International Journal of Modern Physics C Vol. 3, No. 2 (1992) 347–365

PERIODIC AND QUASIPERIODIC TIME-DEPENDENT BEHAVIOR IN COLLECTIVE ORDERED PHASES

JASON A. C. GALLAS*†, ROCH BOURBONNAIS*‡ and HANS J. HERRMANN*

*Höchstleistungsrechenzentrum (HLRZ), KFA,
5170 Jülich, Germany

†Laboratório de Óptica Quântica da UFSC,
88049 Florianópolis, Brazil

‡Thinking Machines Corporation,
245 First Street, Cambridge, MA 02142-1214, USA

Received 24 December 1991

We investigate numerically totalistic cellular automata rules in four and five dimensions that have been recently proposed by Chaté and Manneville and that show periodic or quasiperiodic time sequences in the magnetization. We show how to implement such rules fast on a Connection Machine. We confirm the proposed behavior by analyzing the time and size dependence of the distance from the attractor. Also the dependence of the initial configuration, the behavior of very small systems and mean-field calculations are presented.

Keywords: Cellular Automata; Computer Algorithms; Large-Scale Simulations.

1. Introduction

Can collective systems like cellular automata display ordered phases in which the global magnetization varies periodically, quasiperiodically or even chaotically in time after reaching an attractor in the thermodynamic limit? Until last year the answer was believed to be "no" because of the rather convincing argumentation given by Grinstein and coworkers.[1,2] The totalistic cellular automata rules recently presented by Chaté and Manneville,[3] however, seem to indicate otherwise: for dimensions $d \geq 4$ some numerical evidence was given for the existence of periodic and quasiperiodic behavior.

The appearance of these novel systems poses, of course, many questions. On one hand one would like to be sure that the numerical work has not been fooled by long transients or finite-size behavior. On the other hand the urgency now exists to understand these new phases more profoundly. How stable are these phases under changes in the initial configuration or under thermal noise? What happens on a microscopic scale when the magnetization (density of ones) oscillates in steady state

(i.e., after waiting so long that the attractor is reached)? Is a mean-field description possible? The aim of this paper is to contribute to answer these questions, by simulating cellular automata involving up to 33,554,432 individual sites.

In the next section we describe the models used here. In Sec. 3 we present and describe two computer programs written in Fortran 90 and in PARIS for a parallel computer, namely for a CM–2 Connection Machine. Section 4 is devoted to the investigation of small size lattices, i.e., lattices with no more than $32^4 = 1{,}048{,}576$ sites. In Sec. 5 we discuss the transients and the dependence on the initial configuration. Section 6 is devoted to the analysis of finite time and finite size effects. In Sec. 7 we briefly discuss the effects of external noise and in the last section we present some mean-field calculation and discuss the nature of the new phases. The present paper reports complementary results to those recently given in Ref. 4.

2. The Chaté–Manneville Automata

Let us consider a hypercubic lattice of dimension d, placing on each site i a binary variable σ_i. Let us also define a local field

$$h_i(t) = \sum_{j=nn} \sigma_j(t) \tag{1}$$

where the sum goes over the central site i and its nearest neighbors (von Neumann neighborhood), i.e., over $2d+1$ sites. Then the family of totalistic automata we are interested in is defined by the rule

$$\sigma_i(t+1) = \begin{cases} 1, & \text{if } k \le h_i(t) \le l \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

These automata, like the game of life have a "survival window": If one considers "1" as living and "0" as dead, sites can only survive if the density of surrounding living sites is neither too high nor too low. k and l are the upper and lower limit of the window respectively (see Eq. 2).

Chaté and Manneville[3] presented several examples of k and l in four and five dimensions for which the global magnetization presented either periodic behavior or limit cycles. In particular they discussed the rule $R^d_{k,l} = R^5_{5,9}$ in Ref. 3 and rules $R^4_{4,8}$ and $R^4_{3,8}$ in Ref. 5. The first two rules show quasi-periodic behavior while $R^4_{3,8}$ has a periodic orbit. We want to reinvestigate in this paper these three rules to see whether they are stable or not with respect to several possible 'perturbations' of the rules, and, equally important, if they are not just a kind of long transient behavior. We considered only the case of periodic boundary conditions.

The automata is defined on a hypercube having L sites along each of its d dimensions, thereby having $N = L^d$ sites. We investigate two-state automata, i.e., each site contains a two-state logical variable, either $\sigma_i = 0$ or 1. The physical quantity that we want to consider is the global "magnetization" $m(t) = (1/N)\sum_i \sigma_i$,

namely the fraction (or "concentration") of numbers "1" present at a given time in the hypercubic lattice. Starting from an equiprobable random initial configuration

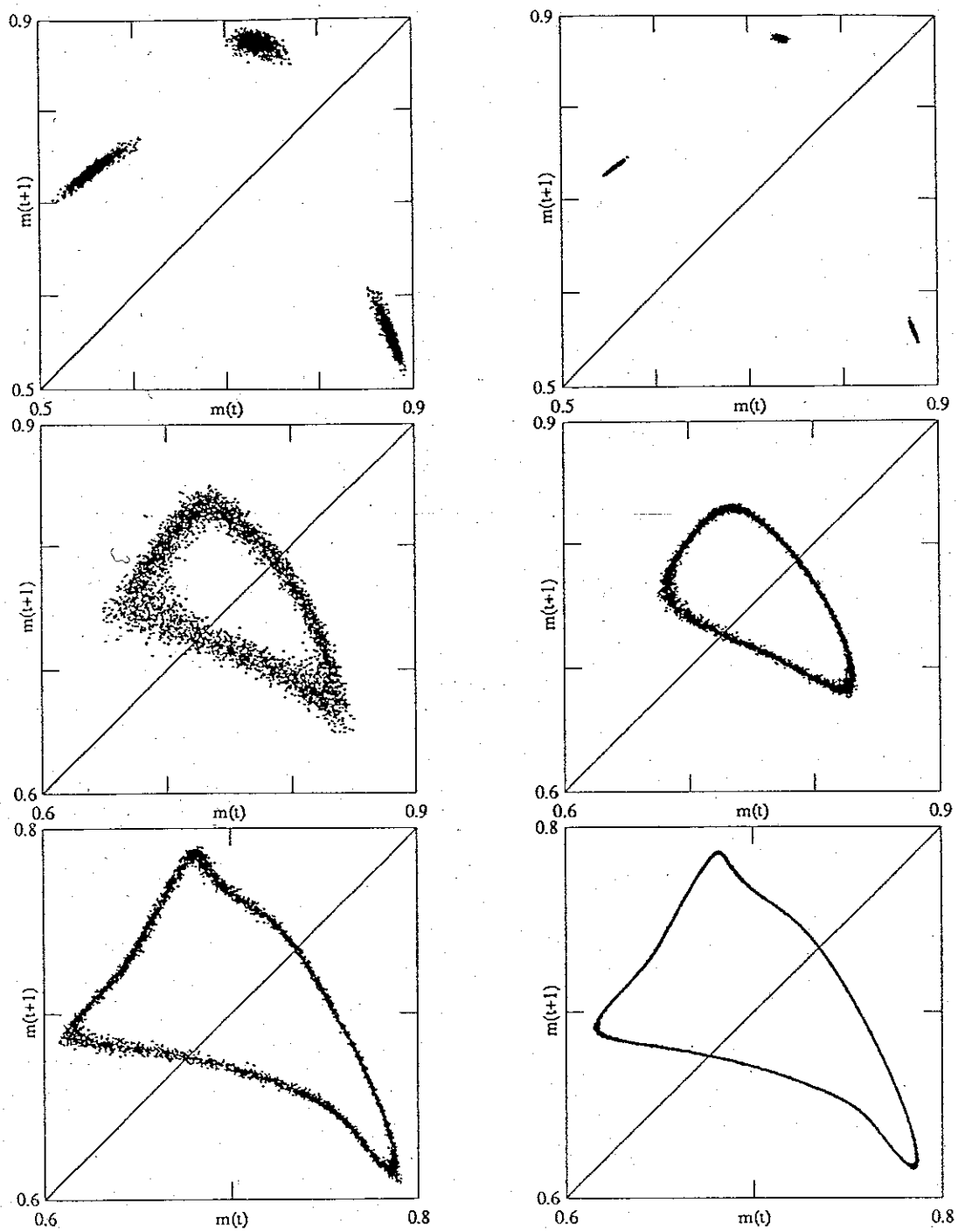

Fig. 1. Evolution space plots of $m(t) \times m(t+1)$ for $L = 16$ (left column) and $L = 32$ (right column), for $100 < t < 5{,}000$. The top horizontal doublet shows the evolution space for $R^4_{3,8}$, the middle doublet for $R^4_{4,8}$ and that on the bottom for $R^5_{5,9}$.

of "0" and "1", like Chaté and Manneville, we monitored the temporal evolution of $m(t)$ as a function of L and d. Most of the time, after a rather short transient the system is attracted to a finite region on an "evolution" plot of $m(t+1) \times m(t)$. Examples of the observed behaviors are shown in Fig. 1, where $m(t+1)$ is plotted against $m(t)$ for $t \geq 100$. The three leftmost figures show for $L = 16$, from top to bottom, the evolutionspace for rules $R^4_{3,8}$, $R^4_{4,8}$ and $R^5_{5,9}$, respectively, while the rightmost ones show the same evolution space for $L = 32$. The trajectories seem to approach a limit cycle and there are good indications for a quasiperiodic behavior.[5] By looking at the movements of the plotter pen while the space for rules $R^4_{4,8}$ and $R^5_{5,9}$ is plotted one realizes that they are obtained from a quasiperiodic rotation of the three "spots" present for $R^4_{3,8}$. So, in those cases the periodic motion turned into a quasiperiodic one, with the magnetization wandering smoothly on a torus. What we want to investigate in this paper is if these data are stable under changes in the initial configuration and initial concentration of "1" and "0", in the system size, in the observation times and under the influence of external noise.

3. Programming Details on the Connection Machine

All automata discussed in this paper were generated on a 16K processor Connection Machine (CM–2) at the HLRZ(GMD) using two FORTRAN 90 programs: a version intended to study lattices of small sizes (the "small size" limit) and another program, used to investigate bigger systems, taking advantage of the 32 bit wordlength of the CM–2 and using multispin coding techniques. Program 1 was used to investigate small size systems.

```
      integer, parameter :: seed=5855, tmax=5100
      integer, parameter :: L1=16, L2=16, L3=16, L4=16
      real, parameter :: size=L1*L2*L3*L4
      integer, array(L1,L2,L3,L4) :: a
      real, array(tmax) :: m
      integer tempo
cmf$ layout m(:serial)
      open(unit=7, file='aus', status='unknown')
      a = 0 ! initializes the full lattice 'a' with zeros
      call cmf_randomize(seed) ! fills randomly 'a' with 1 and 0
      call cmf_random(a,2)
      tempo = 1
      m(tempo) = sum(a)/size ! counts initial magnetization
      do 1000 tempo = 2,tmax ! starts time loop
          a = a + cshift(a,1,-1) + cshift(a,1,1)
     *          + cshift(a,2,-1) + cshift(a,2,1)
     *          + cshift(a,3,-1) + cshift(a,3,1)
     *          + cshift(a,4,-1) + cshift(a,4,1)
```

```
        where( a.lt.4 .or. a.gt.8) ! applies automaton rule
            a = 0
        elsewhere
            a = 1
        end where
        m(tempo) = sum(a)/size ! counts & saves m(t)
1000 continue
     do 2000 tempo = 2,tmax ! output: t, m(t), m(t-1)
     write(7,3000) tempo, m(tempo), m(tempo-1)
2000 continue
3000 format(2x,i6,4x,2(3x,g12.6))
     stop
     end
```

Program 1. Program to investigate small lattices.

The program starts defining working arrays `a(L,L,L,L)` and `m(tmax)`, with `L` and `tmax` corresponding respectively to the number of sites along each dimension and to the maximum time for which we want to follow the dynamics. The command `a=0` automatically sets all L^4 sites equal to zero and informs the compiler that `a` is an array in the Connection Machine. Both `call` statements are intrinsic functions of the CM, with the net effect of filling `a` with a uniform random distribution of "1" and "0". To count the total magnetization of the lattice it suffices to say `sum(a)`. Note the conveniency of FORTRAN 90 when dealing with array indexes. The same statement in FORTRAN 77 would require four `do` loops. This comment applies also to the initialization of `a`. The heart of the program is the `do` loop 1000. This loop first updates the lattice: to each site of `a` we sum the actual values of its ± 1 neighbors in 4 dimensions, according to the definition of a von Neumann neighborhood. This is done by circularly shifting the contents of `a`, using the function `cshift(a,dim,shift)`. Since `a` is four-dimensional, `dim` may be $1, 2, 3$ or 4. The parameter `shift` indicates by how much the dimension `dim` of `a` is to be shifted: von Neumann neighborhood implies either `shift`$= 1$ or -1. After this updating is performed the contents of each site, rather than being "1" or "0", are integer numbers ranging between 0 and 9. The `where` statement restores then "1" and "0" in all sites by broadcasting the proper automaton rule simultaneously to all processors. Therefore, after the `where` statement is executed, `a` contains the updated lattice. The last line in loop 1000 counts and stores the magnetization. At the end, the magnetizations are written on a file "aus" for later use. As can be easily noticed, although this program is convenient to study small symmetric `a(L,L,L,L)` as well as asymmetric `a(L1,L2,L3,L4)` lattices, it is obviously inefficient in both use of memory and computing resources. The most obvious limitation comes from the use of a full 32-bit integer to keep the binary state variable and the four-bits long

sum of nearest-neighbors (in 5-D the maximum number living nearest-neighbors is 11, counting the central site, and can be easily represented using 4 bits only).

Much improvement in the use of memory and computing resource can be achieved by dropping down to the level of the machine language PARIS on the CM (**PA**Rallel **I**nstruction **S**et). This level allows computations with integers of arbitrary lengths such as one-bit integers for the state variable and four-bit integers for the sum of neighbors. However such solution presents still some inefficiencies in dealing with very short field lengths.

Our second program uses a radically different approach and completely overcomes the deficiencies just mentioned. Moreover this approach, which uses the technique of multispin coding, can be programmed in a high level language such as Fortran 90 if one is willing to work with the restriction that one of the dimensions of the lattice be equal to 32 (the wordlength of the CM). This last restriction can be relaxed by dropping to the Paris level. This approach uses both memory and computing in an efficient way. The main point is to code one of the dimensions of the problem along the length of a parallel variable using the well known technique of multi-spin coding, and then to compute, using logical operations, the bit-wise representation of the sum of living nearest-neighbors.

For illustration (see program 2) we used a four-dimensional CM fortran array of integers (32 bits) to represent the state of a 5-D lattice with one of the axis being 32 bits wide. The trick now is to manipulate only words of 32 bits in order to compute the sum of all "living" nearest neighbors. As stated above this sum, a number between 0 and 11, can be represented with four bits. Let us call these bits s_0, s_1, s_2 and s_3. So the aim of the program is to calculate for each site the value of these bits, which give the binary representation of the sum of living nearest-neighbors for each site. For example, $(s_3,\, s_2,\, s_1,\, s_0) = (1,\, 0,\, 0,\,)$ indicates 9 living neighbors. These fours bits can be computed from logical operations on a using SHIFT operations to get access to the neighbors. Let's denote the 11 bits by b_0 to b_{11}. We start with

$$(s_3,\, s_2,\, s_1,\, s_0) = (0,\, 0,\, 0,\, 0)\,.$$

The usual way would go through the addition steps using a carry c in each binary place. First we add b_0 and b_1 and put the sum in s_0 and s_1 ($\wedge$ corresponds to the logical AND; $\otimes$ to XOR; $\vee$ to OR and $\neg$ to NOT):

$$\begin{aligned} s_1 &= b_0 \ \wedge \ b_1, \\ s_0 &= b_0 \otimes b_1\,. \end{aligned}$$

Then we add b_2 to s_0 generating a carry c (in binary place 1) to be added to s_1:

$$\begin{aligned} c &= s_0 \ \wedge \ b_2, \\ s_0 &= s_0 \otimes b_2, \\ s_1 &= s_1 \ \vee \ c. \end{aligned}$$

Note that since this is only the third bit added this last operation cannot generate a carry into s_2 as this would only be possible if the sum reached 4. Next, we add b_3 as follows:

$$\begin{aligned} c &= s_0 \ \wedge \ b_3, \\ s_0 &= s_0 \otimes b_3, \\ s_2 &= s_1 \ \wedge \ c, \\ s_1 &= s_1 \otimes c, \end{aligned}$$

continuing in a similar fashion until b_{10} is added.

We were able to reduce the number of logical operations involved in the method just described by using the technique of *full addition*.[6] According to this technique, three bits b_0, b_1, b_2 can be added together with 5 logical operations to generate the results r_0 and r_1. The operations are

$$\begin{aligned} r_1 &= (b_0 \ \wedge \ b_1) \ \vee \ (b_0 \otimes b_1) \ \wedge \ b_2, \\ r_0 &= (b_0 \otimes b_1) \otimes b_2 . \end{aligned}$$

This operation will now be represented using the notation:

$$(r_1, r_0) = fulladd(b_0, b_1, b_2) .$$

A "fulladd" involves only 5 operations because $b_0 \otimes b_1$ appears twice. The full adder can be used effectively by doing this 11-bit sum using two carry bits in each binary place. We'll denote the first set of carry by c_0, c_1, c_2 and c_3. Whenever a second carry c is generated in some binary place, the full adder is immediately invoked generating in the next binary place either the result, the carry c_n or the second carry c in the next place. At the start of the 11-bit addition, all the sum bits s_0, s_1, s_2 and s_3 and all the carry bits c_0, c_1, c_2 and c_3 are set zero (cleared). We proceed as follows:

$$s_0 \leftarrow b_0 \quad \text{"self add"} .$$

Then we add b_1 and b_2 using the full adder to generate s_0, s_1 :

$$(s_1, s_0) \leftarrow fulladd(b_0, b_1, b_2) .$$

Next we want to add b_3 and b_4 to s_0 generating a first carry in position 1:

$$(c_1, s_0) \leftarrow fulladd(s_0, b_3, b_4) .$$

This c_1 is a carry to be added to s_1. However we do not do this immediately. We will instead wait until a second carry is generated in this position. Having two carries we will then be able to use once again the full adder. So, now we are ready to add b_5 and b_6 to s_0, thereby generating the second carry c,

$$(c, s_0) \leftarrow fulladd(s_0, b_5, b_6) .$$

Since we now have two carries in position 1 we can use the full adder to compute s_2 and s_1. Since s_2 is known to be zero to start with, the carry to binary place 2 can go directly into it,

$$(s_2,\, s_1) \leftarrow fulladd(s_1,\, c_1,\, c)\,.$$

No more carry bit are set now. We are ready to add b_7 and b_8

$$(c_1,\, s_0) \leftarrow fulladd(s_0,\, b_7,\, b_8)\,.$$

This sets the carry c_1. We can again add b_9 and b_{10},

$$(c,\, s_0) \leftarrow fulladd(s_0,\, b_9,\, b_{10})\,.$$

We now have two carries in position 1. So we use the full adder to generate a carry in position 2 (s_2 was generated earlier),

$$(c_2,\, s_1) \leftarrow fulladd(s_1,\, c_1,\, c)\,.$$

We have no more bits to add but we still have a carry set in position 2. To complete the addition we must use a half adder on this carry c_2 and s_2,

$$s_3 \leftarrow s_2 \,\wedge\, c_2\,,$$
$$s_2 \leftarrow s_2 \otimes c_2\,.$$

This completes the update of the lattice. Using the full adder instead of a half adder reduces the number of logical operations from 49 to 37. As the dimension of the problem is increased the net gain in using the full adder over the half adder increases as well.

We now have computed the binary representation of the number of living nearest-neighbors for each site. We must decide if this number is in the window of allowed values, i.e., we must now apply the automaton rule. Our example considers rule $R^5_{5,9}$, for which the only surviving "animals" are those for which the number of neighbors is either 5, 6, 7, 8 or 9. The binary representation of these numbers is

	s_3	s_2	s_1	s_0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

This set of numbers can be expressed as the logical expression ("filter"):

$$\Big(s_3 \,\wedge\, (\neg s_2) \,\wedge\, (\neg s_1)\Big) \vee \Big((\neg s_3) \,\wedge\, (s_2) \,\wedge\, (s_1 \vee s_0)\Big)\,.$$

Extension to different rules presents no difficulty.

To compute the magnetization, one must count the number of bits set in field a. The low-level CM-2 function `CM_u_logcount_2_21` performs this sum in each processor. The sum accumulated in each processor is then globally summed over all processors using the Fortran 90 intrinsic function `sum()`.

As stated earlier, the neighbors are accessed using shift operations. For 4 of the dimensions this is done very simply by using the fortran intrinsic function `cshift` which circularly shifts an array along a given dimension. However we mentioned that the fifth dimension is multi-spin coded along the length of a 32-bit field. These 32 bits are kept inside each processor. There exist one fortran intrinsic function that will execute this "internal" circular shift of bits, namely `ishiftc()`. The program is the following:

```
integer, parameter :: L=16
integer, array(L,L,L,L) :: a,s0,s1,s2,s3,c0,c1,c2,c3,c
integer mag, t, tmax
tmax=1000 ! do 1000 iterations
do t=0,tmax-1
   ! add the number of living neighbors.
   ! generating the bit pattern for the result in
   ! s3-s2-s1-s0
   s0=a ! first add self
        !circular shift along the multi-spin coded axis
   c0=ishftc(a, 1,32)
   c =ishftc(a,-1,32)
   call fulladd(s0,c0,c,s1,s3) !/* max sum is 3 = 0011*/
        ! now do all the other axis
c0 = cshift(a,1,1)
c  = cshift(a,1,-1)
call fulladd(s0,c0,c,c1,s3) !/* max sum is 5 = 0101*/
c0 = cshift(a,2,1)
c  = cshift(a,2,-1)
call fulladd(s0,c0,c,c3,s3)
call fulladd(s1,c1,c3,s2,s3) !/* max sum is 7 = 0111*/
c0 = cshift(a,3,1)
c  = cshift(a,3,-1)
call fulladd(s0,c0,c,c1,s3) !/* max sum is 9 = 1001*/
c0 = cshift(a,4,1)
c  = cshift(a,4,-1)
call fulladd(s0,c0,c,c3,s3)
call fulladd(s1,c1,c3,c2,s3) !/* max sum is 11 = 1011 */
          ! terminate the addition
```

```
      s3=iand(s2,c2)!/* hald-adder : for a max of 11 the last */
      s2=ieor(s2,c2)!/* 2 bits cannot be both 1 */
    !/* whose living; whose dead 5<=sum<=9 */
    !/*
       !this test bit pattern it has to be adapted to the
       !living window. (could it be better done?...)
    !this uses c1 as temp storage.
    !there exist one paris call for each line.
    !*/
        a=ior(s1,s0)
        a=iand(a,s2)
        a=iand(a,not(s3))
        c1=iand(s3,not(s2))
        c1=iand(c1,not(s1))
        a=ior(a,c1)
  write (6,*) "x index ", (1.0*mag(a,s0))/(n*n*n*n*32.0), " put "
     end do
     ! write configuration ...
    end
    subroutine fulladd( b0,b1,b2,r1,t)
C Do a full add of bits b0 b1 b2 putting them in r1 (most significant bit
C (msb)) and b0 (least significant bit (lsb)). In other words,
C take (r1 b0) = b0+b1+b2;
C b0 and r1 are the results; t is used temporary;
C b1 and b2 are not changed
     integer, parameter :: L=16
     integer, array(L,L,L,L) :: b0,b1,b2,r1,t
     r1 = iand(b0,b1)
     b0 = ieor(b0,b1)
     t = iand(b0,b2)
     r1 = ior(r1,t) !msb
     b0 = ieor(b0,b2) !lsb
     end
     integer function mag(a,s0)
C Compute the global magnetisation
C This just counts the total number of bit set in field a
C This subroutine uses temporary storage s0
     integer, array(:,:,:,:) :: a,s0
     integer m
     m=0
     call CM_u_logcount_2_21(s0,a,32,32)
        ! counts bits in 'a', storing the result in s0
     m = sum(s0) ! sums now over all processors
```

```
mag=m
end
```

Program 2. Program to investigate large lattices.

4. Small Lattice Behaviour

In this section we use the first program given in the previous section to investigate a number of questions related with the behavior of all three rules for small lattice sizes. We start by studying the automata in 5 dimensions (rule $R^5_{5,9}$) and consider first *symmetrical* lattices, namely lattices having an equal number L of sites along each dimension. We followed during 15,000 time steps the evolution of lattices having 3^5, 4^5, ... , 16^5 sites and plotted their $m(t) \times m(t+1)$ evolution-spaces. For 3^5 and 4^5 one obtains after a quite long transient, attractors involving just a few points. For example, $m(t)$ was found to oscillate between five values for $L = 3$ and between two values for $L = 4$. Figure 2 shows the evolution-space for a few L values. From it one sees the *genesis* of the hole inside the triangular-like attractor. The figure shows the magnetizations between times 100 and 15,000. If in this figure only $m(t)$ values for, say, $9{,}000 \leq t \leq 15{,}000$ were plotted, with exception of $L = 4$ (where the attractor is just given by two dots symmetrically located with respect to the diagonal), all other evolution-space plots would have essentially the same shape, containing of course less points in them. For comparison purposes we also included $L = 16$ in this figure.

Would it be possible to have a kind of *resonance* inside the lattice, responsible for periodic and quasiperiodic behaviors? To get some insight, we considered the time evolution of *asymmetrically* shaped lattices with $4^4 \times 5$, $4^3 \times 5^2$, $4^2 \times 5^3$ and 4×5^4 sites. For the first three lattices, after a reasonable transient of about 1000 time steps, the systems landed into a periodic attractor having, for example, 2, 12 and 4 points respectively. The exact number of points on the final attractor depends on the initial configuration. The 4×5^4 lattice converged after a little longer transient. This could be a sign for the existence of a bigger transient of the 5^5 lattice in Fig. 2. We investigated this possibility by further evolving both lattices for additional 200,000 time steps. The conclusion is that indeed, for the five different initial configurations for which we followed such long evolution of the magnetization, the system with 4×5^4 sites always landed on a periodic orbit, with period varying from two to twelve dots. In contrast, for 5^5 sites the system did not reach periodic attractors. The "final" attractor was apparently reached for much less than 100,000 time steps in all cases.

To better check the possibility of a transient we further investigated the 7^5 lattice for longer times, searching for an eventual transition to a smaller and periodic attractor. The result found by considering a few different initial configurations and following the dynamics for 200,000 time steps is that it always stays on quasiperiodic orbits.

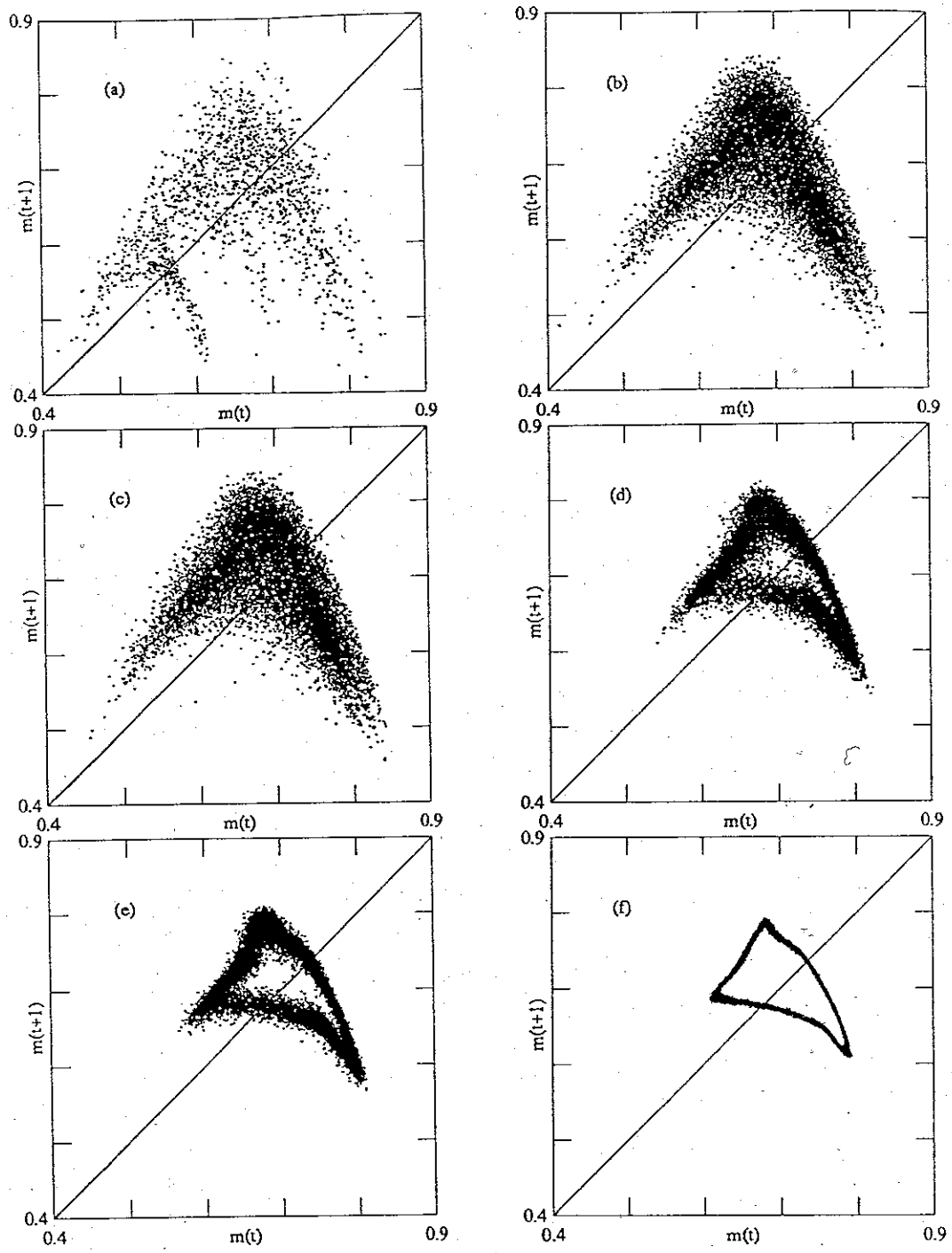

Fig. 2. $m(t) \times m(t+1)$ for rule $R^5_{5,9}$ in symmetric lattices and for $100 < t < 15{,}000$. (a) $L = 4$; (b) $L = 5$; (c) $L = 6$; (d) $L = 7$; (e) $L = 8$ and (f) $L = 16$. Notice the genesis of the quasiperiodic attractor as L increases ($L = 32$ is given in Fig. 1).

Program 1 above was used to investigate lattices containing no more than $32^4 = 1{,}024^2 = 1{,}048{,}576$ sites. Program 2 was used for lattices containing between 32^4

and $32^5 = 33{,}554{,}432$ sites. Our regular runs were iterated for 10,000–15,000 time steps. But some selected runs were extended up to 200,000 time steps.

5. Transients and Dependence on the Initial Concentration

All plots given in Ref. 3 as well as those in our Fig. 1, were obtained by starting from configurations having an equal number of "1" and "0" randomly distributed on the lattice at $t = 0$. What happens if one starts from an initial magnetization $m(0) \neq 0.5$? The answer for rule $R^4_{4,8}$ is given in Fig. 3, for $m(0) = 0.3$: although the final attractor does not seem to change, the way in which the attractor is approached varies substantially. Note that in this figure we did not remove the initial transient. From 0.3 the magnetization increases continuously along the diagonal, converging first to the inner part of the "triangular" attractor. From there it spirals to the final attractor, given by the thicker line, defining the triangle.

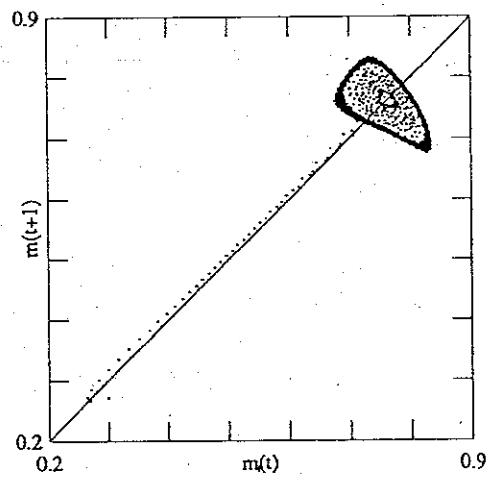

Fig. 3. The first 5,000 points for $R^4_{4,8}$, $L = 32$ and $m(0) = 0.30028$.

Figure 4 displays the transients for rule $R^4_{3,8}$, for two different initial values of the magnetization, namely 0.520 and 0.579 in Figs. 4a and 4b, respectively. One sees that a relatively modest change on the initial density of "1" in the lattice is enough to induce a transition from periodic to a fixed point behavior.

For very small systems one might also find fixed points or periodic behavior between very closely lying points. In this latter case the transients can show remnants of the characteristic triangular shape of the attractor for the larger lattices.

For rule $R^5_{5,9}$ the observed behavior is essentially analogous to that found for rule $R^4_{4,8}$.

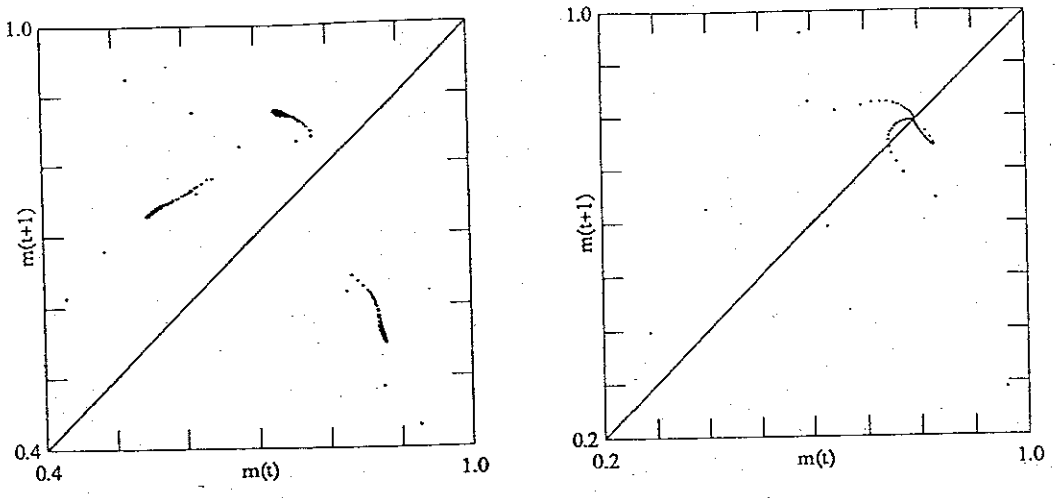

Fig. 4. The first 5,000 points for $R^4_{3,8}$, $L = 32$ and (a) $m(0) = 0.52022$ and (b) $m(0) = 0.57986$.

6. Finite Time and Size Effects

In order to monitor whether the attractors are also stable after a long time evolution or for infinite system sizes one should define a quantitative measure for the fluctuations occurring around the attractor (see Fig. 1). One way of doing this, based on the $\theta^{t+3} = g(\theta^t)$ map introduced by Chaté and Manneville,[3] is the following: one chooses a point $P = (m_c, m_c)$, located on the diagonal and roughly in the "center" of the attractor. Then, for each point $(m(t), m(t+1))$ one finds the angle $\theta(t)$ that corresponds to the point on a polar coordinate system centered at P and with $\theta = 0$ corresponding to the direction of the positive $m(t)$-axis. From the several angular variables so obtained one draws a $\theta(t) \times \theta(t+3)$ graph as shown in our Fig. 5 (see also Fig. 1c of Ref. 3). One finds a wiggly line along the diagonal. The diagonal is then divided into $N_{\rm bin}$ equal intervals (bins). For each bin one evaluates the *mean square deviation* of the orthogonal distances from the diagonal of each data point within the interval with respect to its mean value. The quantity Δ we want to use to characterize *angular* fluctuations is then the average over all these mean square distances over the $N_{\rm bin}$ intervals. In other words, we introduce

$$\Delta = \frac{1}{N_{\rm bin}} \sum_{i=1}^{N_{\rm bin}} \sqrt{\frac{1}{N_i} \sum_{j=1}^{N_i} (x_j^2 - \langle x_i \rangle^2)}\,, \tag{3}$$

where the first sum goes over all bins i, the second sum goes over the N_i points inside each bin and $x_j = \|\theta(j) - \theta(j+3)\|/\sqrt{2}$ is the distance of the point j from the diagonal. The mean value, as usual, is taken to be $\langle x_i \rangle = (1/N_i)\sum_j x_j$. In order not to have a strong dependence of Δ on $N_{\rm bin}$ it is important to drop the swarm of points on the upper left corner of Fig. 5. Our calculations were done using $N_{\rm bin} = 100$.

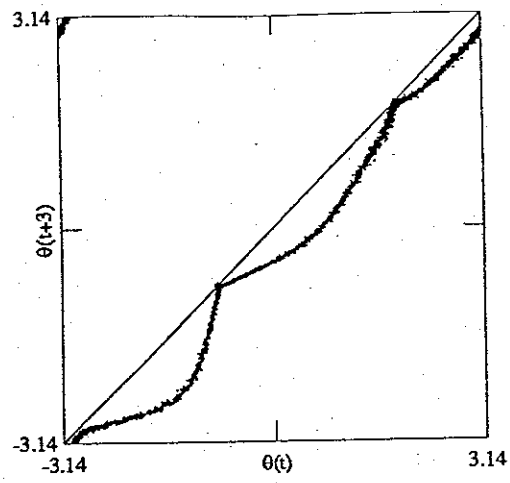

Fig. 5. Polar representation of $\theta(t) \times \theta(t+3)$ used to obtain Δ (see Eq. (3) and the text). Here given for $R^5_{5,9}$, $L = 16$ and $100 < t < 5000$.

To see how Δ varies with time we define $\Delta(\tau)$ as the Δ obtained by considering only those points after the "transient" τ and the maximum time for which we followed the dynamics. Such $\Delta(\tau)$ always reached almost instantaneously a constant value, which we call Δ_∞. Its values remained const. So, we see that for a finite size the attractor is not perfectly periodic but that it consists of three small clouds as shown in Fig. 1, even after waiting an infinite time.

Comparing the several plots in Fig. 1 one notices that the cloud becomes smaller with increasing system size. To investigate this size effect one can get in a similar way the values of Δ_∞ for other system sizes L. In Fig. 6 we see in a log-log plot how Δ_∞ depends on L. Since the data fall reasonably well on a straight line it seems that the width Δ_∞ goes to zero with a power-law $\Delta_\infty \propto L^{-3.5}$. A fit of the data to an exponential decay is less good.

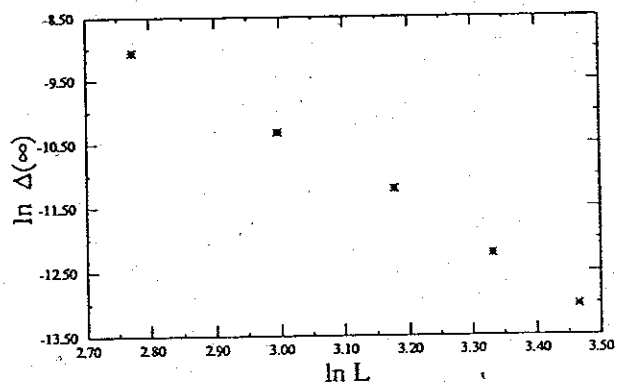

Fig. 6. Log-log plot of $\Delta(\infty)$ as a function of L for rule $R^4_{3,8}$, here for $L = 16$, 20, 24, 28 and 32.

This numerical analysis leads us to conclude that in the thermodynamic limit, the attractor for rule $R^4_{3,8}$ will become perfectly periodic since the cloud found for finite systems even after very long times collapses to three points. Similarly one finds for rules $R^4_{4,8}$ and $R^5_{5,9}$ that the attractor converges more and more towards a single line, i.e., limit cycle, when the system size increases. So we confirm that the reported effect survives when the system size goes to infinity; even more: only in the thermodynamic limit one can really speak of a period three or of a limit cycle behavior.

7. Phase Transitions in the Presence of Noise

An interesting question to be asked is how stable these attractors are with respect to noise. We therefore introduce some noise into the model by flipping after each update of the system a fraction f of sites, i.e., to turn zeros into ones and vice versa. Interestingly the attractor is unchanged for a small amount of noise as seen in Fig. 7a for rule $R^4_{3,8}$. When a critical noise $f^* = 0.0225 \pm 0.0045$ is reached the period suddenly disappears as shown in Fig. 7b and one finds a single fixed point which is also what one expects in the limit of infinite noise where $m(\infty) = 1/2$. So we find a first order transition in "temperature" between a periodic behavior at low temperatures and a disordered phase at high temperatures.

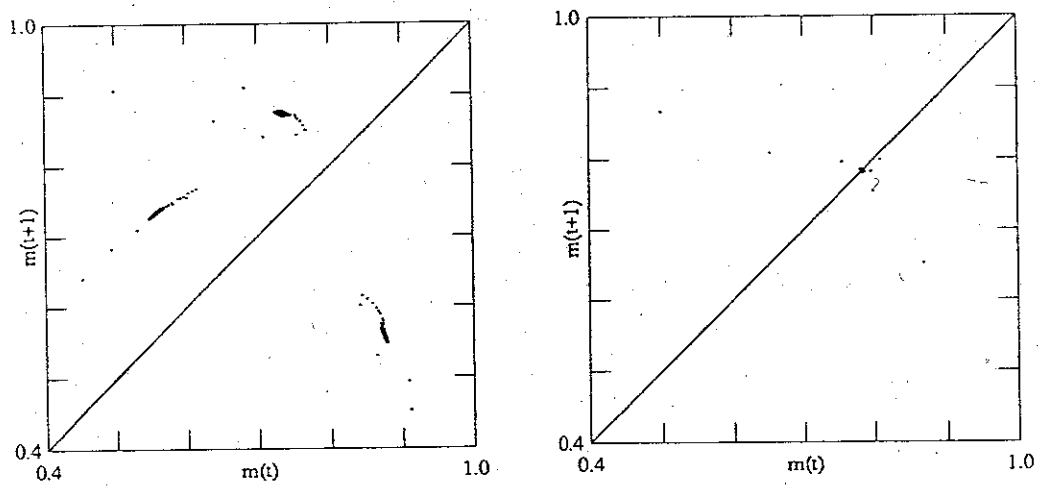

Fig. 7. Phase-space for $R^4_{3,8}$, $L = 4$, when a fraction f of sites is randomly flipped at each time-step for (a) $f = 0.005$ and (b) $f = 0.05$.

8. Discussion

We have shown how to efficiently program on the CM–2 the totalistic automata introduced by Chaté and Manneville. Our numerical results have confirmed the existence of periods and limit cycles for certain rules in the limit of infinite times and sizes. This behavior is stable with respect to small noise or variations in the initial

magnetization. When either of these perturbation increases first order transitions are found to single fixed points.

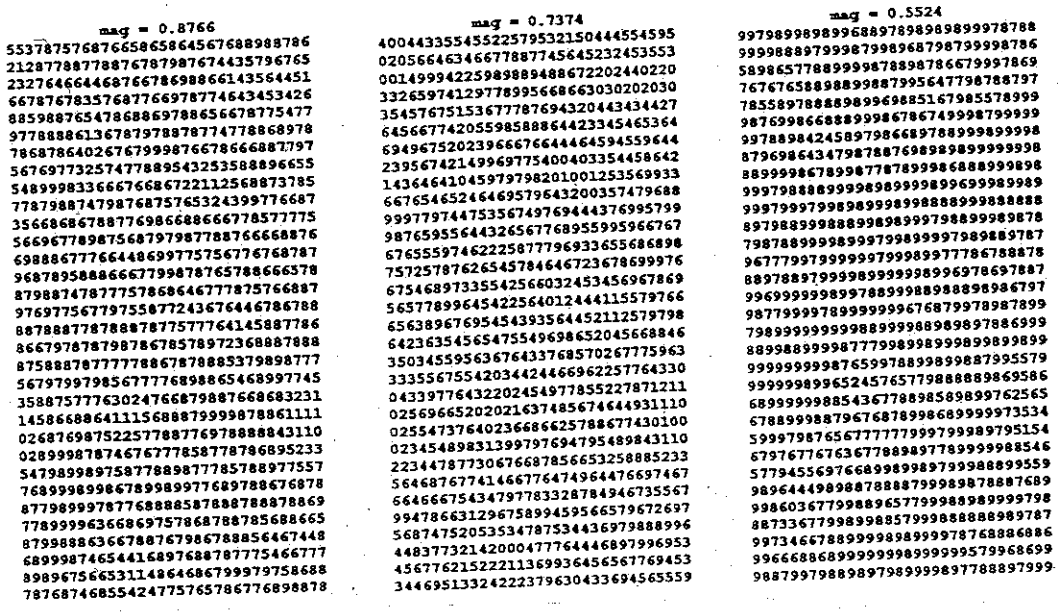

Fig. 8. Two consecutive views of the plane a(1,1,i,j) for $i, j = 1, \cdots 32$, after a transient of 9,000 time steps, for rule $R^4_{3,8}$ with $L = 32$. The numbers represent the sum of nearest-neighbors of each site. The average magnetization of the full lattice in each view is also indicated.

The question remains what the nature is of these unusual periodic and quasi-periodic phases. Even considering the simpler case of the periodic rule $R^4_{3,8}$ it is difficult to see how the system can synchronize the local magnetizations such that the global magnetization can have the observed oscillations. One could for instance imagine having a regular spatial pattern, i.e., some order which propagates the necessary information. To investigate this issue we show in Fig. 8 for rule $R^4_{3,8}$ a two-dimensional slice through a 32^4 lattice at three consecutive time steps after the attractor has been reached. On each site we have a number between 0 and 9 which is the h_i defined in Eq. (1). It is striking not to notice any regularity whatsoever. The same applies when analyzing sequences three time-steps apart, i.e., having the same global magnetization.

The three values of the global magnetization at consecutive times is given on top of each configuration in Fig. 9. They are close to 8/9, 7/9 and 5/9 but the deviations from these values are significant even after extrapolating to infinite time and size. There is therefore no evident symmetry or regularity to be observed in the system.

Another possibility is to see if our results could be explained by a mean-field approximation. Let us consider only the densities $\rho_n(t)$ of sites having a given value of $h_i = n$ at time t. Then for rule $R^4_{3,8}$ the magnetization at the next time step is given by $m(t+1) = 1 - \rho_0(t) - \rho_1(t) - \rho_2(t) - \rho_9(t)$. The ρ_n can themselves be expressed in terms of $m(t)$ by counting all possible local configurations having

$h_i = n$. So one has for instance $\rho_2(t) = 36m^2(t)(1-m(t))^7$. Inserting this into the expression for $m(t+1)$ one finds the mean-field recursion relation for the global magnetization:

$$m(t+1) = 1-(1-m(t))^9 - 9m(t)(1-m(t))^8 - 36m^2(t)(1-m(t))^7 - m^9(t)\,. \quad (4)$$

This equation can be iterated and the corresponding plot of $m(t+1)$ as function of $m(t)$, starting from $m(0) = 0.8766$, is shown in Fig. 9. We see no resemblance to the attractors we have observed in finite dimensions and in particular no periodic attractor. Similar plots were obtained starting from $0.2 \leq m(0) \leq 0.9$. Outside this range, unphysical results were obtained. Therefore we conclude that the above mean-field approximation is not very useful in the present case.

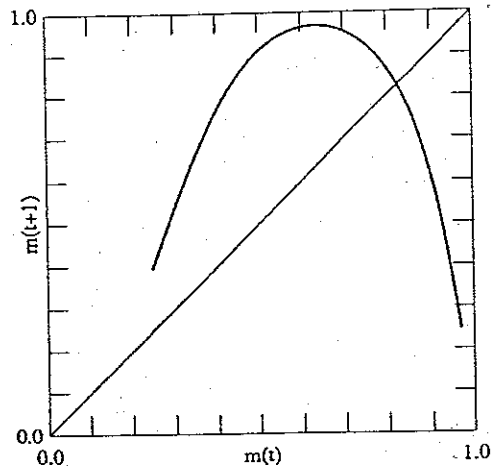

Fig. 9. The attractor $m(t) \times m(t+1)$ as predicted by the mean-field equation (4), starting from $m(0) = 0.8766$, one of the values in Fig. 8.

We showed that periodic and quasiperiodic behavior in the global magnetization is stable for large sizes and times and under small noise and perturbations in the initial configuration (up to a first order transition). But a deeper understanding of the phenomenon is still missing. Certainly the scenario is not simple because no spatial structures could be found and mean-field did not help. More investigations are needed in order to understand the order–propagating mechanism in these phases.

We thank H. Chaté for discussions and for informing us about his work. JACG thanks CNPq, Brazil, for partial support. The authors also thank Bastien Chopard and Thaisa S. Bergmann for help in handling Post-Script and SuperMongo files. Our calculations were performed on the CM–2 Connection Machine at the GMD, Bonn.

References

1. T. Bohr, G. Grinstein, Y. He, and C. Jayaprakash, *Phys. Rev. Lett.* **58**, 2155 (1987).
2. G. Grinstein, *J. Stat. Phys.* **51**, 803 (1988).
3. H. Chaté and P. Manneville, *Europhys. Lett.* **14**, 409 (1991).
4. J. A. C. Gallas, P. Grassberger, H. J. Herrmann, and P. Ueberholz, *Physica* **A180**, 19 (1992).
5. H. Chaté and P. Manneville, private communication and preprint in preparation.
6. M. Neschen, *Int. J. Mod. Phys.* **C2**, 623 (1991).