A Computationally Universal Field Computer That is Purely Linear

David H. Wolpert*
Bruce J. MacLennan†

DRAFT
Not to be Reproduced or Generally Distributed

September 1, 1993

Abstract

As defined in MacLennan (1987), a *field computer* is a (spatial) continuum-limit neural net. This paper investigates field computers whose dynamics is also continuum-limit, being governed by a purely linear integro-differential equation. Such systems are motivated both as a means of studying neural nets and as a model for cognitive processing. As this paper proves, such systems are computationally universal. The "trick" used to get such universal nonlinear behavior from a purely linear system is quite similar to the way nonlinear macroscopic physics arises from the purely linear microscopic physics of Schrödinger's equation. More precisely, the "trick" involves two parts. First, the kind of field computer studied in this paper is a continuum-limit threshold neural net. That is, the meaning of the system's output is determined by which neurons have an activation exceeding a threshold (which in this paper is taken to be 0), rather

*The Santa Fe Institute, 1660 Old Pecos Trail, Suite A, Santa Fe, NM, 87501 (dhw@sfi.santafe.edu).
†Computer Science Department, University of Tennessee, Knoxville, TN 37996-1301 (maclennan@cs.utk.edu).

than by the actual activation values of the neurons. Second, the occurrence of output is determined in the same thresholding fashion; output is available only when certain *output-flagging* neurons exceed the threshold, rather than after a certain fixed number of iterations of the system. In addition to proving and discussing their computational universality, this paper cursorily investigates the dynamics of these systems.

1 Introduction

This paper is an investigation of computation systems which can be viewed as neural nets (Rumelhart, et al., 1986) continuous both in space and time. Such systems are variants of what we have called *field computers* (MacLennan 1987, 1990). Their motivation is twofold: (1) since continuous systems are usually easier to analyze than discrete ones, one might hope that field computers are easier to analyze than traditional discrete neural nets; (2) one might also hope that field computers provide an effective approximation to very massively parallel neural nets.

Section 2 starts by giving several motivations for the system studied in this paper and the dynamical equation governing its evolution. Section 3 then proves that due to the way in which meaning is assigned to the state of the system, the system's dynamics is computationally universal, despite the fact that that dynamics is purely linear. This is perhaps the main result of this paper; it means that no nonlinear (e.g., sigmoidal) neurons are needed to achieve computational universality. The final section of this paper cursorily discusses various approaches to exactly solving the dynamics of the system, and to training the field computer to reproduce an arbitrary training set.

2 The System Under Consideration

2.1 The system as a continuum-limit neural net

A field, as defined in MacLennan (1987, 1990), represents the activation state of a continuum-limit neural network; in mathematical terms a field is a real-valued function on a continuum, typically an n-dimensional Euclidean space. Thus a field ϕ is a function $\phi : \mathbf{R}^n \rightarrow \mathbf{R}$, and the set of all such fields, which we write $\Phi(\mathbf{R}^n)$, is some convenient space of functions over $\mathbf{R}^n$ (such as $L_2(\mathbf{R}^n)$). Since there is a continuum of neurons, they are indexed by real vectors $\mathbf{r} \in \mathbf{R}^n$, and the activation of a neuron is represented by $\phi_\mathbf{r} = \phi(\mathbf{r})$, the field's value at that point.

In accord with motivation (1) for field computers (Section 1), this paper concentrates on systems whose dynamics is exactly linear. The system in this paper is a specification of the state of the field at all times t, i.e., is a function $f : \mathbf{R} \rightarrow \Phi(\mathbf{R}^n)$. Intuitively, $\phi = f(t_0)$ is interpreted as the state of a continuum-limit neural net at time t_0; $\phi_\mathbf{r} = f(t_0, \mathbf{r})$ is the activation value at the time t_0 of the neuron indexed by $\mathbf{r}$.

The dependence of f on t (i.e., the dynamics of the net) is determined by the continuum-limit version of neural net dynamics, i.e., by the continuum-limit version of multiplying by a weight matrix. More precisely, the dynamics is given by the (linear) integro-differential evolution equation,

$$\partial_t f(t, \mathbf{r}) = \int \mathrm{d}V' \, G(\mathbf{r}, \mathbf{r}') f(t, \mathbf{r}'), \tag{1}$$

which we abbreviate as

$$\dot{f}(t) = G f(t). \tag{2}$$

The "weight matrix" of the net corresponds to the kernel (G) of this evolution equation, which is a fixed field $G \in \Phi(\mathbf{R}^n \times \mathbf{R}^n)$. In particular, a kernel that is non-zero for all values of its arguments corresponds to a neural net whose weight matrix is fully (recurrently) connected. Note that no nonlinear sigmoidal function is involved in the dynamics.

The most natural way of assigning meaning to the distribution $f(t)$ is in terms of its support across $\mathbf{R}^n$. (This corresponds to interpreting the state of a neural net by examining which neurons have activation values exceeding a certain threshold). So for example, if the support across $\mathbf{R}^n$ of $f(t_0)$ covers a region Σ_1, then we interpret $f(t_0)$ as having one meaning, whereas if instead the support covers a different region Σ_2, we interpret $f(t_0)$ as having some different meaning. The actual values of f across $\mathbf{R}^n$ are irrelevant, except insofar as they determine the support of f.[1]

In this paper, even the time when output occurs is determined by the support of f (as opposed to via a rule like "output occurs at $t = t_1$" for some pre-determined t_1): output is signaled when the support covers a predetermined *output-flagging* region of $\mathbf{R}^n$. So for example, if t_2 is the earliest time when the support of f covers the output-flagging region, then the output of the system is determined by the distribution of $f(t_2)$'s support over $\mathbf{R}^n$. With this scheme, the amount of time the net runs is a variable, which in general depends on the input values fed into the net (i.e., depends on the field $f(t_0)$).[2]

2.2 The system as a cognitive processor

In addition to the perspective taken above, in which the system and its dynamics are viewed as a continuum-limit linear neural net and the meaning of the system's output is determined via the support of f, there are other ways of interpreting a system f evolving according to the equation $\dot{f}(t) = Gf(t)$. In particular, such a system can be viewed as a "cognitive processor" operating in (massive) parallel. The idea is to view the value of $f(t, \mathbf{r})$ as the "confidence" one has at time t in the proposition labeled by $\mathbf{r}$. The dynamical evolution of the system is the system trying to determine the answer to a question encoded as $f(t_0, \mathbf{r})$. This process can be viewed as infinite parallel streams of thought, each with different confidence levels, interacting with one another in an attempt to answer the question. (The interaction consists of transferring confidence among the various possible $\mathbf{r}$ values according to the evolution equation.)

This confidence-level interpretation doesn't ascribe meaning only to the support of $f(t)$, but also takes into account the actual values of the field $f(t)$. Nonetheless, one might still wish to flag output by running the system until the support of $f(t)$ covers a pre-determined output-flagging region of $\mathbf{R}^n$. In this context, such output flagging means simply that the system processes a question for as long as it takes for it to determine that it has an answer, which (in the form of $f(t)$, the distribution across $\mathbf{R}^n$ of confidence levels) is signaled when the output is flagged (i.e., when one has non-zero confidence that a decision has been made). As an alternative, one could instead have the dynamics halt either when output is flagged or when t exceeds some special value t_c, in which case $t > t_c$ means that the system can't find an answer to the question.

The evolution equation used in this paper accurately reflects this confidence-level interpretation, assuming that we can express G as

$$G(\mathbf{r}, \mathbf{r}') = H(\mathbf{r}, \mathbf{r}') - \delta(\mathbf{r}' - \mathbf{r}) \int \mathrm{d}V'' H(\mathbf{r}'', \mathbf{r}) \qquad (3)$$

for some field $H \in \Phi(\mathbf{R}^n \times \mathbf{R}^n)$. To see this, note that if this assumption holds then our evolution equation $\dot{f}(t) = Gf(t)$ (Eq. 1) can be rewritten as

$$\dot{f}(t, \mathbf{r}) = \int \mathrm{d}V' H(\mathbf{r}, \mathbf{r}') f(t, \mathbf{r}') - \int \mathrm{d}V'' H(\mathbf{r}'', \mathbf{r}) f(t, \mathbf{r}).$$

The second integral represents loss in confidence in the point $\mathbf{r}$ accompanying transfer of confidence from $\mathbf{r}$ to the other points in the space.[3] The first integral, on the other hand, represents a gain in confidence in $\mathbf{r}$ due to loss of confidence in the other points.

Note that if we *can* write G as in Eq. 3, then $f(t)$ automatically maintains normalization through time: $\partial_t[\int \mathrm{d}V f(t, \mathbf{r})] = 0$, which means that our total confidence remains unchanged. Note also that in general H need not be symmetric; $H(\mathbf{r}', \mathbf{r})$ need not equal $H(\mathbf{r}, \mathbf{r}')$, which means that the dynamics governing the loss of confidence in $\mathbf{r}$ need not be the same as the dynamics governing the gain in confidence in $\mathbf{r}$.

There are a number of open questions associated with this confidence-level interpretation of f. For example, if the system is really to be viewed as a cognitive processor, with many parallel "streams of thought," then one might want to have the low-level dynamics contain the laws of deductive logic. Would this necessitate a different evolution equation? As another example of a peculiar feature of the confidence-level interpretation, what is negative confidence level?[4] Is it confidence in the negation of a statement? If so, then in the confidence-level interpretation $f(t, \mathbf{r}) = 0$ can be interpreted either as simultaneous confidence in a proposition $\mathbf{r}$ and its negation, or as no confidence in either $\mathbf{r}$ or its negation. Do we really want to treat these two scenarios as equivalent? Finally, do we really want to have conservation of $\int \mathrm{d}V f(t, \mathbf{r})$? After all, one doesn't become less confident in a given proposition just because you use it to infer other propositions in which you are confident.

To avoid dealing with these issues, this paper will stick to the interpretative scheme where the meaning of $f(t)$ is given by its support. Nonetheless, the main results of this paper hold just as well for the confidence-level interpretation.

3 Computational Universality

3.1 Introduction

Before we present our proof of the computational universality of linear field computers, it will be worthwhile to consider the concept of computational universality. Conventionally, a model of computation is taken to be computationally universal if it is functionally equivalent to the class of Turing machines. The importance of this particular class should not blind us, however, to other notions of computational universality that may be more appropriate in different contexts. More generally, a model of computation is computationally universal *with respect to a class of functions* $\mathcal{F}$ if it can implement every function in that class. Important non-Turing-universal models of computation include finite-state machines and primitive-recursive functions. The class of functions appropriate to a definition of computational universality must be determined by the use to which that definition will be put.

Several researchers have argued that Turing computability is not entirely relevent to continuum-limit computation. For example, Blum and her colleagues have developed a theory of discrete-time computation over the reals (Blum, 1989; Blum, Shub, & Smale, 1988). Also Stannett (1990) has shown that certain machines with continuous dynamics can solve the halting problem for Turing machines, and thus have super-Turing power. In addition, Pour-El and Richards (1979, 1981, 1982) have shown that non-Turing-computable solutions can result from a Turing-computable wave equation with Turing-computable initial conditions.

Indeed, we have argued elsewhere (**??**, **??**, 1993) that a more radical departure from traditional models of computation is required by the continuum limit. One possibility is presented in MacLennan (1987, 1990), where we describe a class of multilinear field computers that is universal with respect to the class of operators that have convergent Taylor series and whose Gâteaux derivatives are integral operators of Hilbert-Schmidt type.

Nevertheless, it is important to understand the relation of field computation to traditional models of computation, so in this paper we restrict our attention to Turing computability.

We mention briefly other work relating neural networks to Turing computability. As early as 1943 McCulloch and Pitts (1943) argued that a neural network *connected to an external tape* is equivalent to a Turing machine; this is quite obvious, since their neurons are threshold-logic gates and can be easily assembled into the control unit of a TM. Pollack, in Chapter 4 of his Ph.D. dissertation (1987), showed how to include the tape in the neural net by encoding the potentially infinite string of bits as a rational number of unlimited precision. His construction uses a finite number of linear-threshold units, but with multiplicative (i.e., higher order) connections, and rational weights and activities of unlimited precision. Pollack hypothesized that multiplicative (i.e., nonlinear) connections "are a critical, and underappreciated, component for neurally-inspired computing," in particular, for general-purpose computing. Hartley & Szu (1987) argued that TMs are equivalent both to potentially infinite neural networks with finite state neurons, and to finite networks of neurons with a countable infinity of states. More recently, Garzon and Franklin (1989, 1990; Franklin & Garzon, 1990) have shown that countably infinite neural nets are more powerful than the class of countably infinite cellular automata, which are in turn more powerful than TMs; in particular they can solve the halting problem for TMs. On the other hand these neural networks are less powerful than "automata nets." Although their nets are infinite, they satisfy certain other "realistic implementability" conditions; see their papers for details.

We note that it is easy to show that the class of purely linear, but countably infinite neural nets are at least as powerful as Turing machines. Simply number in any convenient way the complete states (i.e., internal state plus tape state) of the TM. Now allocate an input neuron j to each complete state j, and an output neuron i to each complete state i. Thus there is a countable infinity of neurons in each layer. Encode the state of the machine by setting the activity of the corresponding neuron to 1 and all the rest to 0; that is, state k is represented by the coordinate vector along the kth axis. Set the weight $W_{ij} = 1$ if state j leads to state i (we are assuming a deterministic machine), and $W_{ij} = 0$ otherwise. Thus the weight matrix represents the transition function. Now if $\mathbf{s}$ is a (unit) vector representing the current state of the machine, then $W\mathbf{s}$ will be a (unit) vector representing its new state. Thus an arbitrary TM can be simulated by an infinite dimensional difference equation $\mathbf{s}' = M\mathbf{s}$.[5]

The construction in this paper differs from the foregoing in going to the continuum limit in both space and time. That is, the TM's states are represented by fields that obey a differential equation. Much of the complexity of the following construction comes from getting a continuous-space, continuous-time dynamical system to emulate a system with discrete states undergoing discrete state transitions. (Of course, this is a difficulty implicit in the construction of any real-world computer.) In this, this paper parallels Omohundro's work showing how to emulate an arbitrary (discrete space and time) cellular automata with differential equations (Omohundro, 1984).

Before proceeding to that construction however, it is worth noting that if we work in $\mathbf{R}^3$ (i.e., if $n = 3$), and if f is allowed to be complex-valued, then

$$G(\mathbf{r}, \mathbf{r}') = \left(\frac{ih}{2\pi}\right)^{-1} \left[\left(\frac{-h^2}{8m\pi^2} \sum_{i=1}^{3} \frac{\partial^2}{\partial^2_{r'_i}} \delta(r_i - r'_i) \right) + \delta(\mathbf{r} - \mathbf{r}') V(\mathbf{r}') \right]$$

results in Schrödinger's equation, $ih/2\pi \times \partial_t f = -h^2 \nabla^2 f / 8m\pi^2 + Vf$. At this point we could note that any real-world TM is built of components which are, ultimately, quantum mechanical in nature, and in quantum mechanics meaning is ascribed to the support of the wave function f (for sufficiently peaked wave functions). Therefore we can immediately conclude that, via appropriate choice of the potential V, our evolution equation allows TM solutions.[6] (Strictly speaking, this correspondence between our evolution equation and quantum mechanics actually requires that Schrödinger's equation for a set of more than one interacting particles be simulated.) Alternatively, one can demonstrate the universality of our evolution equation by noting that there exists a Schrodinger's equation G, and by then appealing directly to the field of quantum Turning machine theory (Deutsch, 1985).

This argument is not particularly insightful or useful however, especially if one is interested in field computers as possible models of the human brain. To put it mildly, there are many poorly understood steps in extrapolating upwards from quantum mechanics to macroscopic human brains.

3.2 How to interpret a field as a Turing Machine

Before giving a precise formulation of how to interpret a field as a Turing machine, some notational comments are in order. We will work in $\mathbf{R}^5$ (i.e., $n = 5$). (However not all five components will be used to specify the state of the TM.) Bold lower-case letters indicate vectors, and subscripted italic letters indicate components of a vector. Let $\boldsymbol{\rho}$ be any vector in $\mathbf{R}^4$ and z a real-valued scalar; we define $(\boldsymbol{\rho}, z)$ to be the $\mathbf{R}^5$ vector $(\rho_1, \rho_2, \rho_3, \rho_4, z)$. So for example, if the 4-dimensional vector $\boldsymbol{\sigma}$ tells us something of the TM's state at time t, and if we want the 5th component of our corresponding $\mathbf{R}^5$ vector to equal t, then that corresponding vector $\mathbf{r} \in \mathbf{R}^5$ is given by $\mathbf{r} = (\boldsymbol{\sigma}, t)$. For convenience define a generator for m-dimensional delta functions:

$$\Delta(\mathbf{r}, \mathbf{s}) \equiv \prod_{k=1}^{m} \delta(r_k - s_k), \quad \text{for } \mathbf{r}, \mathbf{s} \in \mathbf{R}^m.$$

(m is implicitly determined, by the arguments of the Δ.)

The basic idea is to find a kernel G with solution f, such that the support of f, $\mathbf{r}$ can be interpreted in the following manner. First, r_5 serves as a system clock; at any particular time t there is only one value of r_5 such that $f(t, \mathbf{r}) \neq 0$, and this value of r_5 is proportional to t, $r_5 = \omega t$. (This clock is necessary to have the dynamics cycle through the various operations making up an iteration of a TM; without this clock embedded in $\mathbf{R}^n$, the dynamics has no way of knowing what TM operation to apply.) Without loss of generality we take $\omega = 1$.

In addition to this restriction on r_5, we want $f(t, \mathbf{r})$ to never be non-zero except for those r_1 through r_4 on the following lattice: $r_4 \in \{0, 1\}$, and $r_1, r_2, r_3 \in \mathbf{Z}^+$. We define $\Lambda \subset \mathbf{R}^4$ to be this lattice:

$$\Lambda = (\mathbf{Z}^+)^3 \times \{0, 1\}.$$

At any particular time t, there will only be 1 or 2 of these lattice points in Λ for which $f(t, \mathbf{r}) \neq 0$. These values of r_1 through r_4 for which $f(t)$ is non-zero code for the condition of the TM as follows: r_1 represents head position on the TM's tape, r_2 represents the numerical value on the tape (which for simplicity is assumed to have a finite number of 1's), r_3 represents the internal state of the TM, and r_4 is a buffer label. As time changes, the values of r_1, r_2, r_3, and r_4 for which $f(t, \mathbf{r}) \neq 0$

should change in exact accord with the dynamics of the TM being emulated. In effect the lattice points represent possible states of our TM emulation, and at any given time t, $f(t)$ "points" to one or two of these states.

The goal is to find a G such the evolution equation has a solution with the following properties. First, the solution must be of the form

$$f(t, \mathbf{r}) = \sum_{\boldsymbol{\sigma} \in \Lambda} \Delta[\mathbf{r}, (\boldsymbol{\sigma}, t)] K(t, \boldsymbol{\sigma}). \tag{4}$$

This solution is a superposition of five-dimensional Dirac delta functions, all of which are centered in r_5 about the point t. Each delta function is centered in $\mathbf{R}^4$ about a different one of the allowed lattice sites $\boldsymbol{\sigma}$, with magnitude $K(t, \boldsymbol{\sigma})$ at each such lattice site. In general, at any given time t there must only be 1 or 2 values of $\boldsymbol{\sigma}$ such that $K(t, \boldsymbol{\sigma}) \neq 0$. It is the dynamics of this support of $K(t, \boldsymbol{\sigma})$ which corresponds to the dynamics of the TM. In other words, the task is to construct a G such that the evolution equation has solution of the form Eq. 4, where the coordinate projections of the support of the function K obey the dynamics of the TM being emulated. In this way, dynamics over $\mathbf{R}^5$ is reduced to dynamics over Λ.

The next subsection shows how to choose a G with solution (4), for arbitrary K. The subsequent subsection shows how to choose K so that the dynamics over Λ emulates an arbitrary Turing machine. Together, these two subsections show how to choose a G so that the dynamics of the system emulates an arbitrary Turing machine.

3.3 Reducing to countably infinite dynamics

Lemma 1: Let

$$G(\mathbf{r}, \mathbf{r}') = -\partial_{r_5} \Delta(\mathbf{r}, \mathbf{r}') + \delta(r_5 - r_5') \Gamma(\mathbf{r}, \mathbf{r}'), \tag{5}$$

where, for any $\mathbf{u}, \mathbf{s}$ of the form $\mathbf{u} = (\boldsymbol{\rho}, t)$, $\mathbf{s} = (\boldsymbol{\sigma}, t)$, in which $\boldsymbol{\rho} \in \mathbf{R}^4$, $\boldsymbol{\sigma} \in \Lambda$, the function Γ obeys

$$\sum_{\boldsymbol{\sigma}} \Gamma(\mathbf{u}, \mathbf{s}) K(t, \boldsymbol{\sigma}) = \sum_{\boldsymbol{\sigma}} \Delta(\boldsymbol{\rho}, \boldsymbol{\sigma}) \partial_t K(t, \boldsymbol{\sigma}) \tag{6}$$

for some time-varying field $K_t \in \Phi(\mathbf{R}^5)$. Then Eq. 4 satisfies the evolution equation (Eq. 1). (Note that the behavior of $\Gamma(\mathbf{r}, \mathbf{r}')$ (and therefore of $G(\mathbf{r},\mathbf{r}')$) for points $\mathbf{r}'$ whose first four components do not lie on Λ is completely free.)

Proof: Our goal is to show that Eq. 1 is satisfied at the lattice points by Eq. 4 under the conditions of Eqs. 5 and 6. Let the first four components of $\mathbf{r}$ be indicated by $\boldsymbol{\rho}$, and let the first four components of $\mathbf{r}'$ be indicated by $\boldsymbol{\rho}'$. Substituting Eq. 5 and Eq. 4 into the evolution equation (Eq. 1) gives

$$\begin{aligned}
&\sum_{\boldsymbol{\sigma}} \Delta(\boldsymbol{\rho},\boldsymbol{\sigma}) K(t,\boldsymbol{\sigma}) \partial_t \delta(r_5 - t) \;\; + \\
&\sum_{\boldsymbol{\sigma}} \Delta(\boldsymbol{\rho},\boldsymbol{\sigma}) \partial_t K(t,\boldsymbol{\sigma}) \delta(r_5 - t) \\
&\quad = \\
&\qquad \int \mathrm{d}V' \left[-\partial_{r_5} \delta(r_5 - r_5') \Delta(\boldsymbol{\rho},\boldsymbol{\rho}') \sum_{\boldsymbol{\sigma}} \Delta(\mathbf{r}',\mathbf{s}) K(t,\boldsymbol{\sigma}) \right] + \\
&\qquad \int \mathrm{d}V' \delta(r_5 - r_5') \Gamma(\mathbf{r},\mathbf{r}') f(t,\mathbf{r}').
\end{aligned}$$

The first four-fold delta function $\Delta(\boldsymbol{\rho},\boldsymbol{\rho}')$ in the first term on the right-hand side of this equality can be integrated out, giving as the first term on the right-hand side

$$\left[\sum_{\boldsymbol{\sigma}} \Delta(\boldsymbol{\rho},\boldsymbol{\sigma}) K(t,\boldsymbol{\sigma}) \right] \int \mathrm{d}r_5' [-\partial_{r_5} \delta(r_5 - r_5') \delta(r_5' - t)].$$

The remaining integral in this first term reduces to $-\partial_{r_5}\delta(r_5 - t)$. Therefore the first term on the left-hand side of the equality cancels with the first term on the right-hand side, leaving the equality

$$\begin{aligned}
&\delta(r_5 - t) \sum_{\boldsymbol{\sigma}} \Delta(\boldsymbol{\rho},\boldsymbol{\sigma}) \partial_t K(t,\boldsymbol{\sigma}) = \\
&\qquad \int \mathrm{d}V' \Gamma(\mathbf{r},\mathbf{r}') \delta(r_5 - r_5') \sum_{\boldsymbol{\sigma}} \Delta(\mathbf{r}',\mathbf{s}) K(t,\boldsymbol{\sigma}).
\end{aligned} \tag{7}$$

The integral on the right-hand side reduces to

$$\delta(r_5 - t) \sum_{\boldsymbol{\sigma}} \Gamma(\mathbf{r},\mathbf{s}) K(t,\boldsymbol{\sigma}).$$

Therefore Eqs. 1, 4, and 5 jointly reduce to Eq. 6 which by hypothesis is true. **QED**

The second term in Eq. 5 (the one containing the $\Gamma(\mathbf{r},\mathbf{r}')$) is the one which can be used to fix K. The first term in Eq. 5 (the one not containing the $\Gamma(\mathbf{r},\mathbf{r}')$) is the one which forces the $\delta(r_5-t)$ t dependence on the r_5 component of $f(t,\mathbf{r})$. It is this dependence which allows r_5 to serve as a system clock for the operation of the TM; if $\Gamma(\mathbf{r},\mathbf{r}')$ varies with r_5', then as time changes the $\delta(r_5'-t)$ term in $f(\mathbf{r}',t)$ will pick out a different part of $\Gamma(\mathbf{r},\mathbf{r}')$, which means we can cycle through a sequence of different operations governing the dynamics of $f(\mathbf{r},t)$. This can be seen explicitly in the following discussion of the relationship between Γ and the dynamics of K, i.e., in the following discussion showing how lemma 1 allows dynamics over $\mathbf{R}^5$ to be reduced to dynamics over Λ.

Choose $\Gamma(\mathbf{r},\mathbf{r}') \equiv \sum_{\boldsymbol{\sigma}''\in\Lambda} \Delta(\boldsymbol{\rho},\boldsymbol{\sigma}'')A(\mathbf{r},\mathbf{r}')$ for some function $A \in \Phi(\mathbf{R}^5\times\mathbf{R}^5)$ ($\boldsymbol{\rho}$ being the vector of the first four components of $\mathbf{r}$). Because of the delta function $\Delta(\boldsymbol{\rho},\boldsymbol{\sigma}'')$, the terms of this summation are nonzero only when $\boldsymbol{\rho}=\boldsymbol{\sigma}''$, therefore the $\mathbf{r}=(\boldsymbol{\rho},t)$ appearing inside $A(\mathbf{r},\mathbf{r}')$ can be replaced with $(\boldsymbol{\sigma}'',t)$. Hence,

$$\Gamma(\mathbf{r},\mathbf{r}') = \sum_{\boldsymbol{\sigma}''\in\Lambda} \Delta(\boldsymbol{\rho},\boldsymbol{\sigma}'')A[(\boldsymbol{\sigma}'',t),\mathbf{r}'].$$

Substituting this Γ into Eq. 6 yields:

$$\begin{aligned}
&\sum_{\boldsymbol{\sigma}} \Delta(\boldsymbol{\rho},\boldsymbol{\sigma})\partial_t K(t,\boldsymbol{\sigma}) \\
&\quad= \sum_{\boldsymbol{\sigma}} \Gamma[(\boldsymbol{\rho},t),(\boldsymbol{\sigma},t)]K(t,\boldsymbol{\sigma}) \\
&\quad= \sum_{\boldsymbol{\sigma}} \left\{ \sum_{\boldsymbol{\sigma}''} \Delta(\boldsymbol{\rho},\boldsymbol{\sigma}'')A[(\boldsymbol{\sigma}'',t),(\boldsymbol{\sigma},t)] \right\} K(t,\boldsymbol{\sigma}) \\
&\quad= \sum_{\boldsymbol{\sigma}''} \left\{ \Delta(\boldsymbol{\rho},\boldsymbol{\sigma}'') \sum_{\boldsymbol{\sigma}} A[(\boldsymbol{\sigma}'',t),(\boldsymbol{\sigma},t)]K(t,\boldsymbol{\sigma}) \right\} \\
&\quad= \sum_{\boldsymbol{\sigma}''} \left\{ \Delta(\boldsymbol{\rho},\boldsymbol{\sigma}'') \sum_{\boldsymbol{\sigma}'} A[(\boldsymbol{\sigma}'',t),(\boldsymbol{\sigma}',t)]K(t,\boldsymbol{\sigma}') \right\}.
\end{aligned}$$

Then Eq. 6 for Γ in Lemma 1 reduces to

$$\sum_{\boldsymbol{\sigma}} \Delta(\boldsymbol{\rho},\boldsymbol{\sigma})\partial_t K(t,\boldsymbol{\sigma}) = \sum_{\boldsymbol{\sigma}''} \left\{ \Delta(\boldsymbol{\rho},\boldsymbol{\sigma}'') \sum_{\boldsymbol{\sigma}'} A[(\boldsymbol{\sigma}'',t),(\boldsymbol{\sigma}',t)]K(t,\boldsymbol{\sigma}') \right\}.$$

One way this equality can be enforced is if individual terms on the right cancel with individual terms on the left, i.e., if

$$\partial_t K(t, \boldsymbol{\sigma}) = \sum_{\boldsymbol{\sigma}'} A[(\boldsymbol{\sigma}, t), (\boldsymbol{\sigma}', t)] K(t, \boldsymbol{\sigma}').$$

Since t is the fifth component of both $(\boldsymbol{\sigma}, t)$ and $(\boldsymbol{\sigma}', t)$, we can re-express the dependence of A on its arguments to get the following:

$$\partial_t K(t, \boldsymbol{\sigma}) = \sum_{\boldsymbol{\sigma}'} A(t, \boldsymbol{\sigma}, \boldsymbol{\sigma}') K(t, \boldsymbol{\sigma}'),$$

which is an "infinite matrix product,"

$$\dot{K}(t) = A(t)\, K(t). \tag{8}$$

This equality is essentially a discrete-space version of the original evolution equation (Eq. 2), with one important difference. Whereas the evolution equation had a time-independent kernel G, the equation governing the dynamics of K has a kernel $A(t)$ which depends explicitly on t.[7] The only purpose of this section so far has been to arrive at a dynamics equation with such a time-dependent kernel; it is this time-dependent kernel, arising from the relationship between r_5 and t, which allows the dynamics of f to cycle through the various distinct stages making up an iteration of a TM.

The results so far can be summarized as follows. Choose a function $A(t)$. This function specifies a $\Gamma(\mathbf{r}, \mathbf{r}')$. Now choose a function $K(t)$ which satisfies Eq. 8 at the lattice points. This function specifies an $f(t, \mathbf{r})$. We know that this $\Gamma(\mathbf{r}, \mathbf{r}')$ and this $f(t, \mathbf{r})$ together satisfy Eq. 6. Accordingly, this $f(t, \mathbf{r})$ together with the $G(\mathbf{r}, \mathbf{r}')$ given by $\Gamma(\mathbf{r}, \mathbf{r}')$ jointly satisfy the evolution equation (Eq. 1). In other words, so long as we choose an $A(t)$ and a $K(t)$ which jointly satisfy Eq. 8, we will be assured that the $f(t, \mathbf{r})$ based on $K(t)$ satisfies the evolution equation with a $\Gamma(\mathbf{r}, \mathbf{r}')$ based on $A(t)$. Furthermore, $K(t, \mathbf{r})$ and $f(t, \mathbf{r})$ are non-zero for the exact same $\mathbf{r}$ values from within Λ. However the meaning of $f(t, \mathbf{r})$ is given in terms of where over Λ it is non-zero. Therefore the meaning of $f(t, \mathbf{r})$ is given by the Λ-support of the associated $K(t, \sigma)$. So our task is reduced to the following: Given any particular TM, find an $A(t)$ such that the associated $K(t)$ (associated via Eq. 8) has a Λ-support which emulates that TM. The next part of this section describes how to do this.

3.4 Emulating a particular TM

There are several separate logical operations making up an iteration of a TM. It will take the dynamics of the system exactly 1 unit of time to complete each such operation, and $A(t)$ is fixed for each such operation. In other words, if we assume that the system starts evolving at $t = 0$, then $A(t)$ remains unchanged throughout each of the separate intervals $t \in [n, n+1)$, $n = 0, 1, 2, \ldots$, i.e., $A(t)$ only changes when $\lfloor t \rfloor$ changes. Four distinct operations occurring in four such consecutive integer intervals together make up a single iteration of a TM. At the beginning of an iteration the current contents of the TM are stored in the $\sigma_4 = 0$ hyperplane. The first operation clears the buffer hyperplane ($\sigma_4 = 1$); the second operation calculates the condition which the TM being emulated will have at the end of its next iteration and stores the (suitably encoded) result in the $\sigma_4 = 1$ hyperplane (this calculation is based on the current contents of the $\sigma_4 = 0$ hyperplane); the third operation clears the $\sigma_4 = 0$ hyperplane, and the fourth operation copies the contents of the $\sigma_4 = 1$ hyperplane into the $\sigma_4 = 0$ hyperplane. At the end of this cycle, the dynamics repeats itself: $A(t) = A(t+4)$ for all t.

Without the use of a buffer plane to hold the result until the original plane is cleared out (at which time the buffer plane's contents are copied back in), essentially all states of the system would have non-zero support an infinitesimal time after $t = 0$. (I.e., let the succession of TM states be $S_1, S_2, S_3, \ldots$ If there were no buffer plane, and if the dynamics were at all times the "evolve-the-TM" dynamics of the $t = 1 \longrightarrow t = 2$ stage, then an infinitesimal time after the calculation starts (in state S_1) state S_2 would be signaled (i.e., $f(S_2)$ would be non-zero), which would instantaneously signal S_3, and so on.) This means that a system lacking buffer planes doesn't really "emulate" a TM. On the other hand, such a buffer-plane-less system automatically determines whether or not the TM will halt an infinitesimal time after the dynamics starts. (Since all states which will ever be occupied are so occupied an infinitesimal time after the start of the calculation.)

Next we present the four stages of an iteration. Without loss of generality, assume the cycle starts at $t = 0$. It's assumed that at $t = 0$ (i.e., at the beginning of every cycle) the following is true:

1. For all $\boldsymbol{\sigma}$, $K(t, \boldsymbol{\sigma}) \in \{0, 1\}$.
2. For all $\boldsymbol{\sigma}$ such that $K(t, \boldsymbol{\sigma}) = 1$, $\sigma_4 \in \{0, 1\}$.

3. There is exactly one $\mathbf{p} \in (\mathbf{Z}^+)^3$ such that $K[t, (\mathbf{p}, 0)] = 1$. Let $\mathbf{p}^*$ be this $\mathbf{p}$. (The initial value of $\mathbf{p}^*$ when the system starts codes for the initial condition of the TM being emulated.)
4. It is valid for the TM to have current head position given by p_1^*, contents of the tape given by (the binary expansion of) p_2^*, and internal state given by p_3^*.

$\boxed{t = 0 \longrightarrow t = 1}$

This stage clears the $\sigma_4 = 1$ hyperplane. During this stage, for all $\mathbf{p} \in (\mathbf{Z}^+)^3$,

$$A[t, (\mathbf{p}, 1), (\mathbf{p}, 1)] = \begin{cases} 0 & \text{if } t = 1 \\ -1/(1-t) & \text{otherwise} \end{cases} .$$

(We don't simply set $A[t, (\mathbf{p}, 1), (\mathbf{p}, 1)] = -1/(1-t)$ because this is undefined for $t = 1$.) For all other values of its arguments $A(t)$ equals 0.

Since $A(t, \boldsymbol{\sigma}, \boldsymbol{\sigma}') = 0$ unless $\boldsymbol{\sigma} = \boldsymbol{\sigma}'$, Eq. 8 reduces to $\partial_t K(t, \boldsymbol{\sigma}) = A(t, \boldsymbol{\sigma}, \boldsymbol{\sigma}) K(t, \boldsymbol{\sigma})$. In particular, if $\sigma_4 \neq 1$, then $K(t, \boldsymbol{\sigma})$ will not change in value during this stage. If σ_4 does equal 1, then

$$K(t, \boldsymbol{\sigma}) = K(0, \boldsymbol{\sigma}) \times \exp\left[\int_0^t \mathrm{d}t' A(t', \boldsymbol{\sigma}, \boldsymbol{\sigma})\right] .$$

This means that for those $\boldsymbol{\sigma}$ with $\sigma_4 = 1$, $K(1, \boldsymbol{\sigma}) = 0$ if $K(0, \boldsymbol{\sigma}) = 0$, i.e., the value of K can change through this stage only when $\sigma_4 = 1$ and $K(0, \boldsymbol{\sigma}) = 1$. For such a case, we have $K(t, \boldsymbol{\sigma}) = 1 - t$ for $t < 1$, and $K(t, \boldsymbol{\sigma}) = 0$ when $t = 1$. This means that $K(1, \boldsymbol{\sigma}) = 0$ for *all* $\boldsymbol{\sigma}$ such that $\sigma_4 = 1$, that is, the buffer hyperplane has been cleared. $K(t, \boldsymbol{\sigma})$ for all other $\boldsymbol{\sigma}$ is the same at the end of this stage as at its beginning.

$\boxed{t = 1 \longrightarrow t = 2}$

At the end of this stage $K(t, \boldsymbol{\sigma})$ for those $\boldsymbol{\sigma}$ with $\sigma_4 = 0$ is still unchanged from $K(0, \boldsymbol{\sigma})$. At the end of this stage $K(t, \boldsymbol{\sigma}) = 0$ for all $\boldsymbol{\sigma}$ with $\sigma_4 = 1$ (just like at the end of the first stage) except for one: $K(2, \boldsymbol{\sigma}) = 1$ for the point $\boldsymbol{\sigma}$ with $\sigma_4 = 1$, and with σ_1, σ_2, and σ_3 values corresponding to the condition of the TM being emulated one iteration after it had the condition $\mathbf{p}^*$. Thus the new state has been placed in the buffer hyperplane. The detailed description of this stage follows.

Let the TM we're emulating have transition functions taking head position r, numerical tape value $x \equiv \sum_{i=0}^{\infty} \alpha_i \times 2^i$ (where for any fixed i, $\alpha_i \in \{0, 1\}$; the sequence of α_i is the contents of the TM's tape), and internal state q, to position $R(r, \alpha_r, q)$, tape value $\sum_{i=0}^{\infty} \beta_i(r, x, q) \times 2^i$ ($\beta_i \in \{0, 1\}$, and $\beta_i = \alpha_i, \forall i \neq r$) and internal state $S(q, \alpha_r)$ respectively. Then throughout this stage $A(t, \boldsymbol{\sigma}, \boldsymbol{\sigma}')$ equals 0 for all values of its arguments except as given by:

$$\begin{aligned} &A[t, (a', b', c', 1), (a, b, c, 0)] = 1, \forall a, b, c \in (\mathbf{Z}^+)^3, \\ &\quad \text{where } a' = R[a, P(\lfloor b/2^a \rfloor), c], \\ &\quad \text{and } b' = \sum_{i=0}^{\infty} \beta_i(a, b, c) \times 2^i, \\ &\quad \text{and } c' = S[c, P(\lfloor b/2^a \rfloor)]. \end{aligned}$$

Here P is the parity function: $P(x) = 1$ if x is odd, 0 if x is even.

As in the $t = 0 \longrightarrow t = 1$ stage, in this stage only those $K(t, \boldsymbol{\sigma})$ with $\sigma_4 = 1$ are altered by A. Therefore for all $\boldsymbol{\sigma}$ such that $\sigma_4 = 0, K(2, \boldsymbol{\sigma}) = K(0, \boldsymbol{\sigma})$.

Now evaluate $\sum_{\boldsymbol{\sigma}'} A(t, \boldsymbol{\sigma}, \boldsymbol{\sigma}') K(t, \boldsymbol{\sigma}')$ when $\sigma_4 = 1$. Due to A, only those $\boldsymbol{\sigma}'$ with $\sigma'_4 = 0$ terms will contribute. Therefore, letting $\mathbf{p}$ and $\mathbf{p}'$ be the first three components of $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma}'$ respectively, we have:

$$\partial_t K(\mathbf{p}, 1, t) = \sum_{\mathbf{p}'} A[t, (\mathbf{p}, 1), (\mathbf{p}', 0)] \; K[t, (\mathbf{p}', 0)].$$

Now throughout this stage, $K[t, (\mathbf{p}', 0)] = K[0, (\mathbf{p}', 0)]$. Therefore by assumption (3) (see above), throughout this stage there is a unique $\mathbf{p}^*$ such that

$$K[t, (\mathbf{p}^*, 0)] \neq 0.$$

Therefore throughout this stage,

$$\partial_t K[t, (\mathbf{p}, 1)] = A[t, (\mathbf{p}, 1), (\mathbf{p}^*, 0)].$$

Thus, $K[t, (\mathbf{p}, 1)]$ will change in this stage iff

$$\begin{aligned} p_1 &= R[p_1^*, P(\lfloor p_2^*/2^{p_1^*} \rfloor), p_3^*], \text{ and} \\ p_2 &= \sum_{i=0}^{\infty} \beta_i(p_1^*, p_2^*, p_3^*) \times 2^i, \text{ and} \\ p_3 &= S[p_3^*, P(\lfloor p_2^*/2^{p_1^*} \rfloor)]. \end{aligned}$$

Since $K[1,(\mathbf{p},1)] = 0$ for all $\mathbf{p}$, this means that at the end of this stage, $K[1,(\mathbf{p},1)] = 0$ for all $\mathbf{p}$ (just as at the end of the first stage) except for one: $K[2,(\mathbf{p},1)] = 1$ for the point $\mathbf{p}$ corresponding to the condition of the TM being emulated one iteration after it had the condition $\mathbf{p}^*$.

$\boxed{t = 2 \longrightarrow t = 3}$

This stage clears the $\sigma_4 = 0$ hyperplane leaving the $\sigma_4 = 1$ hyperplane alone, exactly as we previously cleared the $\sigma_4 = 1$ hyperplane leaving the $\sigma_4 = 0$ hyperplane alone.

In this stage, for all $\mathbf{p} \in (integer^+)^3$:

$$A[t,(\mathbf{p},0),(\mathbf{p},0)] = \begin{cases} 0 & \text{if } t = 3 \\ -1/(1-[t-2]) & \text{otherwise} \end{cases} .$$

During this stage $A(t)$ for all other values of its arguments equals 0.

$\boxed{t = 3 \longrightarrow t = 4}$

This stage copies the contents of the $\sigma_4 = 1$ hyperplane into the $\sigma_4 = 0$ hyperplane.

All $A(t,\boldsymbol{\sigma},\boldsymbol{\sigma}') = 0$ except $A[t,(\mathbf{p},0),(\mathbf{p},1)] = 1$. Given such an A, $K(t,\boldsymbol{\sigma})$ is unchanged during this stage for any $\boldsymbol{\sigma}$ with $\sigma_4 = 1$. Therefore

$$\partial_t K[t,(\mathbf{p},0)] = K[t,(\mathbf{p},1)] = K[3,(\mathbf{p},1)] = K[2,(\mathbf{p},1)].$$

Since for all $\mathbf{p}$, $K[3,(\mathbf{p},0)] = 0$, we know that if $K[3,(\mathbf{p},1)] = K[2,(\mathbf{p},1)] = 1$, then at the end of this stage $K[t,(\mathbf{p},0)] = 1$. Alternatively, if $K[3,(\mathbf{p},1)] = 0$, then at the end of this stage $K[t,(\mathbf{p},0)] = 0$. Therefore at the end of this stage the contents of the $\sigma_4 = 1$ hyperplane at $t = 2$ have been copied into the $\sigma_4 = 0$ hyperplane.

The cumulative effect of these four stages is to transform the contents of the $\sigma_4 = 0$ hyperplane in exact emulation of the transformation the TM undergoes during one iteration starting from the TM condition $\mathbf{p}^*$. Since assumptions (1) through (4) are valid at the end of the fourth stage, the four stages can be repeated and the dynamics will still be exactly emulating the TM. By induction, the dynamics always emulates the TM. **QED**.

As an aside, note the automatic correspondence between output flagging and how a TM halts. A TM halts when its internal state becomes the halt state. In the scheme recounted above, such a halt state corresponds to a particular value of r_3. So the system emulating the TM should "halt" when the support of f covers that particular value of r_3, i.e., output is flagged when the support of f covers the appropriate range in r_3.

As another aside, note that perhaps the most straightforward way in which a field computer can implement a universal TM is to simply encode a TM exactly as described previously in this section, where that encoded TM happens to be universal. To have such a universal-TM field computer emulate an arbitrary TM operating on an arbitrary input tape, that arbitrary TM's state-transition table together with the arbitrary tape is encoded in $f(0)$. The universal-TM field computer then transforms $f(0)$ in exact analogy to the way a conventional universal TM would transform a tape that coded for an arbitrary TM and arbitrary input tape for that TM.

4 Training, and Solving the Dynamics

For the system considered in this paper, training the system (finding a set of weights so that the net reproduces a particular training set), entails finding G such that when $f(0)$ corresponds to one of the inputs in the training set, then the signaled output

$$f(t \text{ when output signaling occurs})$$

codes for the corresponding output. Finding such a G is an ill-posed problem, of course; in any scheme for "training" G to reproduce a training set, some sort of regularizer is needed to uniquely fix G (otherwise one could simply use the preceding several sections to build an infinite number of distinct TMs, all of which reproduce the training set).

To help illuminate this regularization issue, it's worth making some cursory comments concerning the dynamics when G is not necessarily of the form given in Section 3. To that end, first consider a (spatially) discrete version of our system and its evolution equation, $\partial_t f(t, x_i) = \sum_j G(x_i, x_j) f(t, x_j)$, or equivalently, $\dot{f}(t) = G f(t)$, where G is the matrix with entries $G(x_i, x_j)$. This is just a set of simultaneous first order ordinary differential equations, with solution given by $f(t) = e^{tG} f(0)$. This suggests that the original continuum-version of our system evolves according to a one-dimensional (t) Lie group. In fact, the continuum-version of our system also has solution $f(t) = e^{tG} f(0)$.

To see this, it's useful to consider time-ordered products. Since G is independent of t, using our product notation we can write:

$$\partial_t \dot{f}(t) = \partial_t [G f(t)] = G \partial_t f(t) = G[G f(t)] = G^2 f(t),$$

where the field product $G^2(\mathbf{r}, \mathbf{r}') \equiv \int \mathrm{d}V'' G(\mathbf{r}, \mathbf{r}'') G(\mathbf{r}'', \mathbf{r}')$, the infinite-dimensional "matrix" G "squared". Therefore,

$$\ddot{f}(t) = G^2 f(t).$$

Continuing in this way, and making the assumption that $f(t, \mathbf{r})$ is analytic in t, we get the MacLauren series:

$$f(t) = f(0) + \sum_{n=1}^{\infty} t^n G^n f(0)/n!,$$

that is, using our product notation to define $G^n(\mathbf{r}, \mathbf{r}')$,

$$\begin{aligned} f(t, \mathbf{r}) &= f(0, \mathbf{r}) + \sum_{n=1}^{\infty} \left[t^n \int \mathrm{d}V' \; G^n(\mathbf{r}, \mathbf{r}') f(0, \mathbf{r}')/n! \right] \\ &= \int \mathrm{d}V' e^{tG}(\mathbf{r}, \mathbf{r}') f(0, \mathbf{r}'), \end{aligned}$$

if we identify $e^{tG}(\mathbf{r}, \mathbf{r}') \mid_{t=0}$ with the continuum version of the identity matrix, $\Delta(\mathbf{r}, \mathbf{r}')$. (The notation "$e^{tG}(\mathbf{r}, \mathbf{r}')$" indicates that the exponential is to be viewed as a function of $\mathbf{r}$ and $\mathbf{r}'$, which is parameterized by t.) As promised, we can write the preceding equation in product form as follows:[8]

$$f(t) = e^{tG} f(0).$$

QED

Since G determines f's dynamics by being the generating function of a one-dimensional Lie group giving $f(t)$, G does not have the power to induce arbitrary dynamics in f. Since G can force f to mimic an arbitrary Turing machine, the immediate corollary is that Turing machines can not obey arbitrary dynamics. This is not particularly surprising. As a trivial example, no Turing machine can return to the exact same state-tape configuration with which it started unless it returns to that configuration an infinite number of times.

There are a number of interesting special cases of the dynamics of our system. One occurs when G is translation invariant, so that $G(\mathbf{r}', \mathbf{r})$ can be written as $G(\mathbf{r} - \mathbf{r}')$ for all $\mathbf{r}, \mathbf{r}'$. In this case the evolution equation gives the time derivative of f in terms of the convolution of f with G. In such a situation it is straight-forward to solve for $f(t, \mathbf{r})$: simply take Fourier transforms of both sides of the evolution equation, use the convolution theorem, solve the resulting first order partial differential equation, and then take inverse Fourier transforms to get back $f(t, \mathbf{r})$.

Another interesting special case is where G is degenerate:

$$G(\mathbf{r}, \mathbf{r}') = \sum_i \phi_i(\mathbf{r}) \times \psi_i(\mathbf{r}').$$

For example, assume that $G(\mathbf{r}, \mathbf{r}') = \phi(\mathbf{r})\psi(\mathbf{r}')$. Write

$$\begin{aligned} f(t, \mathbf{r}) &= f(0, \mathbf{r}) + \int_0^t \mathrm{d}t' \int \mathrm{d}V' \; f(t', \mathbf{r}')\phi(\mathbf{r})\psi(\mathbf{r}') \\ &\equiv f(0, \mathbf{r}) + \phi(\mathbf{r}) \times B(t). \end{aligned}$$

If we can solve for $B(t)$, we can solve for the time-evolution of f. To solve for $B(t)$, substitute $f(t, \mathbf{r}) = f(0, \mathbf{r}) + \phi(\mathbf{r})A(t)$ into the definition of $B(t)$:

$$B(t) = \int_0^t \mathrm{d}t' \int \mathrm{d}V' \; [f(0, \mathbf{r}') + \phi(\mathbf{r}')A(t')]\psi(\mathbf{r}').$$

Now define

$$\begin{aligned} \alpha &\equiv \int \mathrm{d}V' \, f(0, \mathbf{r}')\psi(\mathbf{r}'), \\ \beta &\equiv \int \mathrm{d}V' \, \phi(\mathbf{r}')\psi(\mathbf{r}') = \int \mathrm{d}V' \, G(\mathbf{r}', \mathbf{r}'); \end{aligned}$$

our equation for $B(t)$ is $B(t) = \int_0^t \mathrm{d}t'[\alpha + \beta A(t')]$, which has solution $B(t) = (e^{\beta t} - 1)\alpha/\beta$.

As a final example of a special case, begin by considering the situation where f is separable, i.e., $f(t, \mathbf{r}) = u(t) \times \omega(\mathbf{r})$. Now redefine G so that the evolution equation becomes $-i \times \partial_t f(t, \mathbf{r}) = \int \mathrm{d}V' \; G(\mathbf{r}, \mathbf{r}')f(t, \mathbf{r}')$, that is, $-i\dot{f}(t) = Gf(t)$. Then in the usual way (see any introductory quantum mechanics text) one derives $u(t) \propto e^{i\lambda t}$, and $w(\mathbf{r})$ obeys the eigenvalue equation $\int \mathrm{d}V'\, G(\mathbf{r},\mathbf{r}')\omega(\mathbf{r}') = \lambda\omega(\mathbf{r})$, or $G\omega = \lambda\omega$. (Note that if G is Hermitian (which means our original, pre-redefinition G is anti-Hermitian), then $\lambda \in \mathbf{R}$, and $|f(t,\mathbf{r})|$ doesn't change in time.) Since our evolution equation is linear, if both $f(t,\mathbf{r}) = e^{i\lambda t}\omega_\lambda(\mathbf{r})$ and $f(t,\mathbf{r}) = e^{i\lambda' t}\omega_{\lambda'}(\mathbf{r})$ are solutions to our evolution equation then so is a linear combination of them, and in general we can write

$$f(t,\mathbf{r}) = \sum_\lambda f(\lambda) e^{i\lambda t}\omega_\lambda(\mathbf{r}),$$

where $\omega_\lambda(\mathbf{r})$ is a (normalized) solution to the equation $G\omega_\lambda = \lambda\omega_\lambda$ and the $f(\lambda)$ are expansion coefficients. So if G is indeed Hermitian, so that in general its eigenfunctions form a (necessarily orthonormal) complete basis, we can take $f(0,\mathbf{r})$ and use it to solve for $f(\lambda)$, and therefore for all $f(t,\mathbf{r})$. In other words, G Hermitian gives

$$f(t,\mathbf{r}) = \sum_\lambda \left[\int \mathrm{d}V'\ \omega_\lambda(\mathbf{r}')\ f(0,\mathbf{r}')\right] \times e^{i\lambda t} \times \omega_\lambda(\mathbf{r}).$$

5 Discussion

Our construction of Turing machines makes extensive use of delta functions, both in the field representation of the state $f(t)$ and in the construction of the kernel G. The use of such delta functions directly reflects the fact that we are using a field computer. Generically, for a system which evolves continuously in time but consists of discrete positions in $\mathbf{R}^5$, we don't need to use delta functions so extensively to get computational universality. Indeed, the purpose of Section 3.3 is essentially to reduce the original continuum-limit computer to such a lattice computer (see footnote (5)).

However such delta function fields are rarely (if ever) physically realizable, and so we must question the significance of the construction. On one hand it may be argued that since the Turing machine is an idealized model of computation, the use of delta functions is not problematic; physical realizability is not relevent to idealized mathematical models. The TM model itself makes physically unrealizable assumptions, such as the existence of a potentially infinite tape.

On the other hand the TM model is approximately realizable, but one may question whether our construction will work at all if the delta functions are replaced by physically realizable fields (i.e., bounded, continuous functions over compact domains). It is certainly conceivable that physically realizable replacements for the deltas would spread out over time until the state of the TM became indeterminate. We have not investigated whether such spreading can be directly prevented, or whether it would require the use of nonlinear sharpening functions in the evolution equation. Instead we present a different method for eliminating the deltas which seems more interesting.

Our solution hinges on the observation that the Fourier transform of a delta function is a complex exponential, and hence that delta functions in one domain correspond to sinusoids in the other. Therefore, instead of representing the state of the TM by a physically unrealizable field $f(t)$ in the spatial domain, we instead represent it by its Fourier transform $F(t) \equiv \mathcal{F}[f(t, \mathbf{r})]$, a physically realizable field defined over the spatial frequency domain. It is then necessary to show that the evolution equation $\dot{f}(t) = Gf(t)$ can be replaced by a corresponding evolution equation operating on spatial frequency fields:

$$\dot{F}(t) = HF(t).$$

With $\mathcal{F}$ representing the Fourier transform, since $\mathcal{F}[\dot{f}(t, \mathbf{r})] = \mathcal{F}[Gf(t, \mathbf{r})]$, $\partial_t \mathcal{F}[f(t, \mathbf{r})] = \mathcal{F}G\mathcal{F}^{-1}\mathcal{F}[f(t, \mathbf{r})]$, and we see that the required H is given by:

$$H = \mathcal{F}G\mathcal{F}^{-1}.$$

That is,

$$H(\boldsymbol{\omega}, \boldsymbol{\omega}') = \frac{1}{2\pi} \int \mathrm{d}V \int \mathrm{d}V' \, e^{i(\boldsymbol{\omega}\cdot\mathbf{r} - \boldsymbol{\omega}'\cdot\mathbf{r}')} G(\mathbf{r}, \mathbf{r}').$$

(To verify this, write $[HF(t)](\boldsymbol{\omega}) = \{1/2\pi\}^{3/2} \times \int \mathrm{d}\boldsymbol{\omega}' \mathrm{d}V \mathrm{d}V' \, e^{i(\boldsymbol{\omega}\cdot\mathbf{r} - \boldsymbol{\omega}'\cdot\mathbf{r}')} \times G(\mathbf{r}, \mathbf{r}') \times \int \mathrm{d}V'' \, e^{i\boldsymbol{\omega}'\cdot\mathbf{r}''} f(\mathbf{r}'')$, take the inverse Fourier transform with respect to $\boldsymbol{\omega}$, and then use the integrated plane wave definition of a Dirac delta function to arrive at $\int \mathrm{d}V' \, G(\mathbf{r}''', \mathbf{r}') \times f(\mathbf{r}')$.)

Note that transforming the problem into Fourier space integrates the delta functions out of both G and f. Of course, Fourier space as used here isn't a continuum-limit neural net. Rather it is a convenient way to emulate a continuum limit neural net, one which obeys the evolution Eq. 1, with another system which also obeys Eq. 1.

It is instructive to consider the form TM computation takes in the transformed domain. TM states which were represented by deltas (impulses) are instead represented by pure sinusoids (a superposition of two sinusoids in the buffer-copy stage). The specific configuration of the TM is reflected in the wave vector of the sinusoid ($\boldsymbol{\omega}$ or $\boldsymbol{\omega}'$ in the definition of H). Operation of the TM proceeds by a gradual "crossfading" from the wave representing the TM's current state to the wave representing its next state.

Finally we observe that since the fields $f(t)$ are nonzero only at the lattice points, we are in effect doing a *discrete* Fourier transform, and so the transformed fields $F(t)$ are periodic and can be represented on a compact domain (one period in extent in each dimension). Also, we can test for termination of the TM by various physically realizable operations, such as inner products and convolutions, on the transformed fields. For example, a test for termination by a nonzero product of the state with a mask field, $m(x) \times f(x) \neq 0$, can be accomplished in the frequency representation by testing for a a nonzero convolution, $[\mathcal{F}(m)](\omega) * [\mathcal{F}(f)](\omega) \neq 0$. In this way deltas in the mask field are replaced by sinusoids in the transformed field.

Clearly the use of the Fourier transform is not essential to this method; we could use any integral transform that converts deltas into physically realizable functions.

6 Acknowledgements

Some of this work was done under the auspices of the Department of Energy and the Santa Fe Institute. It was also supported in part by NLM grant F37 LM00011.

7 References

Blum, L. (1989). *Lectures on a theory of computation and complexity over the reals (or an arbitrary ring)* (Report No. TR-89-065) Berkeley, CA: International Computer Science Institute.

Blum, L., Shub, M., & Smale, S. (1988). On a theory of computation and complexity over the real numbers: NP completeness, recursive functions and universal machines. *The Bulletin of the American Mathematical Society*, **21**, 1–46.

Deutsch, D. (1985). Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London A*, **400**, 97–117.

Everett, H., III (1957). "Relative state" formulation of quantum mechanics. *Reviews of Modern Physics*, **29**, 454–462.

Franklin, S., & Garzon, M. (1990). Neural computability. In O. M. Omidvar (Ed.), *Progress in neural networks* (Vol. 1, pp. 127–145). Norwood, NJ: Ablex.

Garzon, M., & Franklin, S. (1989). Neural computability II (extended abstract). In *Proceedings, IJCNN International Joint Conference on Neural Networks* (Vol. 1, pp. 631–637). New York, NY: Institute of Electrical and Electronic Engineers.

Garzon, M., & Franklin, S. (1990). Computation on graphs. In O. M. Omidvar (Ed.), *Progress in neural networks* (Vol. 2, Ch. 13). Norwood, NJ: Ablex.

Hartley, R., & Szu, H. (1987). A comparison of the computational power of neural network models. In M. Caudill & C. Butler (Eds.), *Proceedings, IEEE First International Conference on Neural Networks* (Vol. 3, pp. 17–22). New York, NY: Institute of Electrical and Electronic Engineers.

Lloyd, S. (1990). *Any nonlinearity suffices for computation* (report CALT-68-1689). Pasadena, CA: California Institute of Technology.

MacLennan, B. J. (1987). Technology-independent design of neurocomputers: The universal field computer. In M. Caudill & C. Butler (Eds.), *Proceedings, IEEE First International Conference on Neural Networks* (Vol. 3, pp. 39–49). New York, NY: Institute of Electrical and Electronic Engineers.

MacLennan, B. J. (1990). *Field computation: A theoretical framework for massively parallel analog computation; parts I – IV* (report CS-90-100). Knoxville, TN: University of Tennessee, Computer Science Department. See also references therein.

MacLennan, B. J. (in press). Continuous symbol systems: The logic of connectionism. In Daniel S. Levine and Manuel Aparicio IV (Eds.), *Neural networks for knowledge representation and inference* (Ch. 4). Hillsdale, NJ: Lawrence Erlbaum.

MacLennan, B. J. (1993). Characteristics of connectionist knowledge representation. *Information Sciences*, **70**, 119–143.

McCulloch, W.S., and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, **5**, 115–133.

Omohundro, S. (1984). Modeling cellular automata with partial differential equations. *Physica 10D*, 128–134.

Pollack, J. B. (1987). *On connectionist models of natural language processing* (Ph.D. dissertation). Urbana, IL: University of Illinois; also report MCCS-87-100, Las Cruces, NM: New Mexico State University, Computing Research Laboratory.

Pour-El, M. B., & Richards, I. (1979). A computable ordinary differential equation which possesses no computable solution. *Annals of Mathematical Logic*, **17**, 61–90.

Pour-El, M. B., & Richards, I. (1981). The wave equation with computable initial data such that its unique solution is not computable. *Advances in Mathematics*, **39**, 215–239.

Pour-El, M. B., & Richards, I. (1982). Noncomputability in models of physical phenomena. *International Journal of Theoretical Physics*, **21**, 553–555.

Rumelhart, D. E., McClelland, J. L., & the PDP Research Group (1986). *Parallel distributed processing: Explorations in the microstructure of cognition.* Cambridge, MA: MIT Press.

Stannett, M. (1990). X-machines and the halting problem: Building a super-Turing machine. *Formal Aspects of Computing*, **2**, 331–341.

VanHeyningen, M. D., & MacLennan, B. J. (1992). *A constraint satisfaction model for perception of ambiguous stimuli* (report CS-92-152). Knoxville, TN: University of Tennessee, Computer Science Department.

Wolpert, D. H. (1990). A mathematical theory of generalization: Part II. *Complex Systems*, **4**, 201–249.

8 Footnotes

1. The *support* of $f(t)$ is the closure of the set of all $\mathbf{r}$ such that $f(t, \mathbf{r}) \neq 0$. A more general formulation might interpret the state of $f(t)$ in terms of those $\mathbf{r} \in \mathbf{R}^n$ such that $f(t, \mathbf{r}) >$ some threshold ϵ, rather than in terms of the support of $f(t)$. For simplicity, such a formulation is not followed in this paper.

2. It should be noted that there are other ways to flag output besides having the support of $f(t)$ cover a predefined output-flagging region. For example, one might have output flagged when $f(t)$ gets sufficiently peaked. If output is flagged this way, one might want to change the way that the distribution $f(t)$ is assigned meaning, from the meaning being given by the support of $f(t)$ to perhaps something like the meaning being given by the average (according to the distribution $f(t)$) of $\mathbf{r}$. No such alternate scheme for flagging output and assigning meaning to $f(t)$ is considered in this paper. For examples of using output-flagging, in real programs, see Wolpert (1990).

3. Note that this minus term has the perhaps annoying property that, everything else being equal, the more confident we are in a point (i.e., the larger $f(t, \mathbf{r})$), the more quickly we lower our confidence in that point. This might not be such a bad thing — amongst other things, it should help keep behavior stable. It also has the property that if a decision is not reached in spite of the high confidence in a hypothesis, then confidence will gradually leak away from that hypothesis and be transferred to others. In effect the dynamics says, "If that isn't working, try something new." Such a mechanism could help prevent a cognitive processor from becoming locked into unproductive hypotheses, and may help explain multistability in perception (VanHeyningen & MacLennan, 1992). Nonetheless, one might wish to modify it somehow.

4. In this regard, it's worth noting that in Section 4 it's shown that the dynamics of f is intimately connected with the formalism of quantum mechanics. In light of the various quantum mechanical justifications for assigning meaning to the square of the wave function rather than to the wave function itself (e.g., Everett, 1957), this suggests that one might want to assign meaning to

the square of f rather than to f itself. This has no implication for the case where it's the support of f which carries meaning, but it does have implications for the confidence-level interpretation of f. In particular, using the square of f rather than f itself removes the issue of assigning meaning to "negative confidence." It also means that we would be led to replace Eq. 3 with an equation preserving the L_2 norm of f rather than the L_1 norm of f. (In this regard, note that the L_2 norm is more convenient mathematically than the L_1 norm.)

5. We note in passing that the discrete time emulation $\mathbf{s}' = W\mathbf{s}$ could use any infinite orthogonal basis $e_1, e_1, e_2, \ldots$ in any space to represent the states of the TM. Moreover, if such a basis spans a space of functions over a compact domain, e.g. $L_2([0,1])$, and the basis functions are continuous, then the representation will satisfy the conditions for the physical realizability of fields set down in MacLennan (1990). Furthermore, if the representation is chosen so that the longer tapes correspond to higher frequency basis functions, then the resource limitations of physically realizable TMs will correspond to the bandwidth limitations of the medium supporting the fields. See also MacLennan (**??**, 1993).

6. Note that this means, for example, that we can recast the problem of finding a G to reproduce a provided training set as the problem of finding a potential which evolves one set of wavefunctions into another set of wavefunctions. This is nothing other than a quantum mechanical scattering problem! Such problems have been studied intensively for decades. Exploiting this, one way to find a G to reproduce a provided training set (i.e., provided scattering data) is to assume that there are a discrete number of scattering objects and solve for their positions (just as in X-ray diffraction).

7. Note that since our new discrete evolution equation holds regardless of the spacing of the lattice, we can take that spacing $\longrightarrow 0$. This means that without loss of generality, we could have written our original evolution equation as $\partial_t f(t, \mathbf{r}) = \int \mathrm{d}V'\, G(t, \mathbf{r}, \mathbf{r}') f(t, \mathbf{r}')$, or $\dot{f}(t) = G(t) f(t)$, with a time-dependent kernel, where now $\mathbf{r} \in \mathbf{R}^4$ rather than $\mathbf{r} \in \mathbf{R}^5$. In other words, at the expense of losing a dimension in $\mathbf{r}$, we can replace a t-independent G with a t-dependent one.

8. To check this formula, we can differentiate with respect to t:

$$
\begin{aligned}
\partial_t f(t,\mathbf{r}) &= \partial_t \int \mathrm{d}V'\ e^{tG}(\mathbf{r},\mathbf{r}')f(0,\mathbf{r}') \\
&= \int \mathrm{d}V'\ f(0,\mathbf{r}')\,\partial_t(e^{tG})(\mathbf{r},\mathbf{r}') \\
&= \int \mathrm{d}V'\ f(0,\mathbf{r}')(Ge^{tG})(\mathbf{r},\mathbf{r}') \\
&= \int \mathrm{d}V' \left\{ f(0,\mathbf{r}') \int \mathrm{d}V''\ G(\mathbf{r},\mathbf{r}'') \times e^{tG}(\mathbf{r}'',\mathbf{r}') \right\} \\
&= \int \mathrm{d}V'' \left\{ G(\mathbf{r},\mathbf{r}'') \int \mathrm{d}V'\ f(0,\mathbf{r}') \times e^{tG}(\mathbf{r}'',\mathbf{r}') \right\} \\
&= \int \mathrm{d}V''\ G(\mathbf{r},\mathbf{r}'')f(t,\mathbf{r}'').
\end{aligned}
$$