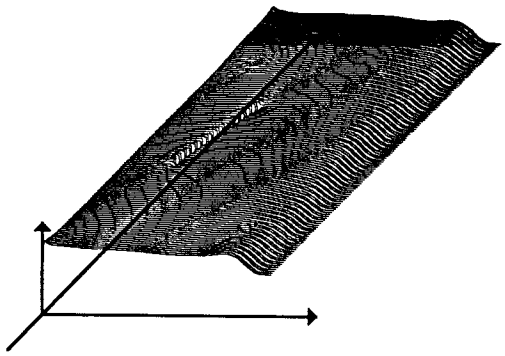


direction 5

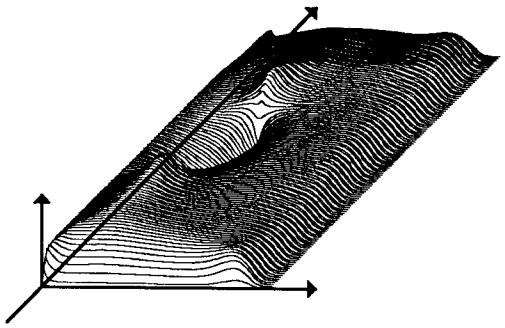


direction 6

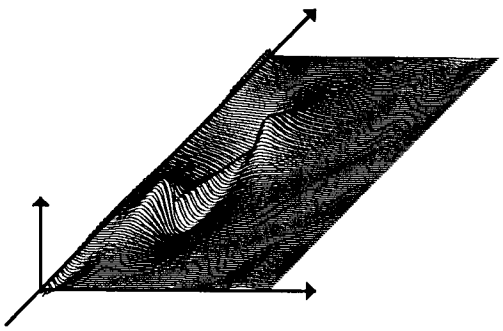
FIG. 10. Particle flow in six different directions.



mass



horizontal momentum



vertical momentum

FIG. 11. Distributions of mass and momentums in the flow.

DISCUSSION

The results of these numerical experiments tell us that M2 can be used in the simulations of fluid-dynamic processes and give similar results obtained by other methods. Moreover, M2 has more potential.

What happens when in solid

An important application of wave theory appears in reflection seismology. In this case, media are solid and we can have several kind of wave traveling with different speeds. For example, S wave, particles moves perpendicular to the wave propagating direction. If we look at the momentum perpendicular to the traveling direction in the simulated results, we noticed that they are zero. So M2 represents a P wave which is the only case in fluid. Now how to modify M2 so that it will work for solid. Answer is unknown. But it seems that we need to add some constraints on the motion of particles at neighboring vertices because S wave is caused by shear tension.

How about other processes

The same scheme can be used for any processes which involve particle collision and in which collision constraints are linear, for example the diffusion process. It can also simulate processes between two extreme cases by alternatively use of two different kinds of constraints.

Extension to 3-D

M2 rules are available for processes of higher dimensions. But computation becomes heavy in three dimensions. For a six-particle 2-D hexagonal lattice, to update a 128×128 array takes about 1 minute cpu time. Think about three dimension of $128 \times 128 \times 128$ array with more particles, the computation for updating once needs at least 128 minutes. Usually lattices need to be update to cover its full length, so we need 128×128 minutes, which is up to 10 days. One possible way to reduce computation time is to increase the allowable error of the collision process. For example, we can try iterating only once at each time step.

CONCLUSION

In this paper, the existence and uniqueness of solution for M2 are proved and a successful algorithm is constructed. The numerical experiments show that M2 provides an efficient way to do simulation in fluid-dynamic process. They also show the potential of M2 in other processes. The same algorithm can be applied to higher dimensions. But computation time gradually becomes a problem. To further explore the power of M2, more experiment should be done.

ACKNOWLEDGEMENTS

I would like to thank Francis Muir for helping me to understand the concepts of lattice gases, giving me suggestions and finally making all kinds of corrections on this paper. Without his constant guidance this paper would not have been possible.

REFERENCES

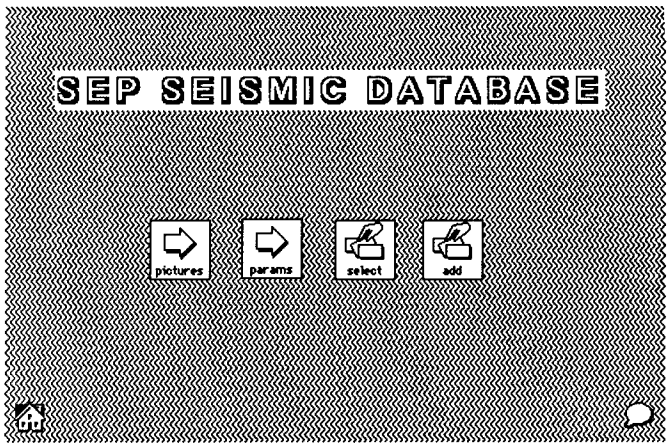
Stoer J., Gulirsch R., 1979, Introduction to Numerical Analysis: Springer-Verlag.

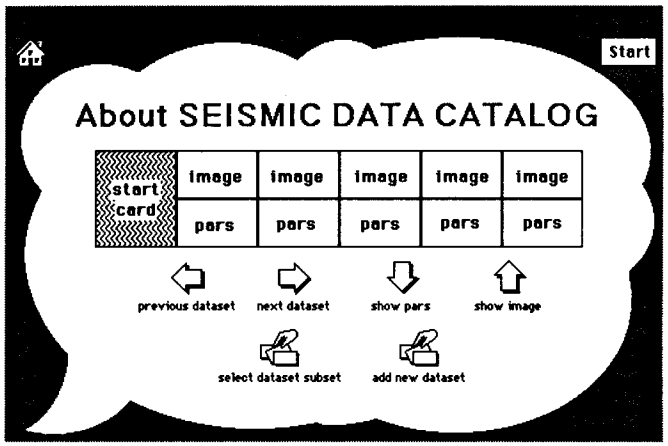
d'Humieres, D., Lallemand, P. and Frisch, U., 1986, Lattice gas models for 3D hydrodynamics: Europhys. Lett. **2** (4), 291-297.

Muir, F., 1987, Three experimental modeling systems: SEP–**51**, 119-128.

Wolfram, S., 1986a, Theory and Applications of cellular automata: World Scientific.

Sword C., 1987, Numerical dispersion and attenuation in the M3 modeling system: SEP–**56**, 1-22.





Data parameters are in a "name=value" format:

name=	name (usually file name)
title=	title for plotting
in=	data file name
author=	data source, processor
place=	geographic location
source=	shot type (airgun, dynamite ...)
geometry=	trace order (profile, gather, split spread)
n1= n2= n3=	fast, medium, slow axis lengths
d1= d2= d3=	fast, medium, slow sample rates
o1= o2= o3=	fast, medium, slow origins
label1= label2= label3	fast, medium, slow axis names
esize=	element size in bytes
transp=	data transposed? (fast axis is time)
clip=	plot clip value
gpow=	plot contrast
tpow=	plot time gain

OK