1 Introduction

Physical modelling is the process of finding a mathematical description that is consistent with observations of the physical world. Many models, from the heat equation to the Schroedinger equation, are formulated in the continuous language of differential equations. Space and time are thought of as a continuum, hence the models potentially specify detail down to arbitrarily fine scales.

Discrete physical models are an attractive alternative to continuous models such as partial differential equations. In discrete models, space is treated as a lattice, and time is discrete. Physical processes are modelled by rules that typically depend on a small number of nearby locations. From a theoretical standpoint, such models have the advantage that they do not have infinitely many locations per unit volume. From a practical standpoint, they correspond to the discrete structure of digital computing machines, and this makes them natural for simulation.

Formulations in which space and time are discrete are widely used. Most often, this is done in order to approximate continuous models: space and time are partitioned in order to integrate a model in the form of a differential equation on a computer.

By contrast, some models have been proposed as alternatives to, rather than an approximations of, differential equations.[6] Cellular automata (CA's)[3] and lattice gases [5] are widely-used classes of discrete models in which space is modelled as a regular lattice, with a state associated with each lattice site. Periodically, all sites on the lattice simultaneously update their states. The rule that determines the new state is local, depending only on nearby sites, and is a function of the state of the neighboring sites in the previous time step. CA's and lattice gases have been used with success to study many different phenomena such as fluid dynamics, polymers, and molecular dynamics.

There are two main features that all commonly-used discrete models share. First, space is a regular lattice. Update rules typically exploit this – for example, by using the fact that two neighbors are at "right angles" to, or "opposite," each other. Second, although the update rule is local in space, the updates for all sites are globally synchronized with each other. The rule is a function of the states from the previous time step, before any of the updates from the current time step have happened. These conditions are very convenient for simulation because they correspond to the structure of the machines used to simulate the models.

We might well ask, however, on the grounds of minimalism, whether global synchronization and crystalline geometry are inherent in any discrete formulation. Is it possible to do without these conditions and still have a useful physical model? Or are they somehow fundamental?

There are reasons to believe that a regular lattice is not fundamental. With this geometry, a particular set of directions are preferred over others, namely the axes of the lattice. Likewise, in most discrete models the update rule operates locally in space, but we might ask whether we can devise rules such that the update can happen independently of other sites in time as well, rather than happen as part of a single global update. Most discrete models depend critically on both the regularity of the lattice and on a universal clock.

These conditions need to be examined if we want to think of discrete models as physical models themselves rather than as approximations of other models. This thesis thus investigates the question of how minimal a useful discrete model can be.

1.1 Overview

This thesis will show that it is possible to formulate a useful discrete model that does not rely on synchrony or regular geometry. Section 2 makes more concrete what is meant by "local" in discrete models. We will introduce the constraints that capture what it means for updates to happen independently of each other in time. A key constraint is that each site has no information about where others are in relation to it (other than whether they are neighbors), or at what time they update. If a discrete model works with these constraints, it does not rely on synchronization or regular geometry.

Section 3 outlines a class of discrete models that satisfy these constraints. This Section then presents an example of a useful instance of this class of models, a model of wave phenomena. It is shown that all interaction can be reduced to transactions between pairs of sites.

Section 4 presents a method for simulating the asynchronous, parallel model on a sequential machine, and gives results of simulations of the discrete wave model. The model is shown to agree both qualitatively and quantitatively with partial differential equation models.

Finally, Section 5 compares physical models to models of computation. It is shown that the model we introduce corresponds to a different kind of highly parallel computing machine known as an amorphous computer.

2 Locality

Our goal is to make everything about the update rule as local as we can. We will now make this more concrete by expressing what it is that should be minimized. In a discrete model, a site interacts with others that are connected to it, and when it updates its state, its new state is a function $f(\mathbf{x}, \mathbf{x}_1..\mathbf{x}_n)$ of its own state and that of these other sites. We introduce the notion of a "box" that encapsulates everything that changes during an update, so that an update event is truly independent from all other updates. A box encompasses sites that update together, during the instant that they update. It is defined by the requirement that a change in the state of any site occurs completely within some box, and the update rule f can depend solely on the states of the sites within the same box. Our task thus becomes to make this box as small as possible. (Fig. 2)

Space is considered to be a graph, with sites being nodes and the possibility that two sites can interact being represented by an edge. Each site s has a state $Q(s)$ associated with it. For simplicity, we will assume that there is a fixed set of locally-connected clusters of sites, and that any box that occurs encloses one of these clusters. A cluster S is a subset $\{s_1 ... s_n\}$ of all sites which has the property that for any s_i,$s_j \in S$, there is a path from s_i to s_j that stays completely in S. The clusters must, of course, overlap if the sites are to interact with each other. A box is thus the event of a cluster changing its state: $Q(S)$ is replaced with $f(Q(S))$.

The dynamics can be viewed as an update rule f combined with a schedule of boxes. For simplicity, updates can be considered instantaneous, since boxes are all self-contained. However, there are many cases where it is undefined whether one box occurs before or after another. Therefore, this schedule only needs to be partially ordered. It is a *preorder*, a partial ordering that allows multiple distinct events to be simultaneous. Not all possible schedules are allowed, however. At least two conditions are necessary for two boxes to be independent of each other rather than be part of the same box:

- **Nondeterministic in time.** The occurrence of one may not be deterministically related to the occurrence of another in the schedule. For example, if we have two clusters A and B, we may not do things like specify that B must update immediately after A; for this to happen, the sites in $A \cup B$ would all be considered enclosed by a single box.

- **Homogeneous update.** Likewise, we require the update function f to be homogeneous in time, a function only of the states of the sites in the cluster: the same update function must be applied on each transaction. We do not allow a cluster to do a “no-op” (the identity function), for example, waiting until some other cluster has updated.

We do allow the specification of statistical properties – for example that the average time between transactions is the same for all sites, or that the average number of neighbors has some variance. The idea is to constrain the geometry and time in the model to be “spacelike” or “timelike”, while ensuring that the ordering of the boxes remains fundamentally nondeterministic and allowing the geometry to be irregular, so that synchronization and detailed knowledge of the geometry are not relied on.

2.1 Implications

An update potentially makes use of the state of every site in a cluster, and these depend on previous updates that involved those sites. Each site can be a member of several clusters, so for any set of clusters that overlap, the boxes that enclose them must be fully ordered in the schedule; they cannot be simultaneous.

We could imagine a site "remembering" the states of its neighbors by incorporating them into its own state. These states could then be used later without being enclosed by a box. However, as the schedule is nondeterministic, between the time a neighbor's state is remembered and the time it is used, that neighbor may have undergone other updates and changed its state. The remembered state would be out of date. Remembering states is therefore unreliable.

Note that models with global synchrony require that all sites have updated once before any one site updates again, and f is a function of the states from the previous time step, before any of the updates of the current time step have happened. So in synchronous models, the update rule itself may be local in space, but by this more stringent requirement that also takes time into account, the box is infinite in size to encompass all the sites during an update.

3 Very local models

We have set the task of minimizing the size of the box that encapsulates the update. According to this framework, the smallest conceivable box that still allows interaction between sites is one that encloses just two sites at a time. We define a new class of models with the following two properties: boxes encompass only pairs of sites (fig. 3), and every transaction is conservative. We will show that a useful model is possible even with the pairwise restriction. As a consequence, we will find that crystalline geometry is not necessary either. Such an update rule does not distinguish the neighbors from each other, in the way that (for example) a CA rule does (Fig. 3). It does not make use of the crystalline structure, and so can be applied to irregular lattices.

3.1 Incorporating conservation

Conservation laws are fundamental in physics, and so they should be a part of a physical model. Local conservation also useful in dealing with the constraint of asynchrony and having only pairwise interactions. Global conservation will be achieved by construction: a certain variable or set of variables are designated as conserved quantities, and we design rules so that every local update leaves the sum of each conserved quantity over all sites unchanged.

One can thus think of the update as a "transaction" between sites. Periodically, a given pair of neighbors exchanges a packet of the conserved quantity; the loss from one is equal to the gain of the other. Each member of the pair then goes on to transact with other sites.

3.2 A simple example: diffusion

A simple example of the above class of models is a diffusion model, representing for example the diffusion of heat. We will call the conserved quantity q (representing the substance diffusing). A transaction between members of a pair transfers some of this quantity. In diffusion, the flux between two sites is proportional to the difference between them, so an appropriate rule is:

$$\begin{aligned} \Delta q_i &= (q_j - q_i)\Delta t \\ \Delta q_j &= (q_i - q_j)\Delta t \qquad (1) \end{aligned}$$

$\Delta q_i + \Delta q_j = 0$ so the transaction is conservative. Figure 5 shows a time evolution of this model.

3.3 Waves

As a more interesting example, we will simulate the physical phenomenon of waves. Waves are a ubiquitous phenomenon in physics, appearing in a variety of different systems – electromagnetic radiation and sound waves being two widely different examples that can nevertheless be described by the same kind of model.

All waves involve an amplitude associated with each point in space that changes over time. The future evolution of these amplitudes cannot be determined only from knowing all the amplitudes at a given moment. Travelling waves have a direction associated with them, and it is not possible to tell what the direction is by looking at an instantaneous snapshot; therefore they are an inherently second-order phenomenon. For this reason, the state of each site will be two quantities.

3.3.1 Momentum conserving model

We choose momentum, p, as the conserved quantity. It will be shown that the system supports propagating waves in multiple dimensions, agreeing closely with the continuous wave equation, even when the sole possible interaction is two sites exchanging a packet of momentum. We will call the other quantity position, q (representing the amplitude of the wave).

In order to determine how much momentum two sites should exchange, we associate a potential with a difference in q's between neighbors. We can think of a site as a mass free to move in one dimension, connected to its neighbors by springs. (Fig. 4) Two neighbors exert a force (*i.e.*, change in momentum) on each other which depends on the displacement between them. Such a system is known to support propagating waves and to be approximated by the scalar wave equation $\frac{\partial^2 q}{\partial t^2} = c^2 \nabla^2 q$. Thus, two sites i and j exchange momentum as follows:

$$\Delta p_i = m \Delta v_i = m a_{ij} \Delta t = F_{ij} \Delta t = f(q_j - q_i) \Delta t$$

$$\Delta p_j = m \Delta v_j = m a_{ji} \Delta t = F_{ji} \Delta t = f(q_i - q_j) \Delta t \qquad (2)$$

where a_{ij} is the contribution to the acceleration of site i from site j. f gives the force as a function of the difference in amplitudes; the difference can be positive or negative. In order for the gain of one site to be equal to the loss of the other, we require f to be anti-symmetric ($f(-x) = -f(x)$), so that we have

$$\Delta p_i = -\Delta p_j \tag{3}$$

and momentum is conserved. We will normally use $f(x) = kx$, which corresponds to Hooke's Law.

Each site can now independently update its value of q using its new value for p:

$$\Delta q_i = v_i \Delta t = \frac{p_i}{m} \Delta t \tag{4}$$

This system supports waves that can exhibit all the phenomena of continuous waves (Fig. 9), including reflection (Fig. 10) and refraction (Fig. 11). It also agrees quantitatively with a solution of the wave equation started from the same initial conditions, as shown in section 4.

3.3.2 Energy and momentum conserving model

In the above model, a difference in amplitude $q_i - q_j$ between neighbors can be considered to have potential energy; and p, because it is directly related to a change in q, can be associated with kinetic energy. Some of the potential energy converted into kinetic energy, or vice versa, when an update happens. Since potentials only make sense with pairs of sites, energy cannot be described for a single site but only for a given pair. An obvious expression for the total energy of a pair $\{i, j\}$ of neighbors is

$$\frac{1}{2} k (q_i - q_j)^2 + \frac{p_i^2 + p_j^2}{2m}. \tag{5}$$

Designing a wave model that has a pairwise interaction rule that conserves this energy as well as momentum is more difficult than it may at first seem, for the following reason. Potential energy depends on a difference between two neighbors' amplitudes q, yet each site is potentially a member of many pairs, so changing q changes the potential energy in a way that depends on all neighbors simultaneously. So to conserve energy, it would seem necessary to know instantaneously the value of q of all the neighbors, and our box would have to enclose them as well. A site cannot avoid this problem by remembering the states of its neighbors: they may participate in other updates after this information is remembered but before it is used. The remembered state would therefore be out of date.

There is however a way to conserve energy with only pairwise interactions. We will abandon the idea of an absolute amplitude associated with each site, and instead associate a quantity with each pair of sites. Associated with the "bond" between two sites will be a quantity l_{ij}. This quantity alone will determine the potential energy, which is "stored" in the bond.[1] l_{ij}, instead of the relative position, will determine the potential and thus the momentum transfer between sites i and j. Its change is determined by the relative momenta of the two sites in the pair. It is now no longer possible to consider an absolute, scalar amplitude associated with each site such that the l_{ij}'s are the differences between these amplitudes. The reason is that the l_{ij}'s are decoupled from one another, unlike $q_i - q_j$.[2]

The update rule is somewhat more complicated in this case. To conserve energy, we use a leapfrog method [13] for the individual transaction. The leapfrog method advances each site a step at a time, but the position updates are a half step offset from the momentum updates:

$$p^i_{t+1} - p^i_t = -k\frac{l^{ij}_{t+1} + l^{ij}_t}{2}\Delta t \tag{6}$$

$$p^j_{t+1} - p^j_t = -k\frac{l^{ji}_{t+1} + l^{ji}_t}{2}\Delta t = k\frac{l^{ij}_{t+1} + l^{ij}_t}{2}\Delta t \tag{7}$$

---

[1] This is reminiscent of the method of [3] for Ising cellular automata, which conserves a quantity measured along bonds, rather than at sites.

[2] One way around this, however, is to consider the amplitudes to be no longer in a single dimension; amplitudes that are consistent with the l_{ij}'s being differences between them could then be found.

$$l^{ij}_{t+1} - l^{ij}_t = \frac{p^i_{t+1} + p^i_t}{2m}\Delta t - \frac{p^j_{t+1} + p^j_t}{2m}\Delta t \tag{8}$$

where the subscript represents time, and the superscript is the index of the site.[3]

Solving for the new values in terms of the old:

$$p^i_{t+1} = \frac{2mp^i_t + k\Delta t^2 p^j_t - 2km\Delta t l^{ij}_t}{2m + k\Delta t^2} \tag{9}$$

$$p^j_{t+1} = \frac{2mp^j_t + k\Delta t^2 p^i_t + 2km\Delta t l^{ij}_t}{2m + k\Delta t^2} \tag{10}$$

$$l^{ij}_{t+1} = \frac{2\Delta t(p^i_t - p^j_t) + l^{ij}_t(2m - k\Delta t^2)}{2m + k\Delta t^2} \tag{11}$$

We can write this a matrix T that transforms a vector of the old values of p^i, p^j, l^{ij} into the new ones:

$$\frac{1}{2m + k\Delta t^2}\begin{bmatrix} 2m & k\Delta t^2 & -2km\Delta t \\ k\Delta t^2 & 2m & 2km\Delta t \\ 2\Delta t & -2\Delta t & 2m - k\Delta t^2 \end{bmatrix} \tag{12}$$

The determinant of this matrix $detT = 1$, implying that volume in (p^i, p^j, l^{ij}) phase space is conserved [2] and hence the transformation conserves energy.

As shown in figure 7, this system also exhibits all wave phenomena, and agrees quantitatively with the partial differential equation model as shown in section 4.

4 Simulation

We will now show a particular realization of the above class of discrete models. We will choose a particular method of placement of sites in space, and a particular rule for the occurrence of the updates. These will be chosen not for maximum

---

[3] Various other methods were tried: choosing some Δp based on l^{ij} and solving for a new l^{ij} that conserves energy; and vice versa. These were found to have no solution for Δp for certain values of l^{ij}.

efficiency; we will merely choose some conditions that satisfy the constraints outlined in section 2.

4.1 Asynchrony

We assume that the time between transactions for a given pair is approximately equal for all pairs. However, to make sure that updates are independent, our updating scheme should have the property that the pairings, when viewed globally, do not happen in the same order. (Otherwise, the identical order could be exploited to have the effect of synchronization.)

A scheme that meets these requirement is the following: let $\{\{i,j\}|j \in N\}$ be the set of pairs that include site i. Each pair in the set will update independently after an interval $T(1+X)$, where T is the period, X is a uniform random variable drawn from $(-a, a)$ and $0 < a \ll 1$. This happens concurrently for all sites.

The above is described from a parallel point of view. Sequentially, the updating scheme can be viewed approximately as updating the pairs in a fixed order, with a small probability in each period for each pair to swap its spot in the schedule with a pair that is near it in time.

4.2 Irregular geometry

To test the model's lack of dependence on regular geometry, we need a lattice that is non-crystalline, but nevertheless "spacelike". This entails at least the following two conditions:

- Each site has about the same number of neighbors, with a low variance in the neighborhood size.
- Neighbors of a site have a high probability of being neighbors of each other.

Each site is assigned a Euclidean position randomly. For a site P, all processors within a radius r from P become P's neighbors, and each in turn has P as a neighbor.

A placement where all the coordinates are chosen at random leads roughly to a Poisson distribution of distances between sites, and therefore to a high variance in the number of neighbors. Instead, a distribution is chosen so that the average distance between connected sites will vary, but be close to some mean distance. This can be done by placing sites at random, but if two sites are too close to each other, eliminating one of them (see below).

An example of an irregular lattice in two dimensions produced as above is given in Fig. 1.

4.3 Realization

The models described here could be implemented on an amorphous computer [1] very straightforwardly. But as the first amorphous computers are just being built, and as we would like to test the effects of asynchrony and irregularity by varying them, it was necessary to simulate an amorphous computer on a sequential machine.

4.3.1 The Simulation environment: HLSIM

HLSIM [8] takes a single program in the high-level language Scheme [12] and simulates the execution of copies on thousands of asynchronous processors. Each processor can only communicate with some neighboring processors. Each has the ability to schedule local events in time using threads. HLSIM transforms these local events into a global schedule, and the order of events in this schedule is consistent with the order the events would fall in if each processor were running independently in parallel.

The connections between the processors can be specified in a geometric way. To satisfy the conditions given in section 4.2, the following algorithm was used:

- N_{start} sites are given a location in a Euclidean space using a Poisson distribution. That is, each coordinate is merely chosen uniformly from $[0, 1)$ at random.

- A site s is chosen at random. All sites within a distance r_{min} from s are deleted. This is repeated with a new randomly-chosen site until all remaining sites have been traversed. There are now no two processors within r_{min} of each other.

- For each site s, all sites within a radius r of s are connected to$s$.

This will eliminate some fraction $1 - d$ of the original sites which is a function of N_{start} and r_{min}. d was determined empirically in order to obtain a desired number of sites $N_{start}d$ (accurate to within a few percent). An example of a lattice produced as above is shown in fig. 12.

The program run by each processor is very simple. A processor N_0 knows the identity of its neighbors, and for each neighbor N_i, it starts a separate thread i. This thread initially starts out by waiting for a random time drawn from $[0, T)$. This is just to prevent the first transaction of every site from happening in a single instant. Then, a transaction takes place between N_0 and N_i, which causes the state of the pair $Q(s_0, s_i)$ to be replaced by $f(Q(s_0, s_i))$. The transaction is assumed to be atomic. The processor waits for a time $T + X$, where X is a random variable drawn uniformly from $[0, a)$. This is then repeated.[4]

The full source code is given in the appendix.

4.4 Observations

By starting the processors in some initial configuration and watching the evolution of the system, it becomes evident that the model supports all phenomena of linear waves. Fig. 6 demonstrates superposition, showing an initial pulse splitting

---

[4] Processor i also has a thread running that updates pair $(0, i)$, so each pair updates, on average, twice in time T. This is arbitrary.

into left-travelling and a right-travelling halves, which eventually wrap around the periodic boundary conditions and pass through each other unchanged.

Fig. 7 shows the same evolution for the energy-conserving model. Momentum is shown (since the model has no absolute position). The total energy over all pairs was computed at the beginning and various points during the evolution and found to be identical within the limits of machine precision, with a value of 0.33980601.

Figure 9 shows a time sequence of snapshots from a two-dimensional simulation with fixed boundary conditions. The initial condition is a sine-shaped perturbation in the center; a wave ripples out and reflects off the boundaries to interfere with itself. Various colors are used to indicate q, as shown in figure 8. There are about 5,000 processors, with an average neighborhood size of 6.365 and variance 1.133. Between the snapshots, each pair has made roughly 15 transactions. Figure 12 shows a single frame from a simulation with the same initial conditions, but with periodic boundary conditions.

In figure 10, the boundaries are fixed as in figure 9, except that the right boundary is shaped like a parabola. This is a “mirror” and the initial line wave is focused into a point.

By altering the properties of the interconnections, different “media” can be created for the waves to propagate through. In figure 11, there is also a line wave, but now there is a central region of more dense processors. Inside this region, the communication radius is shrunk so that the average number of neighbors is about the same as outside it. It thus has a different “index of refraction” than the rest of the medium and refracts the wave.

4.5 Quantitative comparisons

To test the model quantitatively and assess the effects of asynchrony and irregular geometry, we will compare its evolution to a solution to the continuous partial differential equation with the same initial conditions. Figures 13 through 17 show to what extent the model agrees with an exact solution of the continuous wave equation in one dimension:

$$\frac{d^2q}{dt^2} = c^2\frac{d^2q}{dx^2}. \tag{13}$$

with boundary conditions[5] $q(0,t) = 0$ and $q(1,t) = 0$, and initial condition $q(x,0) = sin(\pi x)$. This equation has a solution

$$q(x,t) = sin(\pi x)cos(2\pi t), \tag{14}$$

which is a standing wave. Figure 13 shows q for the processor closest to the center, plotted against a least-squares best fit of equation 14.

Figure 14 shows the difference of the model with the PDE with time for various levels of asynchrony a, plotted against average number of transactions per processor. 101 processors were arranged regularly with two neighbors each. Each site is given an initial value corresponding to $q(x,0) = sin(\pi x)$. That is, if $x(s)$ is the Euclidean coordinate of site s, the initial coordinate is set to $sin(\pi x(s))$. Periodically, at time t, a measure $D(t)$ of the local difference of $q(s,t)$ with the PDE value at the corresponding time and location $q_{PDE}(x(s),t)$ is computed and averaged over all sites. We used the square root of the average square of the difference, $D(t) = \frac{1}{|G|}\sqrt{\sum_{s \in G}[q_{PDE}(x(s)) - q(s)]^2}$, where G is the set of all sites. As with all comparisons of discrete and continuous models, the difference grows over time because there is some dispersion of the wave. Decreasing a below 0.15 does not have any appreciable effect.

The properties of wave propagation in the model were examined by setting the initial conditions to be a sinusoid pulse of width w:

$$g(x,0) = \begin{cases} cos(\frac{2\pi}{w}x) + 1, & \text{if } \frac{1}{2} - \frac{1}{2}w < x < \frac{1}{2} + \frac{1}{2}w \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

The wave equation with these initial conditions has as a solution the superposition of two pulses, one travelling left and the other travelling right:

---

[5] In the model, the boundary conditions are imposed by giving the sites at the boundary infinite mass.

$$q_{left}(x,t) = \begin{cases} \frac{1}{4}[cos(\frac{2\pi}{w}(x - vt)) + 1], & \text{if } vt - \frac{1}{2}w < x < vt + \frac{1}{2}w \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

$$q_{right}(x,t) = \begin{cases} \frac{1}{4}[cos(\frac{2\pi}{w}(x + vt)) + 1], & \text{if } -vt - \frac{1}{2}w < x < -vt + \frac{1}{2}w \\ 0 & \text{otherwise} \end{cases} \quad (17)$$

Figure 15 shows the agreement of the model with the above solution; it plots the square root of the average square of the difference $D(t)$ as above. $D(t)$ is plotted for 101 sites and for 201 sites, versus number of transactions. With a greater density of sites, the wave propagates more slowly; figure 16 shows the same plot, but the x-axis is normalized so that two periods are completed in each case (a period being the time it takes the two halves of the pulse to wrap around the boundaries to their original starting points). With twice as many sites, disagreement with the PDE due to dispersion is significantly reduced. It is close to the disagreement for 101 sites with regular geometry. Thus we can compensate for the effects of irregularity by using a greater density of sites.

Figure 17 shows a comparison of the energy-conserving model to the PDE solution. It shows $D_p(t)$ for 2 periods. $D_p(t)$ is as $D(t)$ but for momentum: $D_p(t) = \frac{1}{|G|}\sqrt{\sum_{s \in G}[p_{PDE}(x(s)) - p(s)]^2}$. The model is compared to the solution of the PDE for momentum,

$$p_{left}(x,t) = \begin{cases} -\frac{1}{4}sin(\frac{2\pi}{w}(x - vt)), & \text{if } vt - \frac{1}{2}w < x < vt + \frac{1}{2}w \\ 0 & \text{otherwise} \end{cases} \quad (18)$$

$$p_{right}(x,t) = \begin{cases} \frac{1}{4}sin(\frac{2\pi}{w}(x + vt)), & \text{if } -vt - \frac{1}{2}w < x < -vt + \frac{1}{2}w \\ 0 & \text{otherwise} \end{cases} \quad (19)$$

5 Digression: Models of computation and physical models

The class of models presented here corresponds to a new parallel computing architecture known as an amorphous computer.[1] This architecture is intended to take advantage the possibility of manufacturing small processors in such large numbers that precise interconnections and regular structure of conventional parallel computers are not feasible. An amorphous computer consists of a large number of processing elements embedded in a surface or volume. Processors are limited in resources, unreliable and connected in imprecise ways. They communicate with each other locally, for example by wireless broadcast. They have no detailed knowledge of their precise geometrical arrangement, and no global clock.

The amorphous model of computation is informed by physical constraints; it is based on the fact that processors are embedded in space, and have the locality constraints of physics (local interconnections and no global clock). Microscopic conservation is another constraint from physics that is useful in amorphous computing. Here it was used in simulating physical systems with conservation laws, but it may find use in non-simulation applications as well, since it is very similar to the idea of "tokens" in distributed systems. Tokens are used when it is required that there be only a fixed number of entities in a distributed system, such as processes that have access to a certain resource. They can be passed around between processors, but not created or destroyed.

Other models of computation informed by physical constraints have close correspondences with physical models. Physical models correspond to models of computation, at a minimum, when the range of possible states in the two is the same, and the same dynamics or evolution are possible in each. In addition, there is a two-way correspondence between them: one could conceivably implement the computer in the physics as described by the model; and conversely, straightforwardly simulate the model of physics on the computer. Thus it is for example with quantum computation and quantum physics. Quantum computation was first conceived [15] as a way of efficiently simulating the evolution of a quantum system; more recently they have been proposed as a general purpose computing architecture.

Being inherently parallel, discrete models correspond to certain models of parallel computation. In addition to amorphous computing and the amorphous physical models presented in this thesis, another pair of corresponding models is Cellular Automata and 3-D mesh architectures. A 3-D mesh architecture [16] is a parallel architecture with local interconnections and regular geometry.

5.1 Classifying models of computation based on physical constraints

The Church-Turing Thesis expresses the belief among computer scientists that all these models are equivalent in the class of computations they can theoretically achieve, given unlimited resources. However, taking physics into account imposes constraints, and provides a more realistic view of what computations are achievable as the computer becomes large and comes up against physical limits [10].

There is often more than one relevant physical constraint. These constraints can conflict with each other, making tradeoffs necessary. We might thus distinguish models of computation from one another based on the degree of various tradeoffs they make in physical resources, by classifying them on an axis that represents different amounts of two mutually conflicting resources (where having more of one implies less of the other is available.)

One axis represents granularity or distributedness. On one side of the axis are architectures that have the ability to access all parts of the system in constant time. Here are found traditional models of computation such as Turing machines. A model that more resembles current sequential machines is the Random Access machine[9]. In this model, there is a memory, every location in which is accessible in constant time. Models with constant-time access, however, do not take space into account. The result is that as the system becomes large, this capability becomes unphysical, since information takes time to propagate from one part of a system to another.

This can be dealt with by distributing the computation over space. The cost of this distribution is that it takes time to move information from one place to another. Distribution entails breaking the computer into a number of elements each of which has high internal bandwidth, connected together by links with limited bandwidth. Thus we give up the ability to reach all parts of the system in constant time. However, distributedness is a resource because when the computation is highly distributed, one can enlarge the computation simply by using more space (processors), without having to go through the "von Neumann bottleneck".

One step along the axis from the Random Access Machine class are conventional parallel computers, which contain a relatively small number of processing elements fully connected with each other. Each processing element is a smaller version of the Random Access Machine. As we move along the axis, the number of elements increases, and we must give something up: it becomes impractical to connect every element to every other. The reason is that as the number of elements scales up, not only the total bandwidth but also the bandwidth per processor would need to increase to maintain full connectedness[10]. Connecting each processor to every other becomes incompatible with the constraints of physical space. Thus, architectures that have local interconnections become necessary with a large number of processors.

Amorphous computing represents a step further along the axis, being even more distributed than conventional parallel architectures. Amorphous processors are assumed to be unreliable, and can fail. This requires redundancy: the computation must be distributed to such an extent that the failure of some elements still allows the computation to proceed successfully.

Granularity or distributedness is only one of several axes that represent physically realizable architectures. An example of another is the energy/time axis: a computation can be made to use an arbitrarily small amount of energy, at the cost of being slow[17]. It is an interesting topic for future work to identify a small number of relevant axes which represent the constraints of physics, each one representing a tradeoff between conflicting resources. This would produce a space of models that are potentially physically realizable; we could then classify models of computation on them. (New models might also come to light by identifying regions of the space that are uninvestigated.)

6 Conclusion

We have shown that two important features that current discrete models have – crystalline geometry and global synchronization – are unnecessary to produce a model with physical behavior. To compare with what happens in three-dimensional space over time, one needs to put constraints on the interconnections of the lattice and the schedule of the updates to be “spacelike” and “timelike”. However, it is not necessary for any one site to exploit detailed knowledge about what time other sites update, and about their precise spatial arrangement.

In particular, synchronization of all sites in space with each other is not required – either actual synchronization, or schemes that effect synchronization in an asynchronous environment. Also, we found that anisotropies due to the irregular lattice can be compensated for by using a higher density of sites, so these will probably not be significant at large scales. Synchrony and crystalline geometry, then, are useful when we want to simulate the models on machines that have these properties, but are not intimately connected with discrete physical models. The wave models presented do without both, but agree well quantitatively with partial differential equations.

Many interesting questions remain. One topic for further investigation is to determine what other kinds of physical systems, including nonlinear ones, can be modelled using this approach. Also, in the model presented, the states of the sites in the wave model are continuous quantities; it is open whether a wave model with the same restrictions, but where sites can take on a very restricted set of states (such as 0 or 1), is possible, and if not, what extra conditions would be needed.

We expect that, if amorphous computers become feasible, models like the class presented will become of practical interest.

References

[1] Sussman, Gerald Jay, Harold Abelson, and Tom Knight. "Amorphous Computing." White paper, 1995.

[2] Abelson, Harold, Gerald Jay Sussman, Meinhard Meyer, and Jack Wisdom. *Structure and Interpretation of Classical Physics*. Draft, December 1998.

[3] Margolus, Norman. *Physics and computation* (Laboratory for Computer Science technical report 415). Cambridge, Mass: Laboratory for Computer Science, Massachusetts Institute of Technology, 1988.

[4] Toffoli, Tommaso, and Norman Margolus. *Cellular Automata Machines: A New Environment for Modeling.* Cambridge, Mass: MIT Press, 1987.

[5] Rothman, Daniel H. *Lattice-gas Cellular Automata : Simple Models of Complex Hydrodynamics*. Cambridge, U.K., New York : Cambridge University Press, 1997.

[6] Toffoli, Tommaso. "Cellular automata as an alternative to (rather than an approximation of) differential equations in modeling physics". *Physica D* **10**, 117 (1984).

[7] Sussman, Gerald Jay, Harold Abelson and Julie Sussman. *Structure and Interpretation of Computer Programs*, 2nd edition. Cambridge, Mass: MIT Press, 1996.

[8] Adams, Stephen. "A high level simulator for Gunk." Draft, 1997.

[9] Papadimitriou, Christos H. *Computational Complexity*. Addison-Wesley, 1994.

[10] Vitanyi, Paul M. B. "Locality, Communication and Interconnect Length in Multicomputers". *Siam Journal of Computing* **17**, 4 (1988).

[11] Fredkin, Edward. “Digital mechanics. An informational process based on reversible universal cellular automata.” *Physica D* **45** 1-3 (1990).

[12] W. Clinger and J. Rees (editors). Revised^4 Report on the Algorithmic Language Scheme. Nov. 1991.

[13] Greenspan, Donald. *Arithmetic Applied Mathematics*. Oxford: Pergamon, 1980.

[14] Frank, Michael P. *Reversibility for Efficient Computing*. Ph.D. Thesis, MIT Artificial Intelligence Laboratory, 1999.

[15] Feynman, Richard. “Simulating physics with computers.” *International Journal of Theoretical Physics* **21**, 467 (1982).

[16] Leighton, F. Thomson. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. San Mateo, California: Morgan Kaufmann Publishers, 1992.

[17] Bennett, Charles H. “Logical Reversibility of Computation”. *IBM Journal of Research and Development* **17**, 525 (1973).

A Simulation program

Below is given the program run on each one of the several thousand simulated processors. It implements the momentum-conserving model.

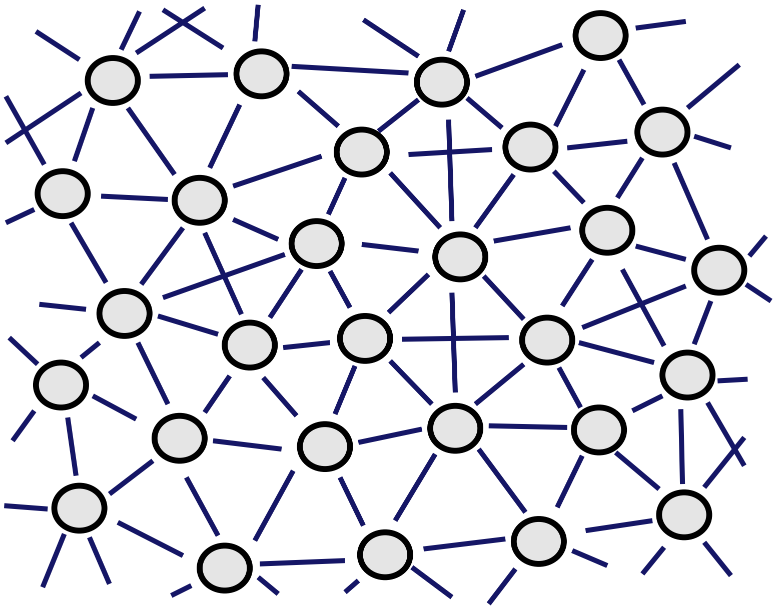

Figure 1: An example of an irregular lattice in two dimensions.

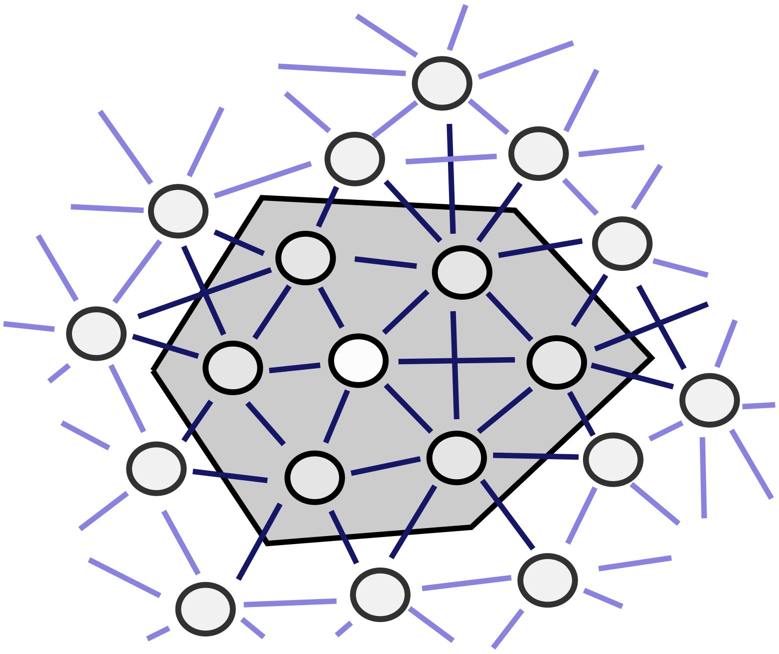

Figure 2: A cluster of sites.

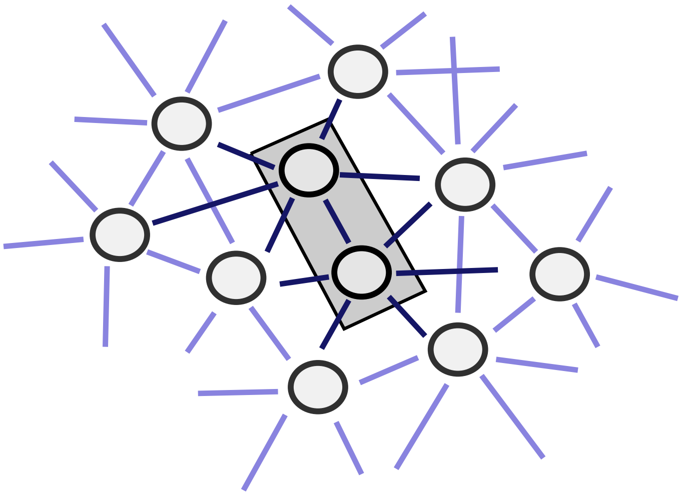

Figure 3: A “box” enclosing a pair of neighbors.

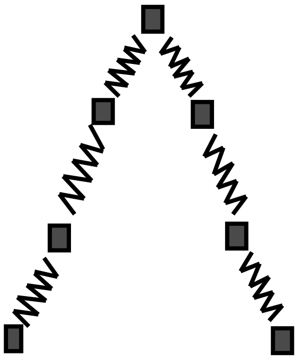

Figure 4: Two neighboring sites exert a “force” on each other as if they were masses connected by springs. Each mass is free to move in one dimension.

-bb-error = =

Figure 5: Two-dimensional evolution of diffusion model. $a = 0.05$, $\Delta t = 0.2$.

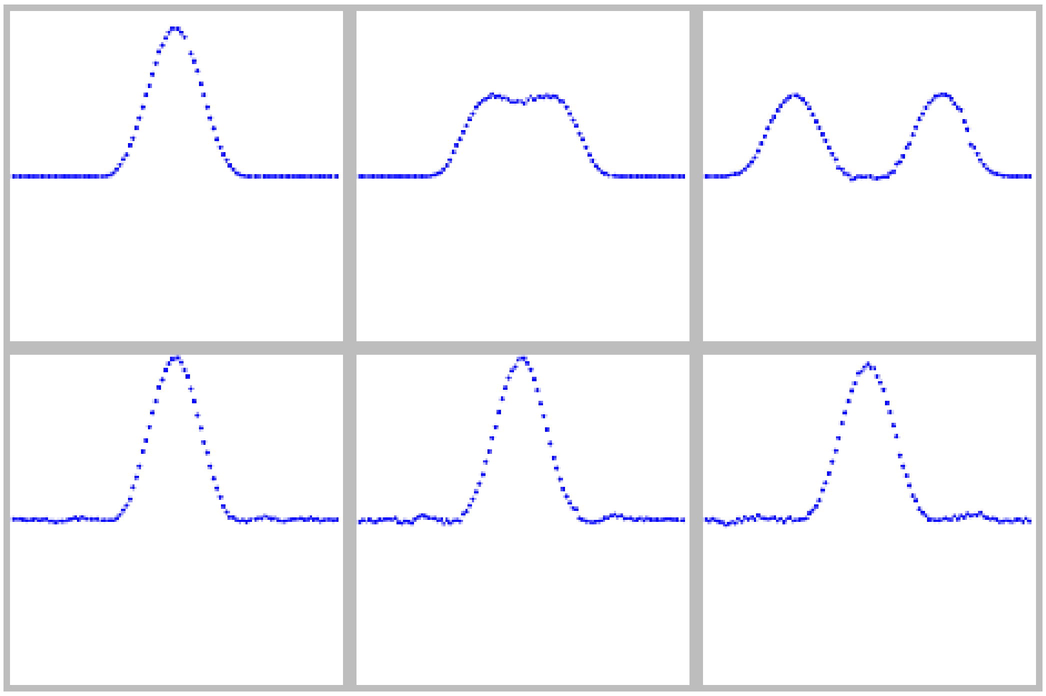

Figure 6: Superposition. $a = 0.05$, $\Delta t = 0.01$, $k = 1$, neighborhood size variance = .0384. Sites are equally spaced but have variable sized neighborhoods. Above, the pulse is shown splitting into left-travelling and right-travelling pulses. Below, the pulses are shown after they have wrapped around the periodic boundary conditions and passed through each other twice, four times, and six times.

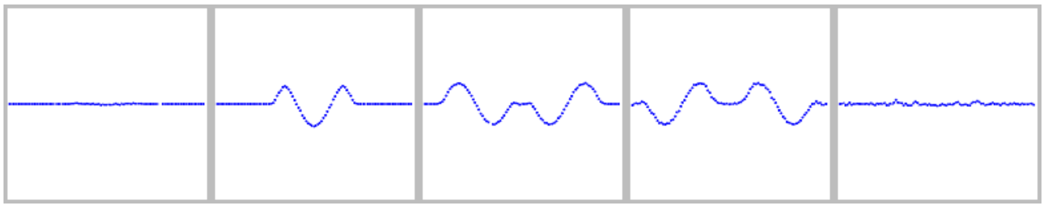

Figure 7: A momentum wave in the energy-conserving wave model. The initial conditions are a pulse as in figure 6, and parameters are the same. There are 101 regularly spaced sites. Momentum rather than position is shown. The first four frame show the pulse in the process of splitting; the last shows it after one period. Energy was calculated to be identical before, during, and after the sequence, with a value of 0.33980601.

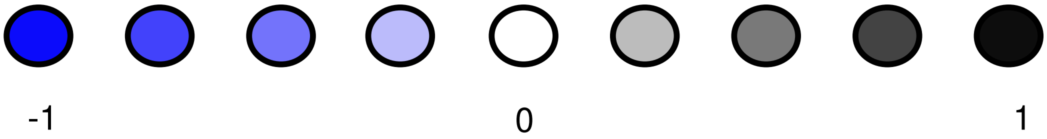

Figure 8: Colors used in illustrations. Shades indicate amplitude from blue (-1) through white (0) to black (1).

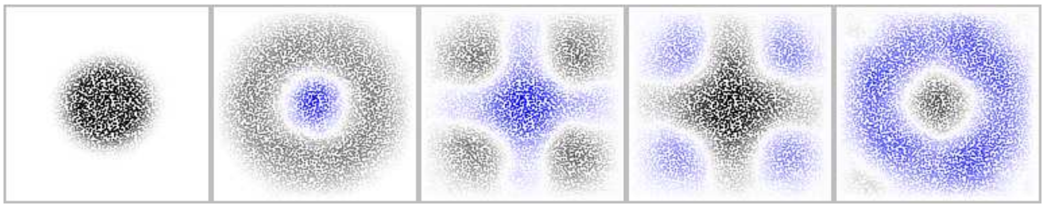

Figure 9: Two-dimensional evolution of wave model. $a = 0.05$, $\Delta t = 0.1$, $k = 1$.

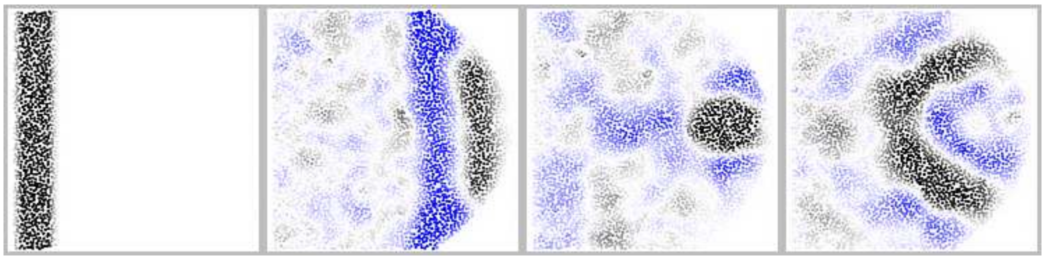

Figure 10: Reflection. There is a parabola-shaped fixed boundary at right. The initial condition is a line wave, which becomes focused to a point in the third frame.

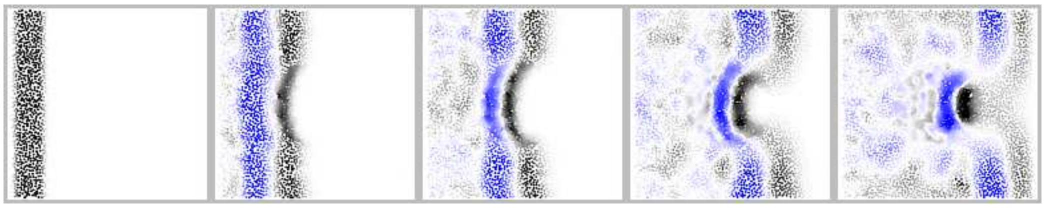

Figure 11: Refraction. The central region has twice the density of sites as the rest of the space, but the neighborhood size remains about the same.

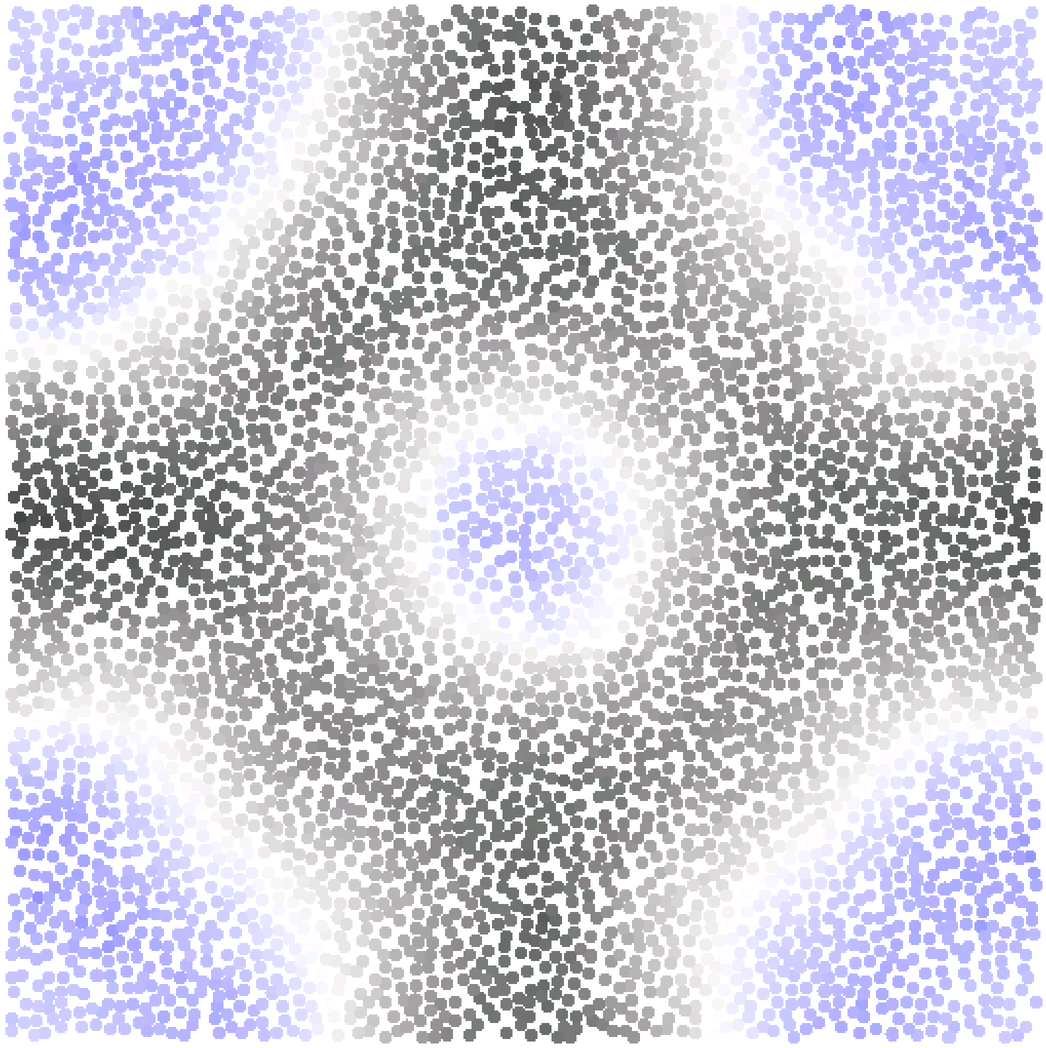

Figure 12: Snapshot of amplitudes. As in Fig. 9 but with periodic boundary conditions.

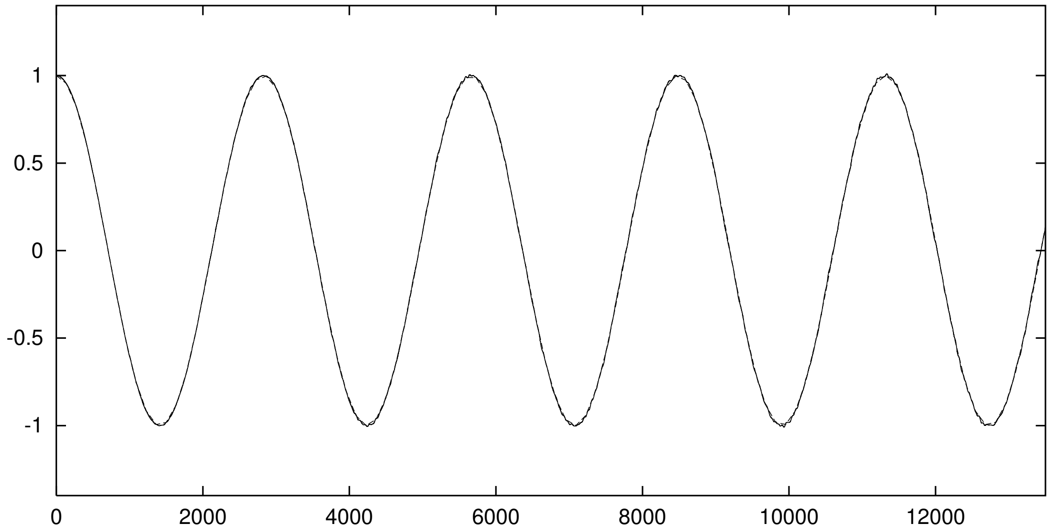

Figure 13: Standing wave, comparison with exact solution. There are 101 processors arranged regularly in one dimension with two neighbors each; the graph shows the amplitude of the processor closest to the center plotted against average number of transactions per processor, versus the exact solution of the wave equation, $q(x,t) = sin(\pi x)cos(2\pi t)$, with initial conditions $q(x,0) = sin(\pi x)$. $a = 0.05$, $\Delta t = 0.05$, $k = 1$.

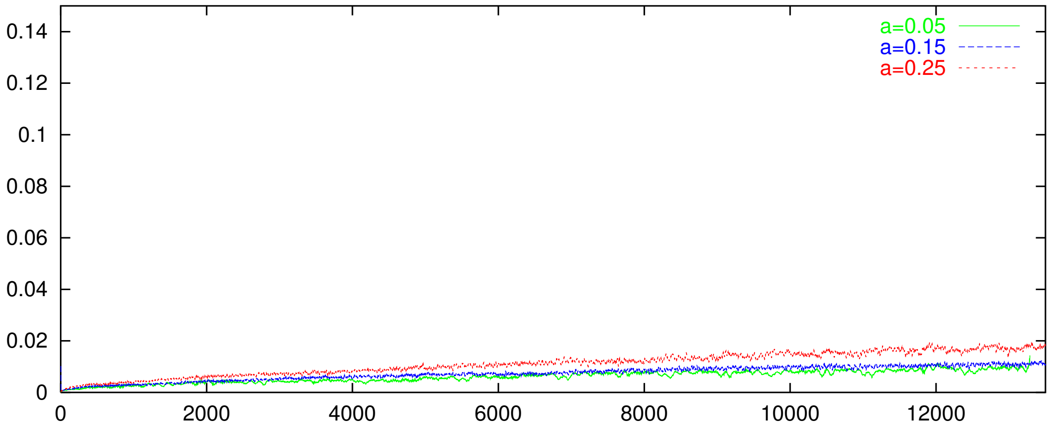

Figure 14: Comparison with solution of PDE: $D(t)$ with exact solution over time for different levels of asynchrony a. The x-axis represents the average number of transactions per processor.

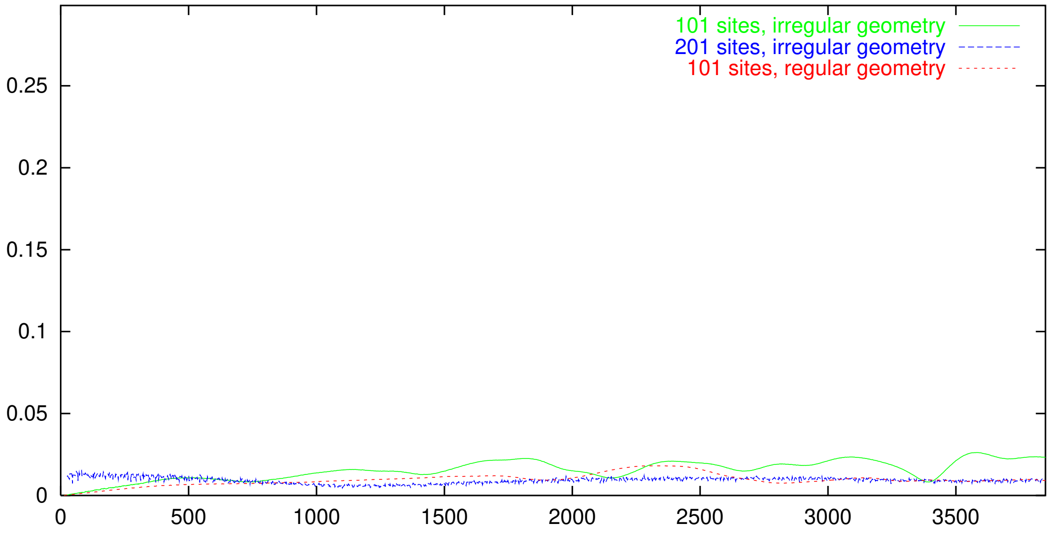

Figure 15: Pulse, comparison $D(t)$ with PDE solution. The sites are in one dimension with variable sized neighborhoods; the variance in neighborhood size is .0384. $D(t)$ is plotted for regular and irregular geometry with 101 sites, and irregular geometry with 201 sites. In the first two, the pulse completed about two periods while in the last, it completed one period (a period being the time it takes the two halves of the pulse to wrap around the boundaries to their original starting points). $a = 0.05$, $\Delta t = 0.01$, $k = 1$. The effect of irregularity is compensated for by using a greater density of sites.

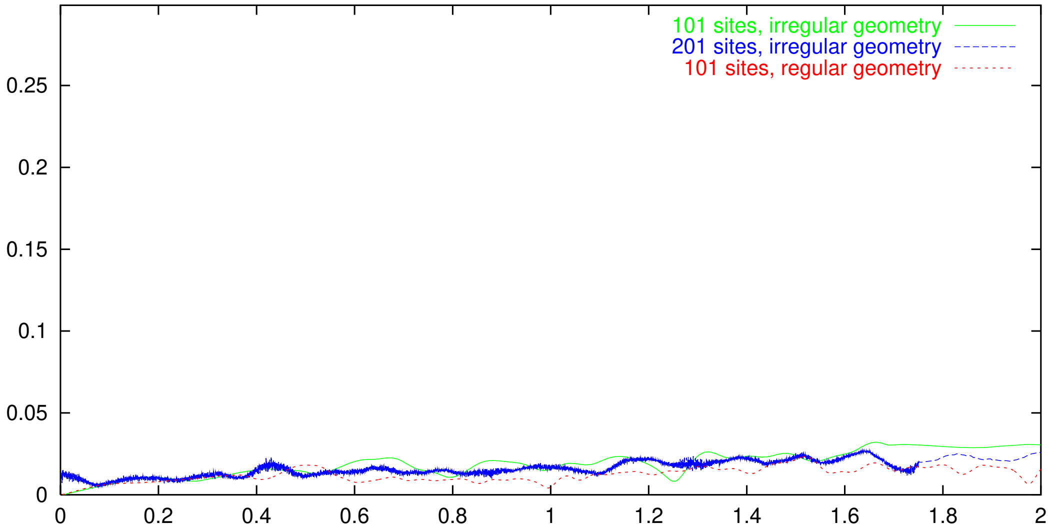

Figure 16: As above, but normalized so that exactly two periods are completed in each case. The x-axis here represents periods.

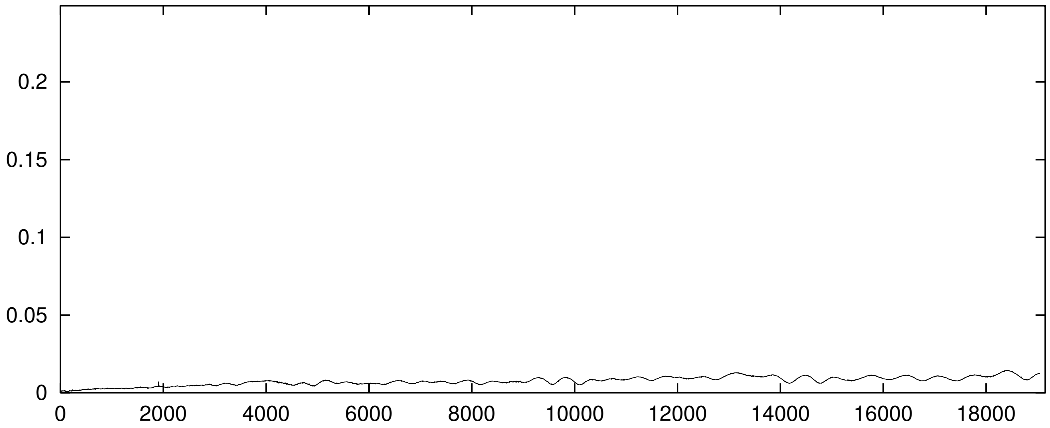

Figure 17: Comparison with PDE solution $D_p(t)$ for the momentum wave.