# Technology-Independent Design of Neurocomputers: The Universal Field Computer

Bruce J. MacLennan
Computer Science Department
Naval Postgraduate School
Monterey, CA 93943

Published in the Proceedings of the IEEE First Annual International Conference on Neural Networks, June 1987.

IEEE Catalog #87TH0191-7

# Technology-Independent Design of Neurocomputers: The Universal Field Computer[1]

Bruce J. MacLennan
Computer Science Department
Naval Postgraduate School
Monterey, CA 93943

## Abstract

We argue that AI is moving into a new phase characterized by biological rather than psychological metaphors. Full exploitation of this new paradigm will require a new class of computers characterized by massive parallelism: parallelism in which the number of computational units is so large it can be treated as a continuous quantity. We suggest that this leads to a new model of computation based on the transformation of continuous scalar and vector fields. We describe a class of computers, called *field computers* that conform to this model, and claim that they can be implemented in a variety of technologies (e.g., optical, artificial neural network, molecular). We also describe a *universal field computer* and show that it can be programmed for the parallel computation of a wide variety of field transformations.

## 1. The "New" AI

Traditional Artificial Intelligence technology is based on *psychological metaphors,* that is, idealized models of human cognitive behavior. In particular, models of conscious, goal-directed problem solving have provided the basis for many of AI's accomplishments to date. As valuable as these metaphors have been, we believe that they are not appropriate for many of the tasks for which we wish to use computers. In particular, symbolic information processing does not seem to be a good model of the way people (or animals) behave skillfully in subcognitive tasks, such as pattern recognition and sensorimotor coordination. Thus, the needs of these applications are driving Artificial Intelligence into a new phase characterized by *biological* metaphors. We call this phase, characterized by a combination of symbolic and non-symbolic processing, *the "new" AI* (MacLennan, in press). The technology of the new AI already includes neural information processing, genetic algorithms, and simulated annealing. The new AI will allow us to make use of massively parallel computers (including neurocomputers), optical computers, molecular computation, and, we expect, a new generation of analog computers.

Current AI technology has been quite successful in a number of tasks, for example, chess, diagnosis of blood diseases and theorem proving. Many other tasks remain beyond its capabilities, including face recognition, autonomous movement and continuous speech recognition. The interesting thing is that the tasks that AI has been most successful with are those that we commonly consider higher cognitive activities, specifically, those activities that can be performed by humans, but by few other animals. On the other hand, the tasks that currently stretch the capabilities of AI technology are those that are lower on the scale of cognitive accomplishment. Specifically, they are activities that almost any animal can perform with skill. A rodent may not be able to prove theorems, but it can effectively navigate its way through a complicated terrain, avoid predators, and find food – and accomplish this with a comparatively small brain constructed of comparatively slow devices. It has been truly said that computers will replace mathematicians long before they will replace carpenters. Unfortunately, many important applications of artificial intelligence require just the sort of activities that stretch the current technology. Therefore it is important to seek the reason for the limitations of the current technology, and to see if there is a way around them.

Current AI technology is based on psychological metaphors; its algorithms mimic conscious, rational thought. Thus, this technology deals best with verbalizable knowledge (knowledge *that*), deductive reasoning and discrete categories. However, as we've seen, there are other kind of intelligent behavior.

1. The research reported herein was supported by the **Office of Naval Research** under contract N00014-87-WR-24037. Author's address after October, 1987: Computer Science Department, University of Tennessee, Knoxville, Tenn. 37996-1301.

In the past AI has attempted to reduce all intelligent activity to *intellectual* activity; the computer is in effect a disembodied brain. We claim that AI is entering a new phase that recognizes the role of the body in intelligent behavior, and that emphasizes unconscious, tacit knowledge – what we might call *skillful* behavior, as opposed to *knowledge-based* behavior. This new phase attempts to come to grips with such problems as unverbalized knowledge (knowledge *how*), immediate perception, sensorimotor coordination, approximate and context-sensitive categorization, and everyday (as opposed to intellectual) behavior. The new AI is characterized by a greater use of *biological* (as opposed to psychological) metaphors. Harbingers of the new AI include the recent research activity in neurocomputation, genetic algorithms, cellular architectures and molecular computation. In this paper we present techniques by which these nontraditional computers can be designed independently of their implementation technology.

## 2. Field Transformation Computers

### 2.1 Massive Parallelism

Many of the newer computational paradigms are characterized by the processing of massive amounts of data in parallel. For example, in neurocomputers and Boltzmann machines (Hinton and Sejnowski, 1983) large numbers of simple processing elements compute in parallel. Similarly, some optical computers process in parallel the elements of an optical wavefront. A key advantage of molecular computers will be the ability of large numbers of molecules to operate in parallel. Consideration of new computing paradigms such as these leads us to offer the following definition of massive parallelism:[2]

**Definition (Massive Parallelism):** A computational system is *massively parallel* if the number of processing elements is so large that it may conveniently be considered a continuous quantity.

Of course, this definition admits borderline cases. For most purposes, a million processors will qualify, but 16 will not. In some circumstances as few as a thousand may be sufficient.

Why is it relevant that the number of processors can be taken as a continuous quantity? One reason is that for some kinds of massively parallel computers the number of processors is in fact continuous, or nearly so. Examples are optical and molecular computers. You don't *count* $10^{20}$ processors; you *measure* their quantity in terms of some macroscopic unit. The second reason for seeking continuity is that the mathematics is simpler. When the number of processing elements is very large, statistical methods can often be applied. Also, continuous mathematics (such as the infinitesmal calculus) can be applied, which is much more tractable than discrete mathematics (e.g. combinatorics).

Under our definition of massive parallelism, it doesn't matter whether the implementation technology is in fact discrete or continuous (or nearly so, as in molecular computing). In either case the design of the computer can be described by continuous mathematics. Then, if the intended implementation technology is discrete, we can select out of the continuum of points a sufficiently large finite number. This selection may be either regular (e.g. in a grid) or random (subject only to statistical constraints). In this way much of the design of massively parallel computers can be accomplished independently of the implementation technology.

### 2.2 Field Transformation

Given our definition of massive parallelism, it is clear that the processing elements of a massively parallel computer cannot be individually programmed; they must be controlled *en masse*. How can this be done?

We suggest that the operation of massively parallel computers is best thought of as *field processing*. That is, we think of a very large aggregation of data as forming a continuous (scalar or vector) *field* (analogous to an electrical field). The individual processing steps operate on entire fields to yield entire fields. Since a continuum of data is transformed in parallel, we achieve massive parallelism. A simple example is an optical convolution, which operates on an entire optical field in parallel.

---

2. Perhaps *infinite* or *continuous* parallelism would be a better term.

Conventional digital (and analog) computers perform *point processing,* that is, they operate on one (or a few) points at a time. We suggest that the full benefit of massive parallelism will be achieved by *field processing,* the parallel transformation of entire fields of data. (The distinction between point processing and field processing is analogous to that between word-at-a-time and vector processing in functional programming; see Backus, 1978.) In the remainder of this section we discuss *field transformation computers:* computers designed for field processing.

### 2.3 Classes of Field Transformations

There are two classes of field transformations: *nonrecursive* and *recursive* (or *functional* and *temporal*).

In nonrecursive processing, fields are passed through various transforms and are combined with one another to yield an output field; there may be feed-forward but no feed-back. Nonrecursive transformation applies a (perhaps complex) function to its input fields to yield its output fields. The input-output dependency is functional: same inputs, same outputs.

Recursive processing is like nonrecursive except that there is feed-back. Therefore the fields evolve in time according to the differential equations describing the system. The output of a recursive transform depends on its inputs *and* on its current state.

We expect field computers to permit elementary field transforms to be connected in a variety of ways to yield more complex recursive and nonrecursive field transforms. We also expect field computers to permit limited point processing. Scalar values are often useful as global parameters for controlling field processing operations. For example, the average light intensity of a scene (a scalar) might be used to control a field transformation for contrast enhancement. Point processing can also be used for controlling the thresholds of large numbers of neural units (e.g., in simulated annealing; see Kirkpatrick et al.), or for determining global reaction parameters for molecular processes.

Many field processing tasks will depend on a number of *fixed* or *constant fields* that must be properly initialized. There are a number of sources for these fixed fields. For example, they may be computed by another field transformation process and loaded into read-only memories. Fixed fields can also be generated by *training processes,* which build them up by recursive field processing. Finally, fixed fields can be modified adaptively as the system runs, in which case they are only *relatively* fixed (i.e., they change at a much slower rate than the *variable fields).*

### 2.4 General Purpose Field Computers

We can imagine implementing various recursive and nonrecursive field processing systems by assembling the appropriate elementary field transforms. We expect that many *special purpose* field computers will be implemented in just this way (indeed, some already are).

On the other hand, the flexibility of *general purpose* digital computers has shown us the value of programmability. In these the connection of the processing elements is transitory and under the control of an easily alterable program. Is it possible to design a *general purpose field computer,* that is, a field computer that can be programmed to emulate any other field computer? We argue that it is, and much of the rest of this paper is in pursuit of this goal.

What would a general purpose field computer be like? We expect that it would have a number of *field storage units,* of various dimensionalities, for holding (bounded) scalar and vector fields. Some of these would hold fixed fields for controlling the processing. Others would hold variable fields (1) captured from input devices, or (2) to be presented to output devices, or (3) as intermediate fields in recursive processes. There would also be some scalar registers.

Field transformation processes would be implemented by programmed connections between elementary field transforms. These elementary operations should permit programming any useful field transformation in a modest number of steps. Note that we are not *too* concerned about the number of steps, since each processes in parallel a massive amount of data. Some of the elementary transforms may be sensitive to scalar parameters, thus permitting global control.

Is it possible to find a set of elementary transforms that can be assembled to yield any useful field transformation? This is exactly what we establish in the next section. We show how a limited variety of processing units can be assembled to compute almost any field transformation to any desired accuracy. Of course, the more accuracy we want, the more units it will take, but that is acceptable. What is not acceptable is to replace field processing by point processing. To do so would be completely impractical: you can't do $10^{20}$ operations serially. Thus we must identify a universal set of field transforms in terms of which all others can be implemented.

## 3. A Universal Field Computer

### 3.1 Introduction

The value of the Turing machine as a model of digital computation is that it allows establishing the limitations and capabilities of discrete symbol processing. In particular, the *universal* Turing machine establishes the possibility of *general purpose* digital computers. On the other hand, the universal Turing machine is an idealization; it has the minimum capabilities required to compute all computable functions, so it is much less efficient than real computers. Real computers extend the facilities of the universal Turing machine for the sake of practical (efficient) computation. In this section we outline an analogous idealized model of computation for massively parallel and analog computation, that is, for field computers. This *universal field computer* is capable of implementing "any" function defined on fields. Of course, there are some limitations on the functions that can be so computed, just as there are limitations on the functions that can be computed by Turing machines. We claim that the class of implementable functions is sufficiently broad to include all those required for practical applications. Also, we expect that real (practical) general purpose field computers will provide more than this minimum of facilities.

There are a number of ways we might design a universal field computer, just as there are many alternatives to the universal Turing machine that compute the same class of functions. Fourier analysis and interpolation theory both suggest ways of implementing arbitrary functions in terms of a limited class of primitives. In the rest of this section we explore a particular approach, based on an extension of Taylor's Theorem to field transformations.

### 3.2 Taylor Series Approximation of Field Transforms

In this section we develop the basic theory of functions on scalar and vector fields and of their approximation by Taylor series. Once it is understood that fields are treated as continuous-dimensional vectors, it will seen that the mathematics is essentially that of finite-dimensional vectors. Thus the treatment here is heuristic rather than rigorous. First we consider scalar fields; later we turn to vector fields.

As usual we take a scalar field to be a function $\phi$ from an underlying set $\Omega$ to an algebraic field $K$, thus $\phi$: $\Omega \rightarrow K$. For our purposes $K$ will be the field of real numbers, $\mathbf{R}$. We use the notation $\Phi(\Omega)$ for the set of all scalar fields over the underlying set $\Omega$ ($K = \mathbf{R}$ being understood). Thus, $\Phi(\Omega)$ is a function space, and in fact a linear space under the following definitions of field sum and scalar product:

$$\begin{aligned} (\phi + \psi)_t &= \phi_t + \psi_t \\ (\lambda\phi)_t &= \lambda(\phi_t) \end{aligned} \tag{1}$$

Note that we often write $\phi_t$ for $\phi(t)$, the value of the field at the point $t$. As a basis for this linear space we take the unit functions $\omega_t$ for each $t \in \Omega$. They are defined

$$\begin{aligned} \omega_t(t) &= 1 \\ \omega_t(s) &= 0, \text{ if } s \neq t \end{aligned} \tag{2}$$

The preceding definitions show that we can think of scalar fields as vectors over the set $\Omega$. Since we want to be quite general, we assume only that $\Omega$ is a measurable space. In practice, it will usually be a closed and bounded subspace of $E^n$, $n$-dimensional Euclidean space. Thus we typically have one, two and three dimensional closed and bounded scalar fields.

Since $\Omega$ is a measure space, we can define an inner product between scalar fields:

$$\phi \cdot \psi \equiv \int_\Omega \phi_t \psi_t \, dt. \tag{3}$$

We also define the norm:

$$||\phi|| = \int_\Omega |\phi_t| \, dt. \tag{4}$$

Thus $\Phi(\Omega)$ is the function space $L_1(\Omega)$. Note that the $\omega_t$ are not an orthogonal set under this norm, since $||\omega_t|| = 0$.

We first consider scalar valued functions of scalar fields, that is functions $f$: $\Phi(\Omega) \to \mathbf{R}$. We prove some basic properties of these functions, culminating in Taylor's theorem.

**Definition (Differentiability):** Suppose $f$ is a scalar valued function of scalar fields, $f$: $\Phi(\Omega) \to \mathbf{R}$, and that $f$ is defined on a neighborhood of $\phi \in \Phi(\Omega)$. Then we say that $f$ is ***differentiable at*** $\phi$ if there is a field $D \in \Phi(\Omega)$ such that for all $\alpha \in \Phi(\Omega)$

$$f(\phi + \alpha) - f(\phi) = \alpha \cdot D + \eta ||\alpha|| \tag{5}$$

where $\eta \to 0$ as $||\alpha|| \to 0$.

**Theorem:** If $f$ is differentiable at $\phi$ then $f$ is continuous at $\phi$.

*Proof:* Since $f$ is differentiable at $\phi$ we know

$$f(\psi) - f(\phi) = (\psi - \phi) \cdot D + \eta ||\psi - \phi||.$$

Therefore,

$$\begin{aligned} |f(\psi) - f(\phi)| &= |(\psi - \phi) \cdot D + \eta ||\psi - \phi|| | \\ &\leqslant |(\psi - \phi) \cdot D| + |\eta| \, ||\psi - \phi|| \\ &\leqslant ||D|| \, ||\psi - \phi|| + |\eta| \, ||\psi - \phi|| \\ &= (||D|| + |\eta|) \, ||\psi - \phi||. \end{aligned}$$

Thus $f$ is continuous at $\phi$. ■

Since our "vectors" are continuous dimensional, partial derivatives are with respect to a "coordinate" $t \in \Omega$ rather than with respect to a coordinate variable. To accomplish this it's convenient to make use of the Dirac delta functions:

$$\begin{aligned} \delta_t(t) &= \infty \\ \delta_t(s) &= 0, \text{ for } s \neq t \end{aligned} \tag{6}$$

Of course, by the first equation above we mean $\delta_t(s) = \lim_{\epsilon \to 0} \epsilon^{-1}$ for $|s - t| < \epsilon/2$. Note the following properties of the delta functions (fields):

$$\begin{aligned} ||\delta_t|| &= 1 \\ \delta_t \cdot \phi &= \phi_t \end{aligned} \tag{7}$$

**Definition (Partial Derivative):** The partial derivative, at coordinate $t \in \Omega$, of $f$: $\Phi(\Omega) \to \mathbf{R}$, evaluated at $\phi$, is defined:

$$\frac{\partial}{\partial \delta_t} f(\phi) = \lim_{h \to 0} \frac{f(\phi + h\delta_t) - f(\phi)}{h}. \tag{8}$$

**Theorem:** If $f$ is differentiable at $\phi$ then the first order partial derivatives exist at $\phi$.

*Proof:* First observe that by differentiability

$$\begin{aligned}\frac{f(\phi + h\delta_t) - f(\phi)}{h} &= \frac{f(\phi) + h\delta_t \cdot D + \eta||h\delta_t|| - f(\phi)}{h} \\ &= \delta_t \cdot D + \eta||\delta_t||\,|h|/h \\ &= \delta_t \cdot D + \eta|h|/h \\ &= D_t + \eta|h|/h\end{aligned}$$

Recalling that $\eta \to 0$ as $h \to 0$, observe

$$\begin{aligned}\lim_{h\to 0}\left|\frac{f(\phi + h\delta_t) - f(\phi)}{h} - D_t\right| &= \lim_{h\to 0}|D_t + \eta|h|/h - D_t| \\ &= \lim_{h\to 0}|\eta| \\ &= 0\end{aligned}$$

Hence, $\frac{\partial}{\partial\delta_t}f(\phi) = D_t$, where $D$ is the field whose existence is guaranteed by differentiability. Thus the partial derivative exists. ■

What is the field $D$ whose points are the partial derivatives? It is just the gradient of the function.

**Definition (Gradient):** the gradient of $f(\phi)$ is a field whose value at a point $t$ is the partial derivative at that point, $\frac{\partial}{\partial\delta_t}f(\phi)$:

$$[\nabla f(\phi)]_t = \frac{\partial}{\partial\delta_t}f(\phi). \tag{9}$$

The gradient can also be expressed in terms of the basis functions:

$$\nabla f(\phi) = \int_\Omega \omega_t \frac{\partial}{\partial\delta_t}f(\phi)\,dt. \tag{10}$$

When no confusion will result, we use the following operator notations:

$$\begin{aligned}\nabla f &= \int_\Omega \omega_t\,\partial f/\partial\delta_t\,dt \\ \nabla &= \int_\Omega \omega_t\,\partial/\partial\delta_t\,dt \\ \partial/\partial\delta_t &= \delta_t \cdot \nabla\end{aligned} \tag{11}$$

Note that by the definitions of differentiability (Eq. 5) and the gradient (Eq. 9) we have that

$$f(\phi + \alpha) - f(\phi) = \alpha \cdot \nabla f(\phi) + \eta||\alpha||, \tag{12}$$

where $\eta \to 0$ as $||\alpha|| \to 0$. This leads to the concept of a directional derivative.

**Definition (Directional Derivative):** The directional derivative in the direction $\alpha$ is given by the following formulas (shown in both explicit and operator forms):

$$\begin{aligned}\nabla_\alpha f(\phi) &= \alpha \cdot \nabla f(\phi) = \int_\Omega \alpha_t \frac{\partial}{\partial\delta_t}f(\phi)\,dt \\ \nabla_\alpha &= \alpha \cdot \nabla = \int_\Omega \alpha_t \partial/\partial\delta_t\,dt\end{aligned} \tag{13}$$

Note that the notation is accurate in that $(\alpha \cdot \nabla)f(\phi) = \alpha \cdot [\nabla f(\phi)]$. Also note that $\partial/\partial\delta_t = \nabla_{\delta_t}$.

**Lemma:** If $f$ is differentiable in a neighborhood of $\phi$, then

$$\frac{d}{dx}f(\phi + x\alpha) = \alpha \cdot \nabla f(\phi + x\alpha). \tag{14}$$

*Proof:* By the definition of the derivative:

$$\begin{aligned}
\frac{d}{dx} f(\phi + x\alpha) &= \lim_{h\to 0} \frac{f[\phi + (x+h)\alpha] - f(\phi + x\alpha)}{h} \\
&= \lim_{h\to 0} \frac{f(\phi + x\alpha + h\alpha) - f(\phi + x\alpha)}{h} \\
&= \lim_{h\to 0} \frac{h\alpha \cdot \nabla f(\phi + x\alpha) + \eta||h\alpha||}{h} \\
&= \lim_{h\to 0} \alpha \cdot \nabla f(\phi + x\alpha) + \eta||\alpha|| \, |h| \, /h \\
&= \alpha \cdot \nabla f(\phi + x\alpha)
\end{aligned}$$

since $\eta \to 0$ as $|h| \to 0$. ■

**Theorem (Mean Value):** Suppose $f$: $\Phi(\Omega) \to \mathbf{R}$ is continuous on a neighborhood containing $\phi$ and $\psi$. Then, there is a $\theta$, $0 \leqslant \theta \leqslant 1$, such that

$$\begin{aligned}
f(\psi) - f(\phi) &= (\psi - \phi) \cdot \nabla f(\chi) \\
\text{where } \chi &= \phi + \theta(\psi - \phi)
\end{aligned} \tag{15}$$

*Proof:* Let $\alpha = \psi - \phi$ and consider the function

$$F(x) = f(\phi + x\alpha) - f(\phi) - x[f(\psi) - f(\phi)].$$

Since $f$ is continuous, so is $F$. Now, since $F(0) = F(1) = 0$, we have by Rolle's Theorem that there is a $\theta$, $0 \leqslant \theta \leqslant 1$, such that $F'(\theta) = 0$. Note that

$$\begin{aligned}
F'(x) &= \frac{d}{dx}\{f(\phi + x\alpha) - f(\phi) - x[f(\psi) - f(\phi)]\} \\
&= \frac{d}{dx} f(\phi + x\alpha) - [f(\psi) - f(\phi)] \, .
\end{aligned}$$

By the preceding lemma

$$F'(x) = \alpha \cdot \nabla f(\phi + x\alpha) - [f(\psi) - f(\phi)]$$

Hence, substituting $\theta$ for $x$,

$$0 = F'(\theta) = \alpha \cdot \nabla f(\phi + \theta\alpha) - [f(\psi) - f(\phi)].$$

Therefore, transposing we have

$$f(\psi) - f(\phi) = \alpha \cdot \nabla f(\phi + \theta\alpha)$$

and the theorem is proved. ■

**Theorem (Taylor):** Suppose that $f$ and all its partial derivatives through order $n+1$ are continuous in a neighborhood of $\phi$. Then for all $\alpha$ such that $\phi + \alpha$ is in that neighborhood there is a $\theta$, $0 \leqslant \theta \leqslant 1$, such that

$$f(\phi + \alpha) = f(\phi) + \nabla_\alpha f(\phi) + \frac{1}{2}\nabla_\alpha^2 f(\phi) + \cdots + \frac{1}{n!}\nabla_\alpha^n f(\phi) + \frac{1}{(n+1)!}\nabla_\alpha^{n+1} f(\phi + \theta\alpha). \tag{16}$$

*Proof:* By the Taylor theorem on real variables,

$$f(\phi + t\alpha) = f(\phi) + d/dt\ f(\phi)t + \frac{1}{2}d^2/dt^2\ f(\phi)t^2 + \cdots + \frac{1}{n!}d^n/dt^n\ f(\phi)t^n + \frac{1}{(n+1)!}d^{n+1}/dt^{n+1}\ f(\phi + \theta\alpha)t^{n+1}.$$

Observe that by the preceding lemma

$$\frac{d^n}{dt^n}\ f(\phi + t\alpha) = \nabla_\alpha^n f(\phi + t\alpha).$$

Therefore,

$$f(\phi + t\alpha) = f(\phi) + \nabla_\alpha f(\phi)t + \frac{1}{2}\nabla_\alpha^2 f(\phi)t^2 + \cdots + \frac{1}{n!}\nabla_\alpha^n f(\phi)t^n + \frac{1}{(n+1)!}\nabla_\alpha^{n+1} f(\phi + \theta\alpha)t^{n+1}.$$

Setting $t = 1$ gives the desired result. ■
The extension to a function of *several* scalar fields is routine.

### 3.3 A Universal Field Computer Based on Taylor Series Approximation

We can use Taylor's Theorem to derive approximations of quite a general class of scalar valued functions of scalar fields. Thus, if we equip our universal field computer with the hardware necessary to compute Taylor series approximations, then we will be able to compute any of a wide class of functions (namely, those functions whose first $n$ partial derivatives exist and are continuous). Therefore, consider the general form of an $n$-term Taylor series:

$$f(\phi) = \sum_{k=1}^{n} \frac{1}{k!}\nabla_\alpha^k f(\phi_0), \quad \text{where } \alpha = \phi - \phi_0 \tag{17}$$

What hardware is required? Clearly we will need a field subtractor for computing the difference field $\alpha = \phi - \phi_0$. We will also need a scalar multiplier for scaling each term by $1/k!$; we will also need a scalar adder for adding the terms together. The harder problem is to find a way to compute $\nabla_\alpha^k f(\phi_0)$ for a vector $\alpha$ that depends on the (unknown) input $\phi$. The trouble is that the $\alpha$s and the $\nabla$s are interleaved, as can be seen here:

$$\begin{aligned}
\nabla_\alpha^k f(\phi_0) &= (\alpha \cdot \nabla)^k f(\phi_0) \\
&= (\alpha \cdot \nabla)^{k-1} [\alpha \cdot \nabla f(\phi_0)] \\
&= (\alpha \cdot \nabla)^{k-1} \int_\Omega \alpha_{t_1} \frac{\partial}{\partial \delta_{t_1}} f(\phi_0)\ dt_1 \\
&\vdots \\
&= \int_\Omega \cdots \int_\Omega \int_\Omega \alpha_{t_1}\alpha_{t_2} \cdots \alpha_{t_k} \frac{\partial^k}{\partial \delta_{t_1}\partial \delta_{t_2} \cdots \partial \delta_{t_k}} f(\phi_0)\ dt_1\ dt_2 \cdots dt_k
\end{aligned}$$

We want to separate everything that depends on $\alpha$, and is thus variable, from everything that depends on $f(\phi_0)$, and is thus fixed. This can be accomplished (albeit, with extravagant use of our dimensional resources) by means of an outer product operation. Therefore we define the outer product of two scalar fields:

$$(\phi \perp \psi)_{s,t} = \phi_s \psi_t \tag{18}$$

Note that if $\phi, \psi \in \Phi(\Omega)$ then $\phi \perp \psi \in \Phi(\Omega^2)$.

To see how the outer product allows the variable and fixed parts to be separated, consider first the case $\nabla_\alpha^2$:

$$
\begin{aligned}
\nabla_\alpha^2 f(\phi_0) &= \int_\Omega\int_\Omega \alpha_s\alpha_t \frac{\partial}{\partial\delta_s}\frac{\partial}{\partial\delta_t} f(\phi_0)\, dt\, ds \\
&= \int_\Omega\int_\Omega (\alpha \perp \alpha)_{s,t} (\nabla)_s(\nabla)_t f(\phi_0)\, dt\, ds \\
&= \int_\Omega\int_\Omega (\alpha \perp \alpha)_{s,t} (\nabla \perp \nabla)_{s,t} f(\phi_0)\, dt\, ds \\
&= \int_{\Omega^2} (\alpha \perp \alpha)_z (\nabla \perp \nabla)_z dx\, f(\phi_0) \\
&= (\alpha \perp \alpha)\cdot(\nabla \perp \nabla) f(\phi_0)
\end{aligned}
$$

Now we can see how the general case goes. First we define the $k$-fold outer product:

$$
\begin{aligned}
\phi^{[1]} &= \phi \\
\phi^{[k+1]} &= \phi \perp \phi^{[k]}
\end{aligned}
\tag{19}
$$

Then,

$$
\nabla_\alpha^k f(\phi) = \alpha^{[k]}\cdot\nabla^{[k]} f(\phi) \tag{20}
$$

The $n$-term Taylor series then becomes

$$
f(\phi) = \sum_{k=1}^{n} \frac{1}{k!}(\phi-\phi_0)^{[k]}\cdot\nabla^{[k]} f(\phi_0) \tag{21}
$$

Since $\phi_0$ is fixed, we can compute each $\nabla^{[k]} f(\phi_0)$ once, when the field computer is programmed. Then, for any given input $\phi$ we can compute $(\phi-\phi_0)^{[k]}$ and take the inner product of this with $\nabla^{[k]} f(\phi_0)$. Thus, in addition to the components mentioned above, computing the Taylor series approximation also requires outer and inner product units that will accommodate spaces up to those in $\Phi(\Omega^n)$.

We consider a very simple example of Taylor series approximation. Suppose we want to approximate defint $\phi$, which computes the definite integral of $\phi$, defint $\phi = \int_\Omega \phi_s\, ds$. First we determine its partial derivative at $t$ by observing:

$$
\begin{aligned}
\lim_{h\to 0}\frac{\text{defint}\,(\phi + h\delta_t) - \text{defint}\,\phi}{h} &= \lim_{h\to 0}\frac{\int_\Omega \phi_s + h\delta_t(s)\, ds - \int_\Omega \phi_s\, ds}{h} \\
&= \lim_{h\to 0}\frac{\int_\Omega \phi_s\, ds + h\int_\Omega \delta_t(s)\, ds - \int_\Omega \phi_s\, ds}{h} \\
&= \lim_{h\to 0} h\|\delta_t\|/h = 1
\end{aligned}
$$

Thus, $\dfrac{\partial}{\partial\delta_t}\,\text{defint}\,\phi = 1$, and we can see that

$$
\nabla\,\text{defint}\,\phi = \mathbf{1}, \tag{22}
$$

where $\mathbf{1}$ is the constant 1 function, $\mathbf{1}_t = 1$. This leads to a one term Taylor series, which is exact:

$$
\text{defint}\,\phi = \phi\cdot\mathbf{1} \tag{23}
$$

Note that $\mathbf{1}$ is a fixed field that must be loaded into the computer.

### 3.4 Transformations on Scalar and Vector Fields

The previous results apply to *scalar* valued functions of scalar fields. These kinds of functions are useful (e.g., to compute the average value of a scalar field), but they do not exploit the full parallelism of a field computer. Achieving this requires the use of functions that accept a (scalar or vector) field as input, and return a *field* as output. We briefly sketch the theory for scalar field valued functions of scalar fields; transformations on vector fields are an easy extension of this.

By a scalar field valued function of scalar fields we mean a function $F$: $\Phi(\Omega) \rightarrow \Phi(\Omega)$. Such a function is considered a family of scalar valued functions $f_t$: $\Phi(\Omega) \rightarrow \mathbf{R}$ for each $t \in \Omega$; these are the *component functions* of $F$. Note that $F$ can be expressed in terms of its components:

$$F(\phi) = \int_\Omega f_t(\phi)\, \omega_t\, dt \tag{24}$$

More briefly, $F = \int_\Omega f_t \omega_t\, dt$. $F$ is decomposed into its components by $\delta_t \cdot F(\phi) = f_t(\phi)$.

Next we consider the directional derivative of a field transformation. For a scalar function $f$, $\nabla_\alpha f(\phi)$ is a scalar that describes how much $f(\phi)$ changes when its argument is perturbed by a small amount in the "direction" $\alpha$. For a field transformation $F$, $\nabla_\alpha F(\phi)$ should be a *field,* each component of which reflects how much the corresponding component of $F(\phi)$ changes when $\phi$ moves in the "direction" $\alpha$. That is, $[\nabla_\alpha F(\phi)]_t = \nabla_\alpha f_t(\phi)$. Hence,

$$\nabla_\alpha F(\phi) = \int_\Omega \omega_t \nabla_\alpha f_t(\phi)\, dt \tag{25}$$

or, more briefly, $\nabla_\alpha F = \int_\Omega \omega_t \nabla_\alpha F\, dt$. It's easy to show that $\nabla_\alpha = \alpha \cdot \nabla$. The corresponding Taylor series approximation is:

$$F(\phi) = \sum_{k=1}^{n} \frac{1}{k!} [(\phi - \phi_0) \cdot \nabla]^k F(\phi_0) \tag{26}$$

As before, outer products can be used to separate the variable and fixed components.

We consider vector fields briefly. Recall that any three-dimensional vector field $\Phi$ can be considered three scalar fields $\phi$, $\psi$, $\chi$ where

$$\Phi_t = \phi_t \mathbf{i} + \psi_t \mathbf{j} + \chi_t \mathbf{k} \tag{27}$$

Similarly, a function that returns a three-dimensional vector field can be broken down into three functions that return scalar fields. Thus, we see that a transformation on finite dimensional vector fields can be implemented by a finite number of transformations on scalar fields.

To ensure the continuity of field valued functions, certain restrictions must be placed on the fields permitted as arguments. Although these restrictions are still under investigation, we believe that it is sufficient that the input field's gradient be bounded at each stage. This will be the case for all physically realizable fields. This restriction on allowable inputs finds its analogy in digital computers: legal input numbers are restricted to some range; numbers outside that range may cause underflow or overflow in the subsequent computation. In the same way here, fields whose gradients are too large may lead to incorrect results.

## 4. Conclusions

We have argued that AI is moving into a new phase characterized by biological rather than psychological metaphors. Full exploitation of this new paradigm will require a new class of computers characterized by massive parallelism: parallelism in which the number of computational units is so large it can be treated as a continuous quantity. We suggest that this leads to a new model of computation based on the transformation of continuous scalar and vector fields. We have described a class of computers, called *field computers* that conform to this model, and have indicated that they may be implemented in a variety of technologies (e.g., optical, artificial neural network, molecular).

To illustrate the capabilities and limitations of this model we have described a *universal field computer* and shown that it can be programmed for the parallel computation of a wide variety of field transformations. The universal field computer is not practical as it stands; it's an idealized computing engine. Nevertheless, just as the universal Turing machine suggests ways of designing practical von Neumann computers, so the universal field computer suggests ways of designing practical general-purpose field computers.

## 5. References


1. Backus, John, "Can Programming be Liberated from the von Neumann Style? A Functional Style and its Algebra of Programs," *Comm. ACM,* Vol. 21, No. 8 (August 1978), pp. 613-641.

2. Hinton, G. E., and Sejnowski, T. J., "Optimal Perceptual Inference," in *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition* (Washington, D.C., 1983), pp. 448-453.

3. Kirkpatrick, S., Gelatt, C. D., Jr., and Vecchi, M. P., "Optimization by Simulated Annealing," *Science,* Vol. 220 (1983), pp. 671-680.

4. MacLennan, B. J., "Logic for the New AI," in *Theoretical Foundations of Artificial Intelligence,* J. H. Fetzer (ed.), D. Reidel, in press.