An automaton for propagating waves through heterogeneous media

Francis Muir

INTRODUCTION

The cellular automaton is a model for a dynamical system with some characteristic features:

A regular lattice

An element from a small, closed set at each lattice node

A uniform set of local rules of evolution on these elements

It thus differs from usual schemes in that there is no continuum equation to be approximated by a finite difference scheme, and no real variables to be rounded-off by floating-point representations. The cellular automaton is a prescription for computation which is also a physical model.

THE MODEL

The scheme described in this paper differs in two respects from conventional dynamical system automata, such as the Navier-Stokes model discussed in the current issue of Physics Today (Sept '86). In the first place, there is no sense of a few particles floating in an otherwise empty void; to the contrary, the positive particles, the words, are evenly balanced by their negative counterparts. In the second place, the rules are not necessarily uniform, but will be spatially variable to introduce the desired heterogeneity into the model.

The lattice is rectangular in two space dimensions, and models evolve in time. The lattice is either bounded, in which case I use an absorbing boundary condition, or it is mapped on to a torus by using circular boundary conditions. The node element is a set of four words, FORD, BACK, HUPP, and DOWN, and each can take on one of the two values, +1, −1. The element

field is initialized by assigning plus or minus unity to each word in each element in a random fashion. An initial condition, a spike, might consist of a 64-by-64 square of elements whose words are all set to plus unity. In the homogeneous version of the automaton, there are two rules which alternate in time. The first is a SWAP rule which is applied at each node:

```
IF (ford.EQ.back) THEN
        IF (hupp.EQ.down) THEN
                IF (ford.NE.hupp) THEN
                        ford = - ford
                        back = - back
                        hupp = - hupp
                        down = - down
                END IF
        END IF
END IF
```

and the second is a MOVE rule:

```
DO k = 1,kmax
        DO j = 1,jmax
                ford(j,k) = ford(j - 1,k)
                back(j,k) = back(j + 1,k)
                hupp(j,k) = hupp(j,k + 1)
                down(j,k) = down(j,k - 1)
        END DO
END DO
```

SWAPping takes place one in eight time-steps, on the average. If there was no SWAP, energy would not spread out evenly in a circle, but would just travel out along the grid directions. Note that both the SWAP and MOVE rules are invertible, which makes the scheme a reversible scheme. That is, it is reversible at this micro level. It will turn out not to be so at the averaged, macro level. Note that the result at this point depends upon the particular realisation of the initial random field. We could repeat this experiment a number of times, using different initial random fields, and then take an ensemble average. Instead, we will invoke an (unproven) ergodic principle, and average spatially. That is, divide the space into, say, 16-by-16 element squares, and sum each of the words over that square:

```
bigford = 0
bigback = 0
bighupp = 0
bigdown = 0
DO kk = 1,16
        DO jj = 1,16
                bigford = bigford + ford(jj,kk)
                bigback = bigback + back(jj,kk)
                bighupp = bighupp + hupp(jj,kk)
                bigdown = bigdown + down(jj,kk)
        END DO
END DO
```

It is these compact arrays which are output at each time step, leaving the uncompacted ones to continue in the iteration scheme. The compact arrays are also further compacted over time:

```
maxford = 0
maxback = 0
maxhupp = 0
maxdown = 0
DO tt = 1,16
        maxford = maxford + bigford(tt)
        maxback = maxback + bigback(tt)
        maxhupp = maxhupp + bighupp(tt)
        maxdown = maxdown + bigdown(tt)
END DO
```

The final step is to sum the four maxwords:

```
out = maxford + maxback + maxhupp + maxdown
```

and we are done. All this at each of, say, 64 big time steps.

ILLUSTRATION

Fig.1 is a snapshot of a wave expanding from an entirely positive-going initial symmetric blob

set in a sea of random bits. It is the 23rd frame in a 64-frame Movie, and a frame has 64 by 64 elements, each the sum of 16,382 signed bits, representing compression factors of 16 in each of the dimensions of time and space, and a factor of 4 from summing the four component words, FORD, BACK, HUPP, and DOWN, from each element in the automaton. Still the picture is quite ragged. Worthy of note is the negative-going inner ring to the wavelet (here plotted in reversed polarity); that is, this automaton has correctly modeled the hankel tail expected in a 2-space. In general, automata properly handle those aspects of a problem that are geometric in origin, or dependent on conservation or continuity. So we may expect most wave phenomena to be properly modeled. Our task is to ensure a degree of isotropy, and to make sure that the appropriate conservative constraints are built in to the system.

VELOCITY VARIATION

Body waves travel through this medium at a constant 0.707 nodes per timestep. There are several ways to introduce velocity variablity, and I have used a stochastic scheme, wherein:

```
EITHER
        SWAP and MOVE
OR
        DO NOTHING
```

according to the outcome of casting some suitably loaded dice. More recently I have decided to go with a deterministic scheme which will tie the DO SOMETHING OR NOTHING decision to the value of the time step modulo some number like 16.

IMPEDANCE VARIATION

Notions of velocity and impedance are kept seperate. At impedance boundaries, the normal, homogeneous MOVE rule is modified to accomodate the possibility of EITHER reflection, OR transmission, again using a stochastic scheme based on the desired relative amplitudes of the reflected and transmitted waves. So that:

```
IF (heads) THEN
        ford(j,k) = ford(j - 1,k)
ELSE
```

```
            IF (impedance increase) THEN
                    ford(j,k) = back(j,k)
            ELSE
                    ford(j,k) = - back(j,k)
            END IF
    END IF
```

which also allows for the possibility of a change of sign on reflection. Again, I will now probably switch to a deterministic scheme, closely matching the velocity variable one. At this point I should also note that I use SQRT(energy) variables, not pressure or displacement. This gets around the problem of amplitude increasing as impedance decreases. This automaton allows for attenuation but not gain. Once out of the micro world and in the summed system, the output field can be transformed to pressure or displacement.

DISCUSSION

I am satisfied with my current progress, and particularly pleased with the way that cellular automaton design appears to be well adapted to the SEP way of doing business. The important questions, as yet unanswered, are:

Does it really work?

Is it cost effective?

Future plans include one-way wave equations, and inversion schemes.

ACKNOWLEDGMENTS

My interest in cellular automata was sparked by Dan Rothman, now Assistant Professor of Geophysics at MIT, and he continues to be a considerable source of wisdom, information, and criticism. For this piece of work I had to learn unix, vi, and seplib in a hurry, and this would have been impossible without the friendly help of the whole sep crew here. In the end, Joe Dellinger gave up on teaching me LaTeX in an hour, and took over the job of getting this article ready for press.

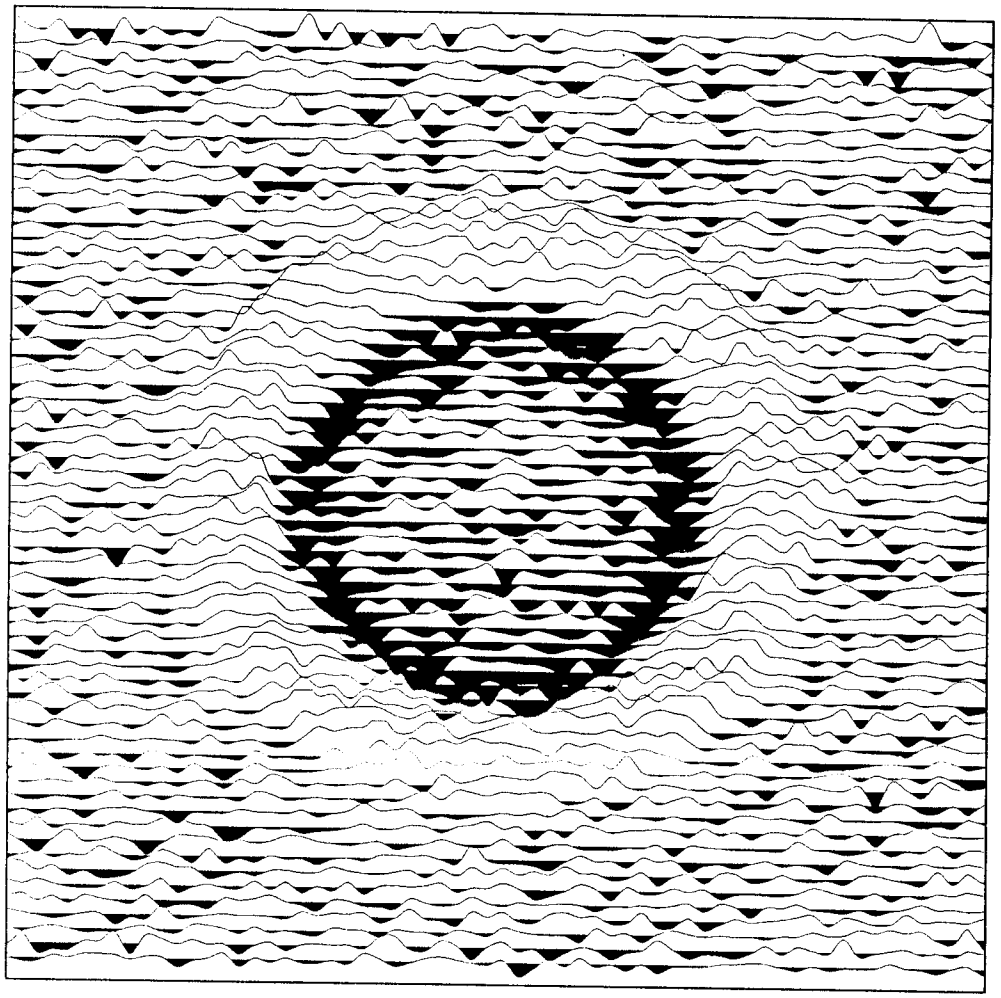

FIG. 1. Snapshot of a wave expanding out from an initial blob