Complex Systems **10** (1996) 91–119

Continuous-valued Cellular Automata for Nonlinear Wave Equations

Daniel N. Ostrov
Department of Mathematics,
Santa Clara University,
Santa Clara, CA 95053

Rudy Rucker
Department of Mathematics and Computer Science,
San Jose State University,
San Jose CA, 95192

Abstract. In 1955, E. Fermi, J. Pasta, and S. Ulam investigated quadratic and cubic schemes for numerically simulating nonlinear waves. We derive the partial differential equations corresponding to the Fermi–Pasta–Ulam schemes, present a discussion of the accuracy and stability of different schemes for the equations, and implement the schemes as continuous-valued celluar automata (CA). Some illustrative runs of CAPOW, a CA simulator, are presented which demonstrate the behavior of the different schemes both in stable and unstable domains. These runs include a confirmation of an observation in [2] regarding the approximate temporal periodicity of the quadratic and cubic nonlinear wave equations.

1. Introduction

Stanislaw Ulam can be thought of as the father of cellular automata (CA). He began investigating discrete CA in 1950 [1]. In 1955, he coauthored the famous paper [2], with Fermi and Pasta, which has been the starting point of many works in nonlinear waves and solitons.

In an introduction to [2], written by Ulam in 1965, he remarks that:

> "Fermi expressed often a belief that future fundamental theories in physics may involve nonlinear operators and equations, and that it would be useful to attempt practice in the mathematics needed for the understanding of nonlinear systems. The plan was then to start with the possibly simplest such physical model and to study the results of the calculation of its long-time behavior." [3]

In [2], nonlinear models for particles attached horizontally by springs where the particles can move vertically are examined. We refer to this model as the *FPU model*, two algorithms for the resulting wave motion are considered in [2]. The first corresponds to a quadratic nonlinearity:

$$\begin{aligned} \ddot{U}_j &= (U_{j+1} - U_j) - (U_j - U_{j-1}) \\ &\quad + \alpha_1 \left((U_{j+1} - U_j)^2 - (U_j - U_{j-1})^2 \right); \end{aligned} \tag{i}$$

the second corresponds to a cubic nonlinearity

$$\begin{aligned} \ddot{U}_j &= (U_{j+1} - U_j) - (U_j - U_{j-1}) \\ &\quad + \alpha_2 \left((U_{j+1} - U_j)^3 - (U_j - U_{j-1})^3 \right), \end{aligned} \tag{ii}$$

where U_j represents the vertical position of the jth particle and $\ddot{U}_j$ is the second derivative of the jth particle position with respect to time. It is pointed out in [2] that for each of these schemes "(t)he corresponding partial differential equation... is the usual wave equation plus nonlinear terms of a complicated nature."

In section 2 of this paper, we determine the precise form of the nonlinear terms of the hyperbolic partial differential equations (PDEs) that correspond to schemes (i) and (ii). In section 3 we discuss the accuracy and stability of CA simulations for the FPU model with an eye to finding reasonable parameter values to use in running the nonlinear wave schemes. In particluar, we address the role of characteristics in determining the Courant–Frederichs–Lewy stability condition.

In the past it has been customary to study only CA that have a finite number of discrete state values. In section 4, we show how it is natural to extend the idea of CA to rules which allow continuous-valued state variables. We describe how specific continuous-valued CA simulations for the linear and nonlinear waves were finally arrived at, resulting in the shareware program CAPOW [4].

Section 5 shows some illustrations generated by CAPOW and discusses their meaning. In particular, we examine the behavior of the quadratic and cubic nonlinear waves, study the ways in which instability can set in for nonlinear waves, and confirm an observation from [2] that a nonlinear wave seeded with a sine wave will eventually cycle back to a close (but not exact) approximation of the initial sine wave seed. Further, we examine a way in which unstable nonlinear wave simulations can be made to behave like discrete-valued CA. This effect occurs when the possible cell values are clamped to lie in a bounded range. A clamped unstable CA rule bounces among the minimum and maximum values and a few in-between values, producing what is effectively a discrete-valued CA. To coin a term, we refer to clamped unstable continuous-valued CA with a finite range of values as "pseudodiscrete CAs." Our experiments indicate that pseudodiscrete CAs can be of Wolfram class 1, 2, 3, or 4, that is, single-valued, periodic, chaotic, or complex [5].

2. Fermi–Pasta–Ulam wave equation

We begin by describing the nonlinear string in the FPU model in more detail. When the string is at rest it lies along the x-axis. When the string moves it only moves in the y direction. The string is modeled by considering a series of particles along the x-axis with fixed x coordinates that are separated by a distance Δx. It then follows that the mass of each particle is $\rho\Delta x$, where ρ is the constant density of the string. The motion of each particle in the y direction is modeled by considering the effect of springs connecting each particle to its two neighbors. In Rheology, long polymeric molecules are often modeled by considering these "bead-spring" type models, where each bead (particle) represents 10 to 30 monomer units. When the springs are assumed to follow Hooke's law, these macromolecular models are called *Rouse chains*; when the springs are not hookean, they are called *Rouse–Zimm* chains [6].

Let $u(x_0, t_0)$ represent the vertical displacement (i.e., the value of y) at time t_0 of the particle whose horizontal location is x_0, and let Δu represent the difference in the vertical displacement between two neighboring particles. We can express the distance Δd between two neighboring particles as $\sqrt{(\Delta x)^2 + (\Delta u)^2}$. The magnitude of the force for most springs will usually be some function G of this distance (divided by Δx so that the force is normalized), that is,

$$||\mathbf{F}_{\text{sp}}|| = G\left(\frac{\Delta d}{\Delta x}\right). \tag{1}$$

Since it is assumed that the particles do not move in the x direction, we are only concerned with the vertical component of the spring force. The absolute value of the vertical component of the force is given by

$$F_{\text{sp,vert}} = G\left(\frac{\Delta d}{\Delta x}\right)\frac{\Delta u}{\Delta d}, \tag{2}$$

which we can express strictly as a function of $\Delta u/\Delta x$:

$$F_{\text{sp,vert}} = G\left(\sqrt{\left(\frac{\Delta u}{\Delta x}\right)^2 + 1}\right)\frac{\frac{\Delta u}{\Delta x}}{\sqrt{\left(\frac{\Delta u}{\Delta x}\right)^2 + 1}}. \tag{3}$$

If the spring is linear (i.e., follows Hooke's law) and has an equilibrium length of $z = 0$, as is the case for Rouse chains, then $G(z) = kz$, where k is the positive-valued spring constant, and therefore $F_{\text{sp,vert}} = k\Delta u/\Delta x$. For linear springs with nonzero equilibrium lengths or nonlinear springs where $G(z)$ is an analytic function near $z = 1$, we can express $G(z)/z$ as a series of even powers of $\Delta u/\Delta x$ when $\Delta u/\Delta x$ is small, which leads to the following expression for $F_{\text{sp,vert}}$:

$$F_{\text{sp,vert}} = a_1\frac{\Delta u}{\Delta x} + a_3\left(\frac{\Delta u}{\Delta x}\right)^3 + a_5\left(\frac{\Delta u}{\Delta x}\right)^5 \cdots . \tag{4a}$$

If the spring force is an analytic function of Δu instead of an analytic function of distance, we may also have even powers in the series:

$$F_{\text{sp,vert}} = a_1\frac{\Delta u}{\Delta x} + a_2\left(\frac{\Delta u}{\Delta x}\right)^2 + a_3\left(\frac{\Delta u}{\Delta x}\right)^3 + a_4\left(\frac{\Delta u}{\Delta x}\right)^4 \cdots . \tag{4b}$$

Now we consider the effect of the forces on a generic particle, located at x_0. If we define $\Delta u_- = u(x_0) - u(x_0 - \Delta x)$ and $\Delta u_+ = u(x_0 + \Delta x) - u(x_0)$ then we can express Newton's law by using (4b) to describe the effect of each of the two springs acting on the particle:

$$\begin{aligned}\rho\Delta x u_{tt}(x_0,t) \;=\; & a_1\left(\frac{\Delta u_+}{\Delta x} - \frac{\Delta u_-}{\Delta x}\right) + a_2\left(\left(\frac{\Delta u_+}{\Delta x}\right)^2 - \left(\frac{\Delta u_-}{\Delta x}\right)^2\right) \\ & + a_3\left(\left(\frac{\Delta u_+}{\Delta x}\right)^3 - \left(\frac{\Delta u_-}{\Delta x}\right)^3\right)\cdots .\end{aligned} \tag{5}$$

As Δx approaches 0 in equation (5), we obtain the nonlinear PDE corresponding to the FPU model:

$$\rho u_{tt} = a_1 u_{xx} + 2a_2 u_x u_{xx} + 3a_3 u_x^2 u_{xx} + 4a_4 u_x^3 u_{xx}\ldots . \tag{6}$$

We consider some special cases of our model. If the springs follow Hooke's law and have an equilibrium length of 0, then the nonlinear terms disappear and we are left with the standard linear wave equation

$$\rho u_{tt} = a_1 u_{xx} \tag{7}$$

where $a_1 = k$, which is the spring constant. If u_x stays small in the nonlinear model we can ignore the effects of higher order terms. This leads to the quadratic FPU equation

$$\rho u_{tt} = a_1 u_{xx} + 2a_2 u_x u_{xx}, \tag{8}$$

which the scheme in equation (i) simulated. When the spring force is strictly a function of distance, we work with equation (4a) instead of (4b), so $a_2 = a_4 = a_6 = \cdots = 0$. For this common case, we obtain the cubic FPU equation when u_x stays small

$$\rho u_{tt} = a_1 u_{xx} + 3a_3 u_x^2 u_{xx}, \tag{9}$$

which the scheme in equation (ii) simulated. Equations (7) through (9) will be the three equations of interest in this paper.

3. Accuracy and stability

Because of the discrete nature of computers, we can only approximate the solution to a PDE at certain grid points on the (x,t) plane. We label these points of approximation (x_j, t^n) where $j = 0,1,2,\ldots,J$ and $n = 0,1,2,\ldots,N$ and employ the abbreviations $U_j^n = U(x_j,t^n)$ and $u_j^n = u(x_j,t^n)$, where $U(x_j,t^n)$ is the numerical approximation at the grid points of interest for

$u(x_j, t^n)$, the actual PDE solution. We are interested, of course, in making $|U_j^n - u_j^n|$ as small as possible. To accomplish this we must consider the *accuracy* and the *stability* of our approximation scheme.

The accuracy of a scheme is really just a matter of minimizing the error term in the Taylor appoximation for the scheme. Because of its simplicity, and its applicablity to equations (7) through (9), we use as an example the first order linear traveling wave equation

$$u_t = -au_x \qquad \text{where } a > 0. \tag{10}$$

From Taylor's formula we have that

$$u_{j+1}^n = u_j^n + (u_x)_j^n \Delta x + \frac{(u_{xx})_j^n}{2}(\Delta x)^2 + \frac{(u_{xxx})_j^n}{3!}(\Delta x)^3 + O((\Delta x)^4) \tag{11}$$

where $\Delta x = x_{j+1} - x_j$ and also that

$$u_j^{n+1} = u_j^n + (u_t)_j^n \Delta t + \frac{(u_{tt})_j^n}{2}(\Delta t)^2 + \frac{(u_{ttt})_j^n}{3!}(\Delta t)^3 + O((\Delta t)^4) \tag{12}$$

where $\Delta t = t^{n+1} - t^n$, so it follows that

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = -a\frac{u_{j+1}^n - u_j^n}{\Delta x} + O(\Delta x) + O(\Delta t) \tag{13}$$

since the u_t and the au_x terms cancel by applying equation (10). This suggests the following numerical algorithm to simulate equation (10):

$$U_j^{n+1} = U_j^n - a\lambda(U_{j+1}^n - U_j^n) \tag{14}$$

where $\lambda = \Delta t/\Delta x$. This algorithm explicitly shows how to compute values for a new time t^{n+1} given information at the current time t^n. To compute the value at a fixed time t_0, the algorithm must be run through $t_0/\Delta t$ time steps. Since the error from each step is $\Delta t(O(\Delta x)+O(\Delta t))$, the total error between the computed solution and the actual solution at $t = t_0$ is $(O(\Delta x)+O(\Delta t))$, assuming that the errors from each time step can be added together (which is not the case when the algorithm is unstable as we soon discuss). Due to the form of the total error (if the algorithm is stable), we refer to the algorithm as being "first order accurate in x and first order accurate in t." If, instead of using the Taylor expansion in equation (11), we use

$$\begin{aligned} u_{j-1}^n \;=\; & u_j^n - (u_x)_j^n \Delta x + \frac{(u_{xx})_j^n}{2}(\Delta x)^2 \\ & - \frac{(u_{xxx})_j^n}{3!}(\Delta x)^3 + O((\Delta x)^4), \end{aligned} \tag{15}$$

we end up with a similar numerical scheme that also is first order accurate in both x and t:

$$U_j^{n+1} = U_j^n - a\lambda(U_j^n - U_{j-1}^n). \tag{16}$$

The scheme in equation (14) is called a *downwind scheme*; the scheme in equation (15) is called an *upwind scheme*. We could also subtract equation (15) from equation (11), which yields

$$u^n_{j+1} - u^n_{j-1} = 2(u_x)^n_j \Delta x + O((\Delta x)^3), \tag{17}$$

which implies that the numerical scheme

$$U^{n+1}_j = U^n_j - \frac{a}{2}\lambda(U^n_{j+1} - U^n_{j-1}) \tag{18}$$

is first order accurate in time and second order accurate in space since the error at a fixed time value is $(O((\Delta x)^2) + O(\Delta t))$. Obviously, one wishes to employ schemes with higher order errors since these schemes require less grid points (and therefore less computational time) to obtain a computed solution that is within a given error of the actual solution.

There is, however, more to numerical simulation of PDEs than just Taylor expansions and accuracy questions. Consider equation (10) where the domain of x is infinite and the initial condition $f(x)$ is specified:

$$u_t = -au_x \qquad \text{where } x \in (-\infty, \infty) \quad t \in [0, \infty) \tag{19a}$$

$$u(x, 0) = f(x). \tag{19b}$$

The unique solution to this equation is simply

$$u(x, t) = f(x - at). \tag{20}$$

In other words, the value of the solution is constant along each line $x = at + C$. These lines are called *characteristic lines*, and they occur when the domain of x is finite as well as infinite. Therefore, the value of the solution at a point depends only on which characteristic line crosses through that point. If we consider x to be the abscissa and t the ordinate, then each characteristic line has a positive slope of $1/a$. The constant a is often referred to as the *wave speed* since it specifies how fast a disturbance at a point x propagates down to other values of x. Now consider what happens when we apply the downwind scheme in equation (14) in an attempt to solve equation (10). The downwind scheme only uses the points (x_j, t^n) and (x_{j+1}, t^n) to estimate the value of the point (x_j, t^{n+1}), but we know from the characteristic lines that the value at this point only depends on the value of the solution at $x = x_j - a\Delta t$ if $t = t^n$, which is outside the range of the two points being used. This causes a computer simulation using the downwind scheme to go haywire and yield gibberish. One may wonder how can this happen since we know the scheme is accurate. The answer is that accuracy only considers *local* error, whereas the stability problem that is occuring in this case is a *global* issue.

The Taylor series analysis of error in determining accuracy *assumes* that the solution is correct at $t = t^n$, and then finds the error that is created when the solution at $t = t^{n+1}$ is approximated. However, if the error accrued in a step is magnified by each subsequent step, we can no longer just add the individual errors produced at each step; in fact, when the errors are magnified

by subsequent steps, we observe that the simulation is unstable, that is, the global error quickly exceeds processing ability. When a scheme attempts to approximate the solution at a point (x_j, t^{n+1}) whose characteristic line crosses the $t = t^n$ line outside the range of points used by the scheme, then the scheme, no matter how accurate, will always magnify errors in each step and therefore be unstable. In other words, the numerical range of dependence must always contain the theoretical range of dependence of the solution or instablity will occur. This principle is called the Courant–Friedrichs–Lewy (CFL) condition (sometimes referred to simply as the Courant condition) [7]. This guarantees that the downwind scheme will always be unstable and that the upwind scheme can only be stable if $a\lambda \leq 1$. Even if a scheme satisfies the CFL condition, it may still be unstable, but we do not analyze any schemes that possess this flaw in this paper.

Before analyzing the FPU nonlinear wave equations, we wish to apply the preceding analysis to the linear wave equation, $u_{tt} = c^2 u_{xx}$, where we use c^2 to denote the constant a_1/ρ for reasons that will soon be apparent. If we define $\mathbf{V} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$ where $v_1 = u_t$ and $v_2 = u_x$ then we have that the wave equation is satisfied, $(v_1)_t = c^2 (v_2)_x$, and also that the mixed partials are equal, $(v_1)_x = (v_2)_t$. This allows us to write the wave equation as the following first order partial differential system:

$$\mathbf{V}_t = \mathbf{A}\mathbf{V}_x \qquad \text{where } \mathbf{A} = \begin{bmatrix} 0 & c^2 \\ 1 & 0 \end{bmatrix}. \tag{21}$$

Equation (21), of course, is just the matrix form of the travelling wave equation from equation (10), and its solution contains many of the same properties as equation (10). As opposed to the solution depending upon one characteristic line, the solution $\mathbf{V}(x,t)$ now depends on two characteristic lines with wave speeds c and $-c$, the eigenvalues of $\mathbf{A}$. Therefore the slopes of the two characteristic lines are the reciprocals of the wave speeds, $1/c$ and $-1/c$. As before, we must satisfy the CFL stability condition, so we now require that $c^2\lambda^2 \leq 1$.

All that remains is to find a sufficiently accurate scheme. To obtain an approximation for u_{xx} we add equation (11) to equation (15), yielding

$$u^n_{j+1} + u^n_{j-1} = 2u^n_j + (u_{xx})^n_j (\Delta x)^2 + O((\Delta x)^4), \tag{22}$$

which we can combine with the analogous expression from the Taylor expansions with respect to time to yield the numerical scheme

$$\frac{U^{n+1}_j - 2U^n_j + U^{n-1}_j}{(\Delta t)^2} = c^2 \frac{U^n_{j+1} - 2U^n_j + U^n_{j-1}}{(\Delta x)^2}, \tag{23a}$$

which we can reexpress as

$$U^{n+1}_j = 2U^n_j - U^{n-1}_j + c^2\lambda^2 (U^n_{j+1} - 2U^n_j + U^n_{j-1}). \tag{23b}$$

Note that this scheme is second order accurate with respect to both time and space.

We are now ready to consider the nonlinear FPU equations. We begin with the quadratic FPU equation

$$u_{tt} = \alpha u_{xx} + \beta u_x u_{xx} \tag{24}$$

where $\alpha = a_1/\rho$ and $\beta = 2a_2/\rho$. (Note that $\alpha > 0$ since a_1, like the spring constant k, is always positive in any realistic spring model.) As with the linear wave equation we can rewrite equation (24) in vector form, but now the matrix $\mathbf{A}$ depends upon the value of v_2:

$$\mathbf{V}_t = \mathbf{A}\mathbf{V}_x \qquad \text{where } \mathbf{A} = \begin{bmatrix} 0 & \alpha + \beta v_2 \\ 1 & 0 \end{bmatrix}. \tag{25}$$

The wave speeds of the two characteristic lines are still the eigenvalues of $\mathbf{A}$, which are $\pm\sqrt{\alpha + \beta v_2}$. We require $\alpha + \beta v_2$ to be positive since otherwise the wave speeds would be imaginary and the PDE would no longer be hyperbolic, which leads to equations beyond the analysis here and causes the scheme to become unstable. Since the speeds are functions of v_2, that means the speeds vary, which implies that the characteristic lines are now curves with slopes that vary with time as opposed to being straight lines as is the case for the linear wave equation. This leads to a stability problem as the CFL condition now requires that $(\alpha + \beta v_2)\lambda^2 \leq 1$. If β is small compared to α, then $\alpha + \beta v_2$ will be positive and the variations in v_2 may not be severe enough to violate the CFL condition, but if the nonlinearity becomes more significant, which means the value of β becomes larger, eventually the CFL condition will be violated (or $\alpha + \beta v_2$ will become nonpositive) and the scheme will become unstable. (In the experiments in section 5, we set $\alpha = 1$ and use values of λ that are slightly smaller than 1 which causes the CFL condition to be violated before $\alpha + \beta v_2$ becomes nonpositive.) Since by definition $v_2 = u_x$, we see that if u_x gets large not only does the numerical scheme become unstable, but also the underlying differential equation becomes less valid since higher order terms in the Taylor approximation given in equation (4) start to become significant.

The analysis for the cubic FPU equations is similar. We now have that

$$u_{tt} = \alpha u_{xx} + \gamma (u_x)^2 u_{xx} \tag{26}$$

where $\gamma = 3a_3/\rho$, which can be expressed in matrix form as

$$\mathbf{V}_t = \mathbf{A}\mathbf{V}_x \qquad \text{where } \mathbf{A} = \begin{bmatrix} 0 & \alpha + \gamma (v_2)^2 \\ 1 & 0 \end{bmatrix}. \tag{27}$$

The corresponding CFL condition is $(\alpha + \gamma (v_2)^2)\lambda^2 \leq 1$. Note that if γ is positive, the wave speeds will be real and we will always have a hyperbolic PDE.

There are numerous accurate schemes that one can apply in the region where the algorithm is stable. To simulate the quadratic FPU scheme (equation (8)), we can combine equation (17) with equation (23) to form

$$\frac{U_j^{n+1} - 2U_j^n + U_j^{n-1}}{(\Delta t)^2} = \left(\alpha + \beta \frac{U_{j+1}^n - U_{j-1}^n}{2\Delta x}\right) \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{(\Delta x)^2}, \tag{28}$$

which can be rewritten as a scheme which is second order accurate in time and space:

$$U_j^{n+1} = 2U_j^n - U_j^{n-1} + \lambda^2 \left(\alpha + \beta \frac{U_{j+1}^n - U_{j-1}^n}{2\Delta x} \right) (U_{j+1}^n - 2U_j^n + U_{j-1}^n). \quad (29)$$

Equation (28) is a generalization of the FPU quadratic scheme (equation (i)). After setting $\alpha/(\Delta x)^2 = 1$ and $\beta/\left(2(\Delta x)^3\right) = \alpha_1$, one can quickly establish that the right-hand sides of equations (28) and (i) are identical.

Similary, we can simulate the cubic FPU scheme (equation (9)), by again combining equation (17) with equation (23) to obtain the following second order accurate scheme:

$$U_j^{n+1} = 2U_j^n - U_j^{n-1} + \lambda^2 \left(\alpha + \gamma \frac{(U_{j+1}^n - U_{j-1}^n)^2}{4(\Delta x)^2} \right) (U_{j+1}^n - 2U_j^n + U_{j-1}^n). \quad (30)$$

The fact that the scheme in equation (30) is not the same as the cubic scheme (equation (ii)) does not pose a problem. Either equation (ii) or (30) can be used to simulate the cubic FPU equation as both schemes are second order accurate and will therefore produce essentially equally valid simulations. (The second order accuracy of equation (i) is quickly established using Taylor series.)

There is another phenomenon that is important in nonlinear hyperbolic PDEs. Because the characteristics are curves, characteristics within a family can intersect each other. (In the linear case, each family of characteristic lines has a constant slope: $1/c$ for one family, $-1/c$ for the other family.) When the characteristics intersect, a *shock*; that is, a discontinuity in the solution, occurs. While multiple weak solutions to the nonlinear equation are possible, one usually desires to simulate the unique observed solution that the phenomena being modeled exhibits. For most physical phenomena the solution of interest is the *Lax entropy solution* of the equation, which can be simulated using monotone schemes [8]. A particularly nice introduction to shock capturing monotone schemes is presented in [9].

We do not pursue these schemes here, however, as we are interested in other aspects of the transition to instability.

4. Continuous-valued cellular automata for wave equations

A CA is a manifold of computing cells which repeatedly update their internal states. The update process is characterized by parallelism, homogeneity, and locality. Parallelism means that all the cells are updated at the same time, in lock-step synchronization. Homogeneity says all cells use the same update rule; and locality says cells only gather information from their nearby neighbors.

Much of the mathematical theory of CAs focuses on the large-scale dynamical behaviors of classes of CAs; this involves looking at many different runs of CAs. As a practical matter, it used to be hard to get large CA computations to run rapidly, so mathematicians have mostly looked at CAs in

which the cells have only a small amount of information—for instance, the "Game of Life" CA uses only one bit of internal state per cell. There is, however, no intrinsic reason why CAs should not have state values characterized as one or even several continuous-valued numbers.

Looking at such continuous-valued CAs to some extent duplicates work which numerical analysts have already done under the name of finite difference method simulations. But approaching these numerical simulations from a CA standpoint has the following positive effects.

1. It incorporates existing CA simulation techniques to achieve a rapidly-running and interactive graphical display.
2. It encourages an artificial life outlook in which emergent simulation properties are examined.
3. It implements a genetic algorithm approach whereby a large space of simulation-parameter values can be efficiently explored.

The CA framework also provides a conceptual language for discussing the behavior of a simulation after instability sets in. Readers with access to a computer using Microsoft Windows may wish to download the CAPOW program [4] in order to assess this for themselves.

For our continuous-valued CAs we let each cell be a small data structure that contains two floating point numbers which we call U and V. These are to stand for the intensity and velocity, that is, U stands for u and V stands for u_t.

To carry out this computation we use three buffers, with each buffer being a linear array of cell data structures. The length of the buffers corresponds to the number of space cells being used (or particles being considered). In [2], 64 and 128 cells are used, but we typically use several hundred, up to the horizontal pixel-width of the active screen resolution. At each step of the computation, one buffer can be thought of as holding the current cell values, with the other buffers holding the "old" cell values and the "new" cell values. The new buffer is updated on the basis of the values found in the current buffer and the old buffer. Rather than copying values from buffer to buffer, after each update, the names of the buffers are cycled: the new buffer becomes the current buffer, the current buffer becomes the old buffer, and the old buffer becomes the new buffer.

One danger in such continuous-valued simulations is that some of the U or V values may run away to very large or small values which produce a floating point overflow. To avoid this, we add a "clamping" step to the update process. Some maximum and minimum allowable values of U and V are selected in advance, and whenever a new U or V is computed, it is clamped to lie between the appropriate minimum and maximum; that is, if a value is above the allowed maximum we set it equal to the maximum, and if it is below the allowed minimum we set it equal to the minimum. Clamping is of course a nonphysical process, so only a simulation in which no clamping takes place can be regarded as modeling a PDE.

The values of the U or the V variables in the cells of the current buffer are displayed in two ways: as a graph, and as a space-time diagram. In each case we scale the allowable minimum to maximum range into some small discrete range of possible values. To show a graph, we use the scaled values to measure the vertical displacement of the pixel intended to represent the cell value in question. To represent the cell value as part of a space-time diagram, we use the scaled value to ascertain a color or gray-scale palette index for the pixel representing the cell upon a horizontal line corresponding to that instant of time.

For a given cell at some time j and space position n, U_j^n and V_j^n stand for the respective U and V values of the cell. U_{j-1}^n and U_{j+1}^n stand for the U values of the left and right neighbors of the cell, while U_j^{n-1} and U_j^{n+1} stand for the old and new values of the U field of the cell. V_j^{n+1} stands for the new value of the V field of the cell.

If we track the V values as well as the U values, the scheme corresponding to equation (23b) is

$$U_j^{n+1} = 2U_j^n - U_j^{n-1} + \left(\frac{c^2\Delta t^2}{\Delta x^2}\right)(U_{j+1}^n - 2U_j^n + U_{j-1}^n) \tag{31a}$$

$$V_j^{n+1} = \frac{(U_j^{n+1} - U_j^n)}{\Delta t}. \tag{31b}$$

Equation (31a) takes a particularly simple form if c is 1 and

$$\Delta x = \Delta t;$$

that is, it becomes

$$U_j^{n+1} = -U_j^{n-1} + U_{j-1}^n + U_{j+1}^n. \tag{31a*}$$

It is typical of the power and economy of CA that such a tiny equation contains the behavior of waves!

One can also think of the scheme in equation (31) in an alternate (but equivalent) way which is perhaps more intuitively obvious: update the velocity by adding Δt times the acceleration, then update the intensity by adding Δt times the velocity, that is,

$$V_j^{n+1} = V_j^n + (\Delta t)c^2\left(\frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{\Delta x^2}\right) \tag{32a}$$

$$U_j^{n+1} = U_j^n + (\Delta t)V_j^{n+1}. \tag{32b}$$

A conceptual advantage of the scheme in equation (32) is that it suggests how to expand the scheme to different kinds of accelerating forces. In general, if F is a numerical representation of the acceleration caused by a force, then this scheme implies that appropriate schemes will be of the form

$$V_j^{n+1} = V_j^n + (\Delta t)F \tag{33a}$$

$$U_j^{n+1} = U_j^n + (\Delta t)V_j^{n+1}. \tag{33b}$$

In computational practice, the scheme in equation (31) works better than that in equation (32) because it is more "balanced." That is, we generally are going to be working with small values of Δt and Δx which are similar in size. In equation (31a), we use them in the form $\Delta t^2/\Delta x^2$, but in equation (32a) we use the form $\Delta t/\Delta x^2$. Typically the first of these numbers will be near 1, but the second will be quite large. Multiplying by large numbers reduces the computational accuracy of a scheme.

As was mentioned above, unless sufficient care is taken, runaway values of U and V can occur. We discussed how to use a clamping technique to prevent these runaway values from crashing the simulation, but for any physically meaningful simulation, runaway values should not arise in the first place. An analysis of our numerical methods for the preceding algorithms reveals that runaway values are the result of either of the following.

1. Instability caused by choosing the space and time steps so as to violate the CFL condition.
2. Inaccuracy caused by seeding the CA with a clearly discontinuous function.

Regarding the first issue, we recall the CFL stability condition from section 2:

$$\frac{c^2\Delta t^2}{\Delta x^2} \leq 1.$$

This condition has implications on the time and space scaling of the model. Since the values of U_j^{n+1} depend only on information in the cell U_j^n, its two nearest neighbors U_{j-1}^n and U_{j+1}^n, and its past state U_j^{n-1}. The fastest speed at which information can propagate is one space cell per one time update. So in terms of a cell metric, the maximum wave speed c is always 1. In other metrics we can define the maximum speed c to take any value as long as we choose an appropriate scale for how distance is measured on the x and t axes. If a horizontal screen inch represents X spatial units, then a vertical screen inch will represent X/c time units. Thus if we want c to be the speed of light (roughly 3×10^{10} cm/sec), and we want the distance scale to be natural, with one horizontal screen centimeter representing one centimeter of x distance, then we must say that one vertical screen centimeter represents no more than (roughly) 0.33×10^{-10} seconds.

Regarding the second cause of runaway values, note that if you look at equation (31a*), you see that space and time behave like a red and black checkerboard of odd and even cells. That is, the values in the "black" cells depend only on the values in the other "black" cells, while the "red" cells depend only on the other "red" cells. This means that it is entirely possible for completely different patterns to be evolving in the odd and the even space-time cells. In the case where $c^2\Delta t^2/\Delta x^2$ is strictly less than 1.0, there will be some averaging of information between the black and the red cells,

but even then the disassociation of the odd-cell and even-cell simulations can make itself evident.[1] Attempts to recover a solution by blending or averaging the values from the two patterns is unwarranted and, unsurprisingly, leads to bizarre, nonphysical behavior.

When you seed this simulation with a discontinuous initial profile, then the numerical scheme does not simulate the underlying wave equation, since the Taylor series expansion for u used to derive the computational scheme in section 3 is, of course, not valid at discontinuities. If the CA is seeded discontinuously, it is entirely likely that the odd and even cells will behave differently in the resulting pattern. In order to obtain a range of continuous results we can use initial shapes for $u(x,0)$ such as a simple sine wave, a Fourier sum of several sine waves, or we can start with any random smooth pattern. If a simple discontinuity is introduced into one of our simulations, one sees the expected "Gibbs ringing" phenomenon whereby high frequency wave components form around the discontinuity. When the CFL quotient is near unity, the simulation becomes very sensitive to added discontinuities, as will be discussed in the next section.

For the quadratic nonlinear wave equation, we use the scheme in equation (29), and set $\alpha = 1$, so that we have

$$\begin{aligned} U_j^{n+1} &= -U_j^{n-1} + 2U_j^n + \left(\frac{\Delta t^2}{\Delta x^2}\right) \\ &\times \left(\left(U_{j+1}^n - 2U_j^n + U_{j-1}^n\right) + k_1\left(\left(U_{j+1}^n - U_j^n\right)^2 - \left(U_j^n - U_{j-1}^n\right)^2\right)\right) \end{aligned} \quad (34)$$

where $k_1 \equiv \beta/2\Delta x$. In the case of the cubic nonlinear wave equation, equation (30) gives the scheme:

$$\begin{aligned} U_j^{n+1} &= -U_j^{n-1} + 2U_j^n + \left(\frac{\Delta t^2}{\Delta x^2}\right) \\ &\times \left(\left(U_{j+1}^n - 2U_j^n + U_{j-1}^n\right) + k_2\left(\left(U_{j+1}^n - U_j^n\right)^3 - \left(U_j^n - U_{j-1}^n\right)^3\right)\right) \end{aligned} \quad (35)$$

where $k_2 \equiv \gamma/4\Delta x^2$.

5. Experiments

In this section we discuss several experiments performed with the CAPOW software. The illustrations in this section use a gray-scale palette of the form shown in Figure 1 to represent the intensity, or u value, of a cell. The scale consists of a series of 17 bands that alternately shade from black to white to black *etcetera* as the value of u increases.

In the CAPOW experiments discussed here, we look at one-dimensional CA. The arrays are about two hundred cells wide, and periodic boundary conditions are used, wrapping the left end to match the right end. In order

[1] In [10], this checkerboard effect is analyzed for simulations of Burgers's nonlinear wave equation.

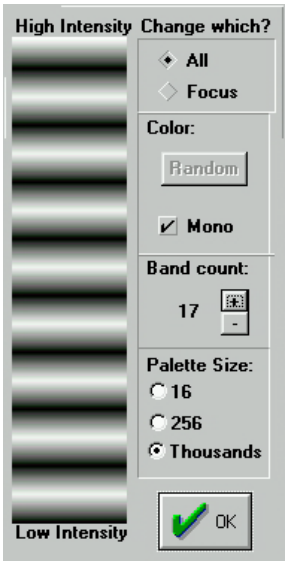

Figure 1: The CAPOW color dialog box, showing the color map used in the subsequent figures.

to view the activity in a given CA, we show two different views of the CA, as in Figure 2. The lower part of the figure shows an instantaneous image of the CA in the form of a graph of cell intensity against cell position. The upper part of the figure shows a space-time diagram of the CAs history of states, with the time axis running from top to bottom. In the space-time diagram, each instantaneous state of the CA is represented as a single horizontal line, with the pixel colors being based on the intensity at the corresponding cell position. The earlier CA states correspond to horizontal lines higher up in the image, and the bottom line of the image corresponds to the CA state currently being displayed as a graph. Figure 2 shows the stable wave equation simulation scheme in equation (31a), seeded by a randomly chosen Fourier sum of several sine waves. The value of Δt used is 0.02, the value of Δx used is 0.024, and the clamping range is -150.0 to 150.0. Wave equation simulations are not particularly sensitive to these values, as long as one is sure that Δx is greater than Δt, and that the initial seed values chosen lie within the clamping range.

It is possible to show several CAs at once with CAPOW. Figure 3 shows the results of seeding the wave simulations with; clockwise from upper left, a single tent-peak, a sine wave, a Fourier sum, and random values. The sine wave seed yields a space-time pattern that looks a bit like a box of sushi. The random seed does not settle down for the linear wave scheme.

We previously discussed that the space step must be greater than or equal to the time step. Experiment shows the simulation to be more robust if it is

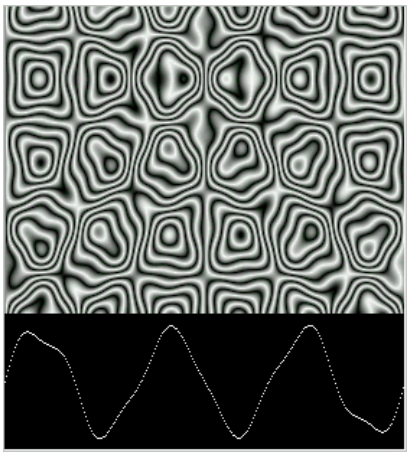

Figure 2: The wave equation simulation, seeded by a Fourier sum of sine waves.

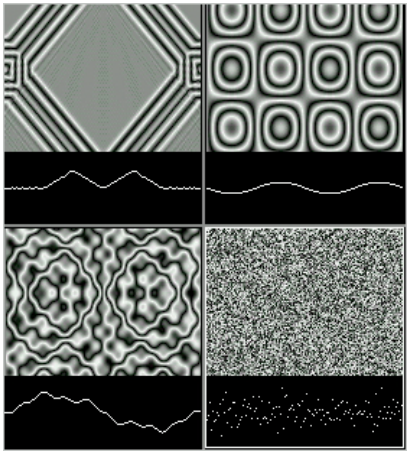

Figure 3: The wave equation seeded by a spike, wave, Fourier sum, and random noise.

strictly greater. As mentioned before, if we do set the space step equal to the time step, the scheme reduces to the scheme in equation (31a*). In Figure 4 we show how this rule can lead to an alternating pattern.

Figure 4 was arrived at by seeding the CA with a sine wave, and then introducing a step-discontinuity at a small interval overlapping the left and right boundaries (recall that we use periodic boundary conditions). More precisely, we patch in a small interval of values taken from a sine wave of a different phase.

When the perturbation first happens, we see a pattern similar to the middle of the space-time picture in Figure 4, with the disturbance moving

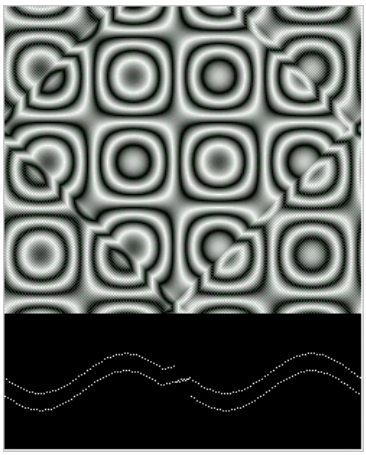

Figure 4: An unstable wave, with the odd space-time cells out of phase with the even space-time cells.

inward from both the left and the right ends. As the CA continues to evolve, the two perturbation waves repeatedly pass through each other at the center of the CA, proceed out to the ends, wrap around to move back towards the center and so on. Figure 4 shows the CA after it has already undergone several of these cycles since the initial perturbation event. If left alone, the CA would repeat this pattern indefinitely.

When we refer to the CA in Figure 4 as having an "alternating" pattern of instability, we mean that the odd-numbered and the even-numbered cells are out of step with each other. If the Δx were strictly greater than the Δt in Figure 4, the image would change in two ways: the upper left and lower right rows of dots would be vertically translated so as to lie evenly with the rest of the curve; and the spacetime diagram part would be less fuzzy. The perpetually criss-crossing perturbation pattern would remain.

We now turn to the quadratic nonlinear wave based on the scheme equation (34). This simulation very easily becomes unstable. If we set the nonlinearity to 0.385 and seed it with a Fourier seed, we see a result similar to that in Figure 5. The simulation breaks down and becomes unstable at several points, and the breakdowns spread at the speed of the wave in the medium. In order to avoid floating point overflow, we clamp our maximum and minimum wave values to be within a certain range (-12.0 to 12.0 in the case of Figure 5). A result of this is that the cell values spend a lot of time at the maximum and/or minimum values. Because these special values are visited so often, certain intermediate values calculated from these special values are visited often as well. This has the effect of making the continuous CA behave somewhat like a discrete-valued CA with a limited number of states. Note in Figure 5 how the unstable zone shows behavior similar to discrete-valued CA. This is an example of the "pseudodiscrete CA" mentioned in the

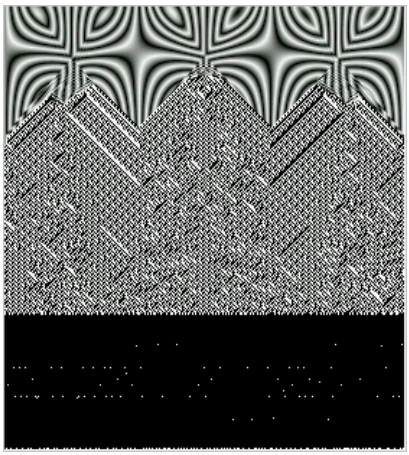

Figure 5: Quadratic nonlinear wave breakdown, nonlinearity 0.385.

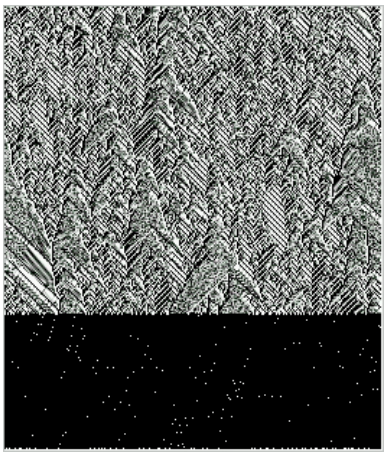

Figure 6: Clamped quadratic wave seeded with random values, after 500 generations.

introduction. In terms of Wolfram classification, this CA seems to be of the glider-sustaining type 4, as evidenced by the zig-zagging patterns which send off diagonal lines.

For the rest of our quadratic nonlinear wave experiments we use these parameters: Δx is 0.109, Δt is 0.090, the clamping range for U is -12.0 to 12.0, and the k_1 "quadratic nonlinearity" value in the scheme in equation (35) is 0.076.

For our first experiment with these settings, we seed the CA with a random selection of values, (Figure 6). Due to instability, the CA values get repeatedly clamped, producing an appearance like a discrete-valued type 4

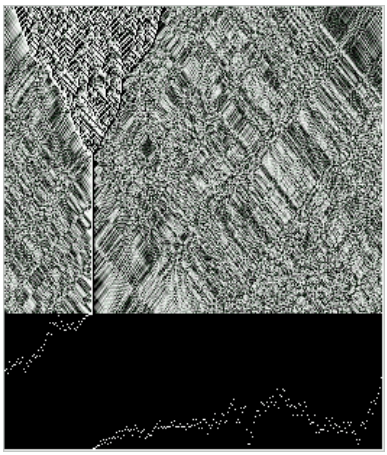

Figure 7: Clamped quadratic wave seeded with random values, after 1500 generations.

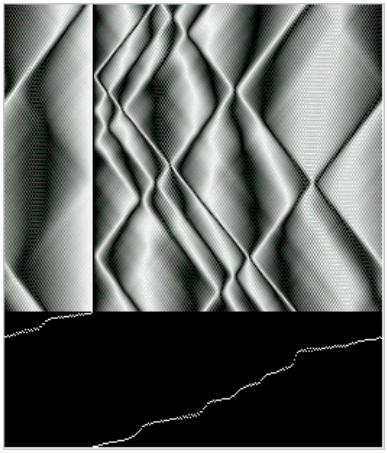

Figure 8: Clamped quadratic wave seeded with random values, after 5000 generations.

CA, as shown in Figure 7. For this particular value of the nonlinearity parameter, the repeated clampings of the CA happen to bring groups of cells into approximate continuity, and the CA then evolves into a smooth pattern marked by step-shaped waves and a single maximal discontinuity as shown in Figure 8. For lower values of the nonlinearity parameter, there is not enough clamping to impose this artificial continuity, while for higher values of the nonlinearity parameter, the imposed continuity takes over to make the CA cell values constant across the entire space. From a CA standpoint these patterns are interesting, although from a physical standpoint they are simply artifacts of a broken simulation.

In [2], experiments are carried out by seeding schemes based on the quadratic and cubic forms of the nonlinear wave equation with a sine wave, and then repeatedly examining the power spectrum (i.e., Fourier modes) of the shape assumed by the "nonlinear string" as more and more generations go by.

In his autobiography, Ulam gave a nice summary of the surprising results of some of their experiments.

> "It was the consideration of an elastic string with two fixed ends, subject not only to the usual elastic force of strain proportional to strain, but having, in addition, a physically correct small nonlinear term. The question was to find out how this nonlinearity after very many periods of vibration would gradually alter the well-known periodic behavior... and how, we thought, the entire motion would ultimately thermalize, imitating perhaps the behavior of fluids which are initially laminar and become more and more turbulent.... The results were entirely different qualitatively from what even Fermi, with his great knowledge of wave motions, had expected. The original objective had been to see at what rate the energy of the string, initially put into a single sine wave... would gradually develop higher tones with the harmonics, and how the shape would finally become 'a mess'.... To our surprise the string started playing a game of musical chairs, only between several low notes, and... came back almost exactly to its original sinusoidal shape." [11], pages 226–227.

CAPOW confirms the behavior Ulam observed for both the quadratic and the cubic forms of the nonlinear term.

In Figures 9 through 16, we look at the long-term evolution of a nonlinear quadratic CA seeded with a sine wave. The time-step counts given are approximate, the behavior does not change rapidly enough for greater precision to be required.[2] Because the quadratic scheme is not spatially symmetric, the sushi wave patterns become pointed. When observed running, the moving graph of the CA produces an illusion of rotation. The rotation effect centers on the vertices where the longer pieces of the curve connect the shorter pieces, as in Figures 10 and 11. These vertices, which correspond to the maxima and minima of the original sine wave seed, seem to rotate clockwise around the points at the centers of the segments, which correspond to the initial seed values of 0. As the vertices approach and recede from each other, the connecting segments look as if they are being squashed and stretched, with secondary waves being sent off from the "rotating" tips, as in Figure 12.

As the evolution continues, higher frequency terms are produced by the perturbations caused by the stretching and shrinking of the wave, as in Figures 13, 14, and 15. The pattern remains periodic. After about one hundred

[2] A Pentium-class computer running CAPOW for a single CA about two hundred cells wide does about 100 complete CA updates per second, and can perform one hundred thousand steps in about 15 minutes.

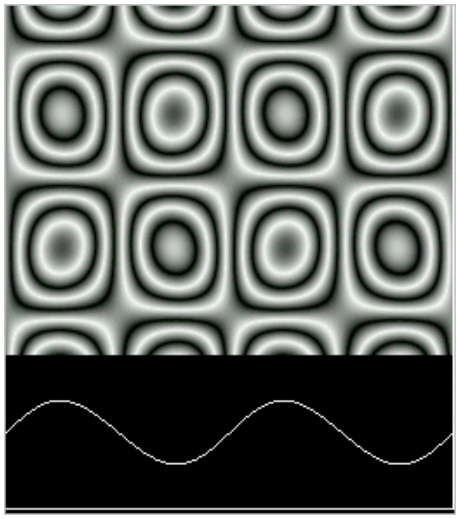

Figure 9: Quadratic wave seeded with sine wave, after 100 steps.

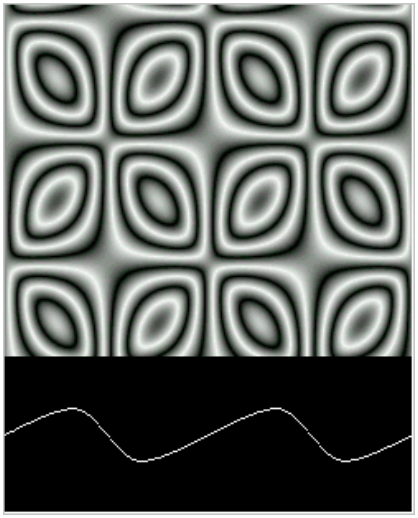

Figure 10: Quadratic wave after 1500 steps.

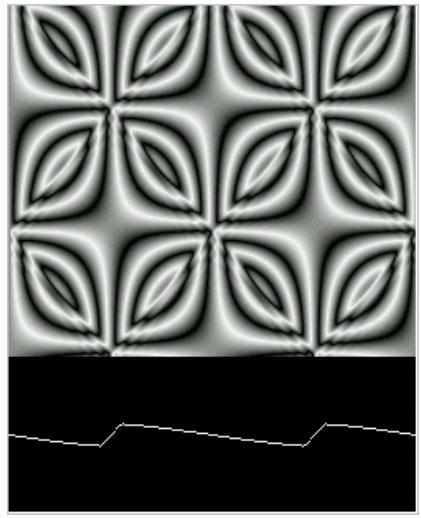

Figure 11: Quadratic wave after 3,000 steps.

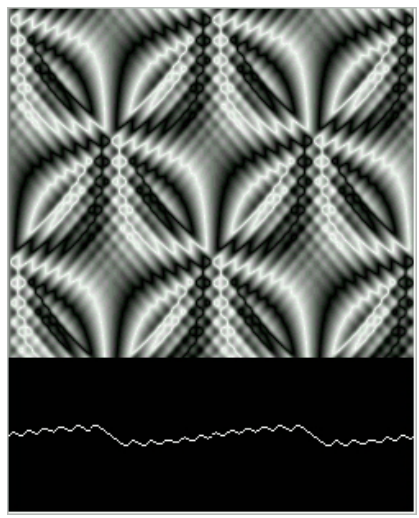

Figure 12: Quadratic wave after 6000 steps.

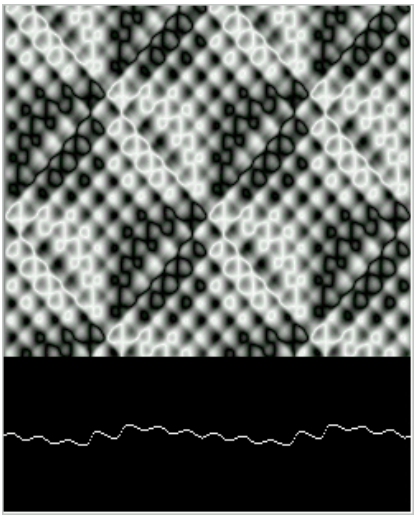

Figure 13: Quadratic wave after 12000 steps.

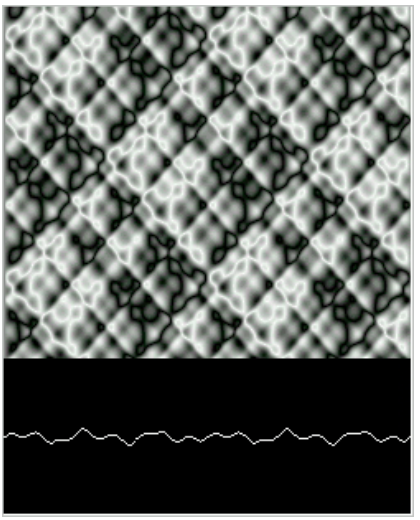

Figure 14: Quadratic wave after 30000 steps.

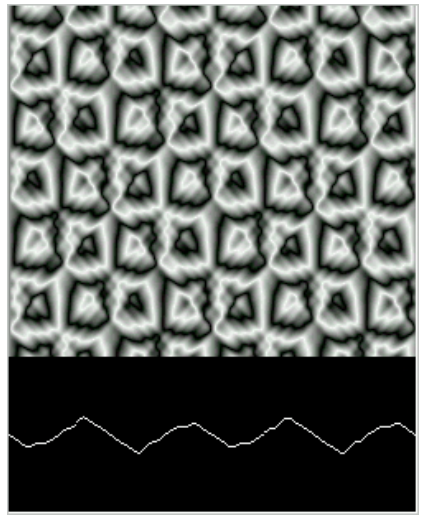

Figure 15: Quadratic wave after 42000 steps.

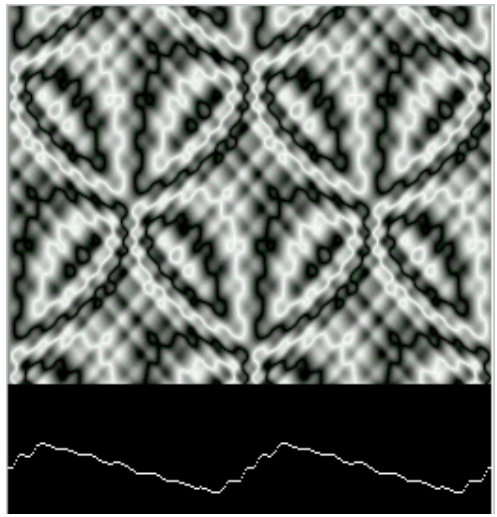

Figure 16: Quadratic wave after 100000 steps.

thousand iterations a pattern similar to the starting pattern seems briefly to reemerge, shortly after the pattern shown in Figure 16. The reemergence of the seed wave happens even more clearly for the cubic form than for the quadratic form of the nonlinear wave simulation.

For our cubic nonlinear wave experiments we use these parameters: Δx is 0.109, Δt is 0.090, the clamping range for U is -1.0 to 1.0, and the k_2 "cubic nonlinearity" value in the scheme in equation (36) is 10.720.

Because of the left-right symmetry of this scheme, the cubic scheme is less apt to break down than is the quadratic scheme, so we can set the nonlinearity to a fairly large value. On the other hand, cubing a number greater than one can greatly amplify it, which is why we restrict our intensities to the range -1.0 to 1.0.

We seed the cubic wave with a sine wave as shown in Figure 17. The nonlinear effects of the scheme show themselves more slowly than in the quadratic case. The first effect visible is a bending of the sushi into badge shapes, with the wave becoming pointy instead of smooth as in Figure 18. Soon secondary points occur, with the peaks of the wave folding down while the sides are still moving up, as in Figure 19. Increasingly complicated, but still periodic, patterns occur, as in Figure 20.

Finally a kind of maximum level of "thermalization" is obtained with most of the energy of the wave in low amplitude high frequency oscillations, as shown in Figure 21, when the return begins. Slowly the evolution of the wave retraces itself, eventually getting back to something very much like the initial condition, as in Figures 22 and 23.

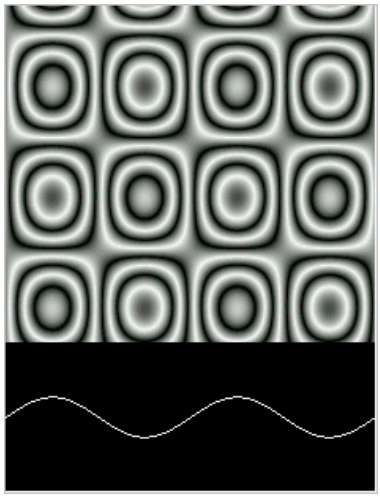

Figure 17: Cubic wave seeded with sine wave, after 100 steps.

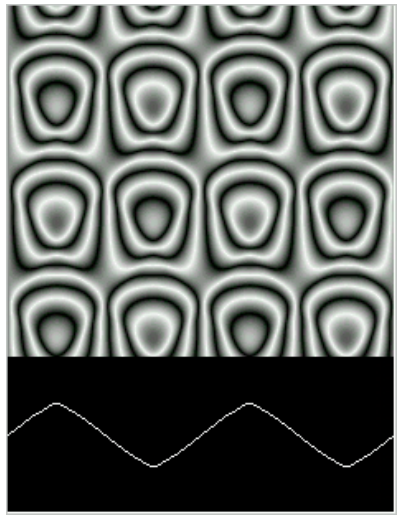

Figure 18: Cubic wave after 6000 steps.

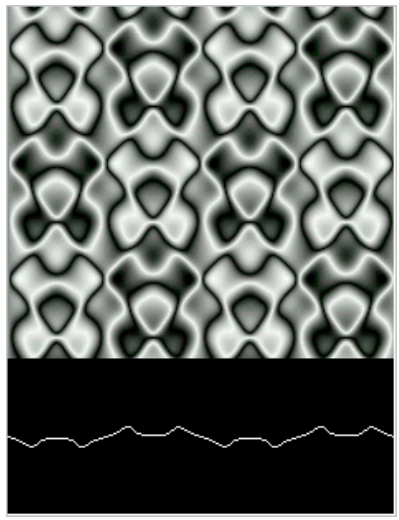

Figure 19: Cubic wave after 24000 steps.

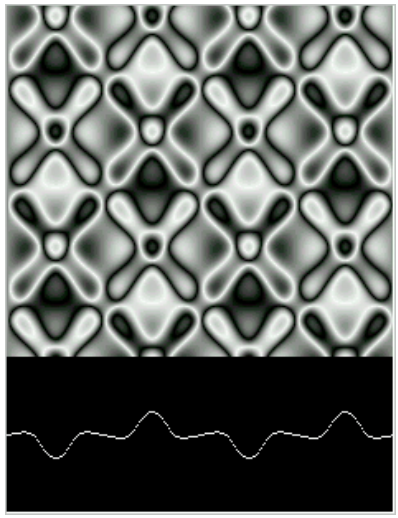

Figure 20: Cubic wave after 36000 steps.

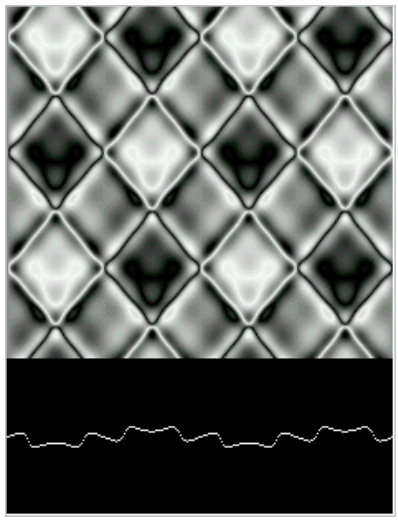

Figure 21: Cubic wave after 48000 steps.

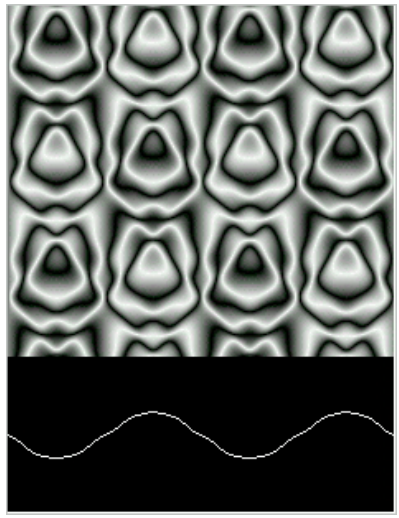

Figure 22: Cubic wave after 90000 steps.

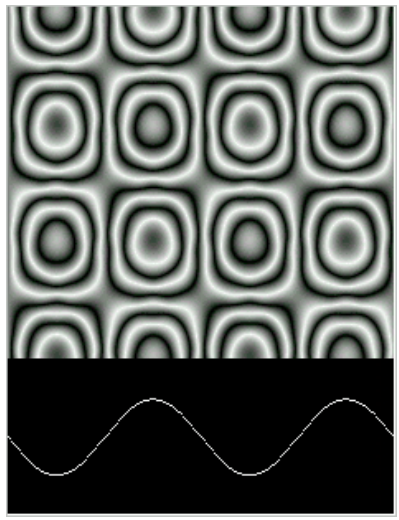

Figure 23: Cubic wave after 120000 steps.

In the quadratic case, this quasiperiodic behavior for the differential equation being simulated (equation (8)) is explained in [12]. Recently, in [13], Sobolev estimates are used to expand the results in [12] to explain this quasiperiodic behavior for the discrete simulations presented here. Proof of the quasiperiodicity in the cubic case has not yet been established, though it seems likely to have an explanation that parallels the quadratic case.

Acknowledgment

Rudy Rucker gratefully acknowledges support from the Electric Power Research Institute (EPRI) of Palo Alto, CA, under a contract entitled "Evolving Complex Cellular Automata for Power Grid Simulation," which ran from Spring 1994 through Fall 1997 and funded the development of the CAPOW shareware at the Center for Applied Math and Computer Science (CAMCOS) of the Department of Mathematics and Computer Science at SJSU by Rucker and his students.

References

[1] S. Ulam, "Random Processes and Transformation," in *S. Ulam, Sets, Numbers and Universes* (MIT Press, Cambridge, 1974).

[2] E. Fermi, J. Pasta, and S. Ulam, "Studies of Nonlinear Problems," originally in *Los Alamos Report LA-1940*, 1955; later in [1].

[3] S. Ulam, *Collected Papers of Enrico Fermi, volume 2*, (University of Chicago Press, 1965).

[4] R. Rucker, *et. al.*, "CAPOW! Version 5.0," software available for free download at `http://www.mathcs.sjsu.edu/capow`, 1997.

[5] S. Wolfram, "Universality and Complexity in Cellular Automata," *Physica D*, **10** (1984) 1–35.

[6] R. B. Bird *et al.*, *Dynamics of Polymeric Liquids, volume 2: Kinetic Theory* (John Wiley and Sons, 1987).

[7] R. Courant, K. O. Friedrichs, and H. Lewy, "Uber die Partiellen Differenzengleichungen der Mathematisches Physik," *Math. Ann.*, **100** (1928) 32–74.

[8] M. G. Crandall and A. Majda, "Monotone Difference Approximations for Scalar Conservation Laws," *Mathematics of Computation*, **34** (1980) 1–21.

[9] R. J. LeVeque, "Numerical Methods for Conservation Laws," (Birkhauser-Verlag, 1992).

[10] B. T. Hayes, *Binary Modulated Oscillations in a Semi-discrete Version of Burgers Equation* (based on Ph.D. Thesis, Courant Institute of New York University, 1994).

[11] S. Ulam, *Adventures of a Mathematician* (University of California Press, 1976).

[12] V. E. Zakharov, "On Stochastization of One-dimensional Chains of Nonlinear Oscillators," *Sov. Phys. JETP*, **38** (1974) 108–110.

[13] J. F. Bukowski, *The Boussinesq Limit of the Fermi–Pasta–Ulam Equation* (Ph.D. thesis, Brown University, 1997).