A Review of Field Computation

Technical Report UT-CS-07-606

Bruce J. MacLennan*

Department of Electrical Engineering & Computer Science
University of Tennessee, Knoxville
www.cs.utk.edu/~mclennan

October 31, 2007

Abstract

This report reviews the basic principles of *field computation*, a model of massively parallel analog computation, and discusses its applications in natural and artificial intelligence. We begin with the mathematical foundations and notation for field computation; Hilbert spaces provide the basic mathematical framework. Next we discuss examples of field computation in the brain, especially in its computational maps. Fields appear in a number of contexts, including activity at the axon hillocks, in patterns of axonal connection between areas, and in patterns of synaptic connection to dendrites. The following section presents examples of field computation in the brain and in other natural and artificial systems, including fields for sensorimotor processing, excitable media, and diffusion processes. Next we consider special topics in field computation in cognition, including the separation of information (semantics) from pragmatics, and the analysis of discrete symbols as field excitations. We also consider the relevance of universal mutivariate approximation theorems to general-purpose field computation. Then we discuss hardware specifically oriented toward field computation, including electronic, optical, and chemical technologies. Finally, we consider future directions for field computation research.

Report Outline

Glossary

1. Definition
2. Introduction
3. Basic Principles
4. Field Computation in the Brain
5. Examples of Field Computation
6. Field Computers
7. Future Directions
8. Bibliography

Glossary

Axon: Nerve fiber adapted to the efficient, reliable, active transmission of neural impulses between locations in the brain or body.

Dendrite: Nerve fibers adapted to the (primarily passive) sensing and integration of signals from other neurons, which are transmitted to the neuron cell body.

Dirac delta function: A *distribution* or *generalized function* that is defined to be infinite at the origin, zero everywhere else, and to have unit area (or volume). More generally, such a function but with its infinite point located elsewhere than the origin. Dirac delta functions are idealized impulses and exist as limit objects in Hilbert spaces.

Eigenfield: An eigenfield of a linear operator has the property of passing through the operator with its shape unchanged and only its amplitude possibly modified. Equivalent to an eigenfunction of the operator, but stresses the function's role as a field.

Field: A continuous distribution of continuous quantity. Mathematically, an element of an appropriate space, such as a Hilbert space, of continuous-valued functions over a continuum. See also *phenomenological field* and *structural field*.

Field computation: Computation in which data are represented by fields, or by other representations that can be mathematically modeled by fields.

Field space: A suitably constrained set of fields. Generally field spaces are taken to be subspaces of Hilbert spaces.

Field transformation: Functions between field spaces; more generally, functions whose input and/or outputs are fields. Synonymous with *operator* in this article.

Functional: A scalar-valued function of functions, and in particular a scalar-valued field transformation.

Idempotency: An operation is idempotent when repeating it several times has the same effect as doing it once.

Impulse response: The response of a system to an input that is an idealized impulse (a *Dirac delta function*, q.v.).

Microfeature: Features of a stimulus or representation that are much smaller and at a lower level than ordinary (macro-)features, which are the sort of properties for which natural languages have words. Typically microfeatures have meaning (are interpretable) only in the context of many other microfeatures. Pixels are examples of microfeatures of images.

Nullcline: In a two-dimensional system of differential equations ($\dot{u}_k = f_k(u, v)$, $k = 1, 2$), the lines along which each of the derivatives is zero ($f_k(u, v) = 0$).

Operator: A function of functions (i.e., a functions whose inputs and/or outputs are functions), and in particular a function whose inputs and/or outputs are fields. Operators may be linear or nonlinear. Synonymous, in this article, with *field transformation*.

Orthonormal (ON): A set of vectors, fields, or functions is orthonormal if they are: (1) mutually orthogonal (i.e., inner products of distinct elements are 0), and (2) individually normalized (i.e., inner products of elements with themselves are 1).

Phenomenological field: A physical distribution of quantity that for practical purposes may be treated mathematically as a continuous distribution of continuous quantity, although it is not so in reality (cf. *structural field*)

Physical realizability: A field is physically realizable if it can be represented in some physical medium.

Population coding: Neural representation in which a population of neurons jointly represent a stimulus or other information. Each individual neuron is broadly tuned to a range of stimuli, but collectively they can represent a stimulus accurately.

Post-Moore's law computing: Refers to computing paradigms that will be important after the expiration of Moore's Law (Moore 1965), which predicts a doubling of digital logic density every two years.

Projection: A systematic pattern of axonal connections from one region of a brain to another.

Radial basis function (RBF): One of a set of real-valued functions, each of whose value decreases with distance from a central point (different for each function). The set as a whole satisfies some appropriate criteria of completeness (ability to approximate a class of functions).

Receptive field: The receptive field of a sensory neuron is the set of stimuli to which it responds. By extension, the receptive field of a non-sensory neuron is the set of inputs (from other neurons) to which it responds. Each neuron has a *receptive field profile* which describes the extent to which particular patterns of input stimulate or inhibit activity in the neuron (and so, in effect, its receptive field is fuzzy-boundaried).

Structural field: A physical field that is in reality a continuous distribution of continuous quantity (cf. *phenomenological field*).

Synapse: A connection between neurons, often from the axon of one to the dendrite of another. Electrical impulses in the pre-synaptic neuron cause neurotransmitter molecules to be secreted into the synapses between the neurons. These chemicals bind to receptors in the post-synaptic neuron membrane, and cause fluctuations in the membrane potential.

Transfer function: A function expressing the effect of a linear system on its input, expressed in terms of its effect on the amplitude and phase of each component frequency.

Unit doublet: A generalized function that is the derivative of the Dirac delta function (q.v.). It is zero except infinitesimally to the left of the origin, where it is $+\infty$, and infinitesimally to the right of the origin, where it is $-\infty$.

1 Definition

A *field* may be defined as a spatially continuous distribution of continuous quantity. The term is intended to include physical fields, such as electromagnetic fields and potential fields, but also patterns of electrical activity over macroscopic regions of neural cortex. Fields include two-dimensional representations of information, such as optical images and their continuous Fourier transforms, and one-dimensional images, such as audio signals and their spectra, but, as will be explained below, fields are not limited to two or three dimensions. A *field transformation* is a mathematical operation or function that operates on one or more fields in parallel yielding one or more fields as results. Since, from a mathematical standpoint, fields are defined over a continuous domain, field transformations operate with *continuous parallelism*. Some examples of field transformations are point-wise summation and multiplication of fields, Fourier and wavelet transforms, and convolutions and correlations.

Field computation is a model of computation in which information is represented primarily in fields and in which information processing is primarily by means of field transformations. Thus is may be described as *continuously parallel analog computing* (MacLennan 2007, 2008). Field computation may be *feed-forward*, in which one or more fields progress through a series of field transformations from input to output, or it may be *recurrent*, in which there is feedback from later stages of the field computation back to earlier stages. Furthermore, field computations can proceed in discrete sequential steps (similar to digital program execution, but with each step applying a field transformation), or in continuous time according to partial differential equations.

A distinction is often made in science between *structural fields* and *phenomenological fields*. Structural fields are physically continuous distributions of continuous quantity, such as gravitational fields and electromagnetic fields. Phenomenological fields are distributions of quantity that can be treated mathematically as though they are continuous, even if they are not physically continuous. For example, the velocity field of a macroscopic volume of fluid is a phenomenological field, because it is not physically continuous (each discrete molecule has its own velocity), but can be treated as though it is. Similarly, a macroscopic charge distribution is a phenomenological field because charge is quantized but can be treated as a continuous quantity for many purposes. Although structural fields are sometimes used, often field computation is concerned with phenomenological fields, that is, with information that can be treated as a continuous distribution of continuous quantity, regardless of whether it is physically continuous. Practically, this means that quantization in both the distribution and the quantity must be sufficiently fine that they can be modeled mathematically by continua.

One of the goals of field computation is to provide a mathematical language for describing information processing in the brain and in future large artificial neural networks intended to exhibit brain-scale intelligence. Neural computation is qualitatively different from ordinary digital computation. Computation on an ordinary

computer can be characterized as *deep but narrow*; that is, the processor operates on relatively few data values at a time, but the operations are very rapid, and so many millions of operations can be executed each second. Even on a modern parallel computer, the degree of parallelism is modest, on the order of thousands, whereas even a square millimeter of cortex has at least 146 000 neurons operating in parallel (Changeux 1985, p. 51). On the other hand, since neurons are quite slow (responding on the order of milliseconds), the "100-Step Rule" says that there can be at most about 100 sequential processing stages between sensory input and response (Feldman & Ballard 1982). Therefore, neural computation is *shallow but wide*; that is, it uses relatively few sequential stages, but each operates with a very high degree of parallelism (on the order of many millions). In addition to its speed, modern electronic digital arithmetic is relatively precise compared to the analog computation of neurons (at most about one digit of precision) (McClelland, Rumelhart & the PDP Research Group 1986, p. 378). Therefore we can conclude that neuronal information processing operates according to quite different principles to ordinary digital computing.

It is not unreasonable to suppose that achieving an artificial intelligence that is comparable to the natural intelligence of mammals will require a similar information processing architecture; in any case that seems to be a promising research direction. Therefore we should be aiming toward components with computational capabilities comparable to neurons and densities of at least 15 million per square centimeter. Fortunately, the brain demonstrates that these components do not have to be high-speed, high-precision devices, nor do they have to be precisely connected, for the detailed connections can be established through self-organization and learning. The theory of field computation can contribute in two ways: first, by providing a mathematical framework for understanding information in massively parallel analog computation systems, such as the brain, and second, by suggesting how to exploit relatively homogeneous masses of computational materials (e.g., thin films, new nanostructured materials). For the same reasons, field computers may provide an attractive alternative for "post-Moore's law computing."

2 Introduction

The term "field computation" dates from 1987 (MacLennan 1987), but examples of field computation are much older. For example, G Kirchhoff (1824–87) and others developed the *field analogy method* in which partial differential equation (PDE) problems are solved by setting up an analogous physical system and measuring it (Kirchhoff 1845). Thus a two-dimensional boundary value problem, for example determining a steady-state temperature or magnetic field distribution, could solved by setting up an analogous system with a conductive sheet or a shallow tank containing an electrolytic solution (Small 2001, p. 34). When the boundary conditions were applied, the system computed the steady-state solution field in parallel and at electronic speed. The resulting potential field could not be displayed directly at that time, and

so it was necessary to probe the field at discrete points and plot the equipotential lines; later devices allowed the equipotentials to be traced more or less automatically (Truitt & Rogers 1960, p. 2-6). In either case, setting up the problem and reading out the results were much slower than the field computation, which was comparatively instantaneous. Three-dimensional PDEs were similarly solved with tanks containing electrolytic solutions (Truitt & Rogers 1960, pp. 2-5–6). (For more on conductive sheet and electrolytic tanks methods, see Soroka (1954, ch. 9).)

Non-electronic field computation methods were also developed in the nineteenth century, but continued to be used through the first half of the twentieth century to solve the complex PDEs that arise in practical engineering (Truitt & Rogers 1960, pp. 2-8–9). For example, so called "rubber-sheet computers" were used to compute the complex electric fields in vacuum tubes. A thin elastic membrane represented the field, and rods or plates pushing the sheet down from above or up from below represented constant negative or positive potential sources. The sheet assumed the shape of the electrical potential field, which could be viewed directly. By altering the rods and plates and observing the effects on the sheet, the engineer could develop an intuitive understanding of the field's dependence on the potential sources. These simple mechanical devices used effectively instantaneous field computation to display the steady-state field's dependence on the boundary conditions.

Electrolytic tanks and conductive and elastic sheets are all examples of the use of structural fields in computation, but other mechanical field computers used discrete approximations of spatially continuous fields, and therefore made use of phenomenological fields. For example, "pin-and-rod systems," which were developed in the nineteenth century, exploited the fact that equipotential lines and flux (or stream) lines always cross at right angles (Truitt & Rogers 1960, pp. 2-9–11). A (two-dimensional) field was represented by two arrays of flexible but stiff wires, representing the flux and equipotential lines. At each crossing point was a pin with two perpendicular holes drilled through it, through which the crossing wires passed. The pins were loose enough that they could move on the wires, while maintaining, of course, their relative position and the perpendicular crossings of the wires. To solve a PDE problem (for example, determining the pressure potentials and streamlines of a non-tubulent flow through a nozzle), the edges of the pin-and-rod system were bent to to conform to the boundary conditions; the rest of the system adjusted itself to the steady-state solution field. Like the rubber-sheet computers, pin-and-rod systems allowed the solution field to be viewed directly and permitted exploration of the effects on the solution of changes in the boundary conditions.

Through the first half of the twentieth century, *network analyzers* were popular electronic analog computers, which were often used for field computation (Small 2001, pp. 35–40) This was similar to the field analogy method, but a discrete network of resistors or resistive elements replaced such continuous conducting media as the electrolytic tank and conductive sheet. Nevertheless, a sufficiently fine mesh of resistive elements may be treated as a phenomenological field, and network analyzers were

used to solve PDE problems (Truitt & Rogers 1960, pp. 2-6–8). Boundary conditions were defined by applying voltages to the appropriate locations in the network, and the resulting steady-state field values were determined by measuring the corresponding nodes in the network. As usual, it was possible to monitor the effects of boundary condition changes on particular locations in the field, and to plot them automatically or display them on an oscilloscope.

The field computers discussed so far were suited to determining the steady-state solution of a system of PDEs given specified boundary conditions; as a consequence they were sometimes called *field plotters* or *potential analyzers* (Truitt & Rogers 1960, p. 2-3). These are essentially static problems, although, as we have seen, it was possible to simulate and monitor changes in the (relatively) steady-state solution as a consequence of (relatively slowly) changing boundary conditions. On the other hand, truly dynamic problems, which simulated the evolution of a field in time, could be addressed by *reactive networks*, that is, networks incorporating capacitive and inductive elements as well as resistors (Truitt & Rogers 1960, pp. 2-11–13). For example an *RC network analyzer*, which had capacitance at each of the nodes of the resistor network, could solve the diffusion equation, for the charge on the capacitors corresponded to the concentration of the diffusing substance at corresponding locations in the medium. An *RLC network analyzer* had inductance, as well as resistance and capacitance, at each node, and so it was able to address a wider class of PDEs, including wave equations.

Although these twentieth-century field computers were constructed from discrete resistors, capacitors, and inductors, which limited the size of feasible networks, analog VLSI and emerging fabrication technologies will permit the implementation of much denser devices incorporating these and similar field computation techniques (see Sec. 6).

The following section will present the mathematical foundations and notation for field computation; Hilbert spaces provide the basic mathematical framework. Next we discuss examples of field computation in the brain, especially in its computational maps. Fields appear in a number of contexts, including activity at the axon hillocks, in patterns of axonal connection between areas, and in patterns of synaptic connection to dendrites. The following section presents examples of field computation in the brain and in other natural and artificial systems, including fields for sensorimotor processing, excitable media, and diffusion processes. Next we consider special topics in field computation in cognition, including the separation of information (semantics) from pragmatics, and the analysis of discrete symbols as field excitations. We also consider the relevance of universal mutivariate approximation theorems to general-purpose field computation. Then we discuss hardware specifically oriented toward field computation, including electronic, optical, and chemical technologies. Finally, we consider future directions for field computation research.

3 Basic Principles

3.1 Mathematical Definitions

Mathematically, a field is (generally continuous) function $\phi : \Omega \to K$ defined over some bounded domain Ω (often a compact subset of a Euclidean space) and taking values in an algebraic field K. Typically K is the real numbers, but in some applications it is the complex numbers or a vector space (see Secs. 4.1, 5.3, 5.4, and 5.6 below).

As usual, the value of a field ϕ at a point $u \in \Omega$ of its domain can be denoted by $\phi(u)$, but we more often use the notation ϕ_u with the same meaning. The latter is especially convenient for time-varying fields. For example, the value of a field ϕ at point u and time t can be denoted by $\phi_u(t)$ rather than $\phi(u, t)$. The entire field at a particular time t is then written $\phi(t)$. As is commonly done in mathematics, we may consider ϕ to be a variable implicitly defined over all $u \in \Omega$. (In this article lower-case Greek letters are usually used for fields. We occasionally use bold-face numbers, such as $\mathbf{0}$ and $\mathbf{1}$, for constant-valued fields; thus $\mathbf{0}_u = 0$ for all $u \in \Omega$. When it is necessary to make the field's domain explicit, we write $\mathbf{0}_\Omega$, $\mathbf{1}_\Omega$, etc.)

For practical field computation (e.g., in natural and artificial intelligence) we are interested in fields that can be realized in some physical medium, which places restrictions on the space of allowable fields. These restrictions vary somewhat for different physical media (e.g., neural cortex or optical fields), but we can specify a few general conditions for *physical realizability*. Generally, fields are defined over a bounded domain, although sometimes we are interested in fields that are extended in time with no prespecified bound (e.g., an auditory signal). Furthermore, since most media cannot represent unbounded field amplitudes, it is reasonable to assume that a field's range of variation is also bounded (e.g., $|\phi_u| \leq B$ for all $u \in \Omega$). In addition, most media will not support unbounded gradients, so the field's derivatives are bounded. Indeed, physically realizable fields are band-limited in both the spatial and temporal domains. Although different assumptions apply in different applications, from a mathematical perspective it is generally convenient to assume that fields are *uniformly continuous square-integrable functions* (defined below), and therefore that they belong to a Hilbert function space. In any case we use the notation $\Phi_K(\Omega)$ for a physically realizable space of K-valued fields over a domain Ω, and write $\Phi(\Omega)$ when the fields' values are clear from context.

The foregoing considerations suggest that the inner product of fields is an important concept, and indeed it is fundamental to Hilbert spaces. Therefore, if ϕ and ψ are two real-valued fields with the same domain, $\phi, \psi \in \Phi(\Omega)$, we define their inner product in the usual way:

$$\langle \phi \mid \psi \rangle = \int_\Omega \phi_u \psi_u \mathrm{d}u.$$

If the fields are complex-valued, then we take the complex conjugate of one of the

fields:

$$\langle \phi \mid \psi \rangle = \int_\Omega \phi_u^* \psi_u \mathrm{d}u.$$

For vector-valued fields $\boldsymbol{\phi}, \boldsymbol{\psi} \in \Phi_{\mathbb{R}^n}(\Omega)$ we may define

$$\langle \boldsymbol{\phi} \mid \boldsymbol{\psi} \rangle = \int_\Omega \boldsymbol{\phi}_u \cdot \boldsymbol{\psi}_u \mathrm{d}u,$$

where $\boldsymbol{\phi}_u \cdot \boldsymbol{\psi}_u$ is the ordinary scalar product on the vector space $\mathbb{R}^n$. Finally, the inner-product norm $\|\phi\|$ is defined in the usual way:

$$\|\phi\|^2 = \langle \phi \mid \phi \rangle.$$

As previously remarked, the elements of a Hilbert space are required to be square-integrable ("finite energy"): $\|\phi\| < \infty$.

3.2 Field Transformations

A *field transformation* or *operator* is any continuous function that maps one or more input fields into one or more output fields. In the simplest case a field transformation $F : \Phi(\Omega) \to \Phi(\Omega')$ maps a field in the input space $\Phi(\Omega)$ into a field in the output space $\Phi(\Omega')$.

We do not want to exclude degenerate field transformations, which operate on a field to produce a single real number, for example, or operate on a scalar value to produce a field. (An example of the former is the norm operation, $\|\cdot\|$, and an example of the latter is the operator that produces a constant-valued field over a domain.) In these cases we can consider the inputs or outputs to belong to a space $\Phi(\Omega)$ in which Ω is a singleton set. For example, the real numbers can be treated as fields in

$$\Phi^0 = \Phi_{\mathbb{R}}(\{0\}).$$

Since $\mathbb{R}$ and Φ^0 are isomorphic, we will ignore the difference between them when no confusion can result.

Another class of simple field transformations are the *local transformations*, in which each point of the output field is a function of the corresponding point in the input field. In the simplest case, the same function is applied at each point. Suppose that for input field $\phi \in \Phi_J(\Omega)$, the output field $\psi \in \Phi_K(\Omega)$ is defined $\psi_u = f(\phi_u)$, where $f : J \to K$. Then we write $\overline{f} : \Phi_J(\Omega) \to \Phi_K(\Omega)$ for the local transformation $\psi = \overline{f}(\phi)$. For example, $\overline{\log}(\phi)$ applies the log function to every element of ϕ and returns field of the results. More generally, we may apply a different function (from a parameterized family) at each point of the input field. Suppose $F : \Omega \times J \to K$, then we define $\overline{F} : \Phi_J(\Omega) \to \Phi_K(\Omega)$ so that if $\psi = \overline{F}(\phi)$, then $\psi_u = F(u, \phi_u)$.

Field transformations may be linear or nonlinear. The most common linear transformations are *integral operators of Hilbert-Schmidt type*, which are the field analogs

of matrix-vector products. Let $\phi \in \Phi(\Omega)$ and $L \in \Phi(\Omega' \times \Omega)$ be square-integrable fields; then the product $L\phi = \psi \in \Phi(\Omega')$ is defined:

$$\psi_u = \int_\Omega L_{uv}\phi_v \mathrm{d}v.$$

L is called the *kernel* of the operator. It is easy to show that physically realizable linear operators have a Hilbert-Schmidt kernel, because physically realizable fields and the operators on them are band-limited (MacLennan 1990). Therefore they can be computed by field products of the form $L\phi$.

According to the Riesz Representation Theorem (e.g., Brachman & Narici 1966, Sec. 12.4), a continuous linear functional (real-valued operator) $L : \Phi(\Omega) \to \mathbb{R}$ has a *representer*, which is a field $\rho \in \Phi(\Omega)$ such that $L\phi = \langle \rho \mid \phi \rangle$. However, since linear operators are continuous if and only if they are bounded, and since practical field transformations are bounded, all practical linear functionals have representers.

We define multilinear products in the same way. Suppose $\phi_k \in \Phi(\Omega_k)$, for $k = 1, \ldots, n$, and $M \in \Phi(\Omega' \times \Omega_n \times \cdots \Omega_2 \times \Omega_1)$. Then $M\phi_1\phi_2 \cdots \phi_n = \psi \in \Phi(\Omega')$ is defined

$$\psi_u = \int_{\Omega_n} \cdots \int_{\Omega_2} \int_{\Omega_1} M_{uv_n \cdots v_2 v_1} \phi_1(v_1)\phi_2(v_2) \cdots \phi_n(v_n) \mathrm{d}v_1 \mathrm{d}v_2 \cdots \mathrm{d}v_n.$$

Again, physically realizable multilinear operators are band limited, and so they can be computed by this kind of multilinear product (MacLennan 1990).

Like the field analogs of inner products and matrix-vector products, it is also convenient to define an analog of the outer product. For $\phi \in \Phi(\Omega)$ and $\psi \in \Phi(\Omega')$ we define the outer product $\phi \wedge \psi \in \Phi(\Omega \times \Omega')$ so that $(\phi \wedge \psi)_{(u,v)} = \phi_u \psi_v$, for $u \in \Omega$, $v \in \Omega'$. Inner, outer, and field products are related as follows for $\phi, \chi \in \Phi(\Omega)$ and $\psi \in \Phi(\Omega')$:

$$\phi(\chi \wedge \psi) = \langle \phi \mid \chi \rangle \psi = (\psi \wedge \chi)\phi.$$

In the simplest kind of field computation (corresponding to a feed-forward neural network), an input field ϕ is processed through one or more field transformations $F_1, \ldots, F_n$ to yield an output field ψ:

$$\psi = F_n(\cdots F_1(\phi) \cdots).$$

This includes cases in which the output field is the continuously-varying image of a time-varying input field,

$$\psi(t) = F_n(\cdots F_1(\phi(t)) \cdots).$$

More complex feed-forward computations may involve additional input, output, and intermediate fields, which might be variable or not.

In an ordinary artificial neural network, the activity y_i of neuron i in one layer is defined by the activities $x_1, \ldots, x_n$ of the neurons in the preceding layer by an

equation such as

$$y_i = s\left(\sum_{j=1}^{N} W_{ij}x_j + b_i\right), \tag{1}$$

where W_{ij} is the weight or strength of the connection from neuron j to neuron i, b_i is a *bias term*, and $s : \mathbb{R} \to \mathbb{R}$ is a *sigmoid function*, that is a non-decreasing, bounded continuous function. (The hyperbolic tangent is a typical example.) The field computation analog is obtained by taking the number of neurons in each layer to the continuum limit. That is, the activities ψ_u in one neural field ($u \in \Omega'$) are defined by the values ϕ_v in the input field ($v \in \Omega$) by this equation:

$$\psi_u = \int_\Omega L_{uv}\phi_v \mathrm{d}v + \beta_v,$$

where $L \in \Phi(\Omega' \times \Omega)$ is an *interconnection field* and $\beta \in \Phi(\Omega')$ is a *bias field*. More compactly,

$$\psi = \overline{s}(L\phi + \beta). \tag{2}$$

Typically, the input is processed through a series of layers, each with its own weights and biases. Analogously, in field computation we may have an N-layer neural field computation, $\phi_k = \overline{s}(L_k\phi_{k-1} + \beta_k)$, $k = 1, \ldots, N$, where $\phi_0 \in \Phi(\Omega_0)$ is the input, $\phi_N \in \Phi(\Omega_N)$ is the output, $L_k \in \Phi(\Omega_k \times \Omega_{k-1})$ are the interconnection fields, and $\beta_k \in \Phi(\Omega_k)$ are the bias fields. Other examples of neural-network style field computing are discussed later (Sec. 5).

Many important field computation algorithms are iterative, that is, they sequentially modify one or more fields at discrete moments of time. They are analogous to ordinary computer programs, except that the variables contain fields rather than scalar quantities (integers, floating-point numbers, characters, etc., and arrays of these). Since the current value of a field variable may depend on its previous values, iterative field computations involve *feedback*. Examples of iterative algorithms include field computation analogs of neural network algorithms that adapt in discrete steps (e.g., ordinary back-propagation), and recurrent neural networks, which have feedback from later layers to earlier layers.

Analog field computers, like ordinary analog computers, can operate in continuous time, defining the continuous evolution of field variable by differential equations. For instance, $\dot{\phi} = F(\phi)$ is a simple first-order field-valued differential equation, which can be written $\mathrm{d}\phi_u(t)/\mathrm{d}t = F_u[\phi(t)]$. An example is the familiar diffusion equation $\dot{\phi} = k^2\nabla^2\phi$.

Continuously varying fields arise in a number of contexts in natural and artificial intelligence. For example, sensorimotor control (in both animals and robots) depends on the processing of continuously varying input fields (e.g., visual images or auditory signals) and their transformation into continuously varying output signals (e.g., to control muscles or mechanical effectors). One of the advantages of field computing for these applications is that the fields are processed in parallel, as they are in the brain.

Often we find continuous field computation in optimization problems, in adaptation and learning, and in the solution of other continuous problems. For example, a field representing the interpretation of perceptual data (such as stereo disparity) may be continuously converging to the optimal interpretation or representation of the data.

Optimization problems are sometimes solved by continuous gradient ascent or descent on a potential surface defined by a functional F over a field space ($F : \Phi(\Omega) \to \mathbb{R}$), where $F(\phi)$ defines the "goodness" of solution ϕ. Gradient ascent is implemented by $\dot{\phi} = r\boldsymbol{\nabla}F(\phi)$, where r is the rate of ascent. This and other examples are discussed in Sec. 5.9, but the use of the gradient raises the issue of the derivatives of field transformations, such as F, which we now address.

3.3 Derivatives of Field Transformations

Since fields are functions, field spaces are function spaces (generally, Hilbert spaces), and therefore the derivatives of field transformations are the derivatives of operators over function spaces (Mathematical Society of Japan 1980, §251G). There are two common definitions of the differentiation of operators on Hilbert spaces (more generally, on Banach spaces), the Fréchet and the Gâteaux derivatives, which turn out to be the same for field transformations (MacLennan 1990). Therefore suppose that $F : \Phi(\Omega) \to \Phi(\Omega')$ is a field transformation and that U is an open subset of $\Phi(\Omega)$. Then $D \in \mathcal{L}(\Phi(\Omega), \Phi(\Omega'))$, the space of bounded linear operators from $\Phi(\Omega)$ to $\Phi(\Omega')$, is called the *Fréchet differential* of F at $\phi \in U$ if for all $\alpha \in \Phi(\Omega)$ such that $\phi + \alpha \in U$ there is an $E : \Phi(\Omega) \to \Phi(\Omega')$ such that,

$$F(\phi + \alpha) = F(\phi) + D(\alpha) + E(\alpha)$$

and

$$\lim_{\|\alpha\| \to 0} \frac{\|E(\alpha)\|}{\|\alpha\|} = 0.$$

The *Fréchet derivative* $F' : \Phi(\Omega) \to \mathcal{L}(\Phi(\Omega), \Phi(\Omega'))$ is defined by $F'(\phi) = D$, which is the locally linear approximation to F at ϕ.

Similarly $\mathrm{d}F : \Phi(\Omega) \times \Phi(\Omega) \to \Phi(\Omega')$ is a *Gâteaux derivative* of F if for all $\alpha \in U$ the following limit exists:

$$\mathrm{d}F(\phi, \alpha) = \lim_{s \to 0} \frac{F(\phi + s\alpha) - F(\phi)}{s} = \left.\frac{\mathrm{d}F(\phi + s\alpha)}{\mathrm{d}s}\right|_{s=0}.$$

If the Fréchet derivative exists, then the two derivatives are identical, $\mathrm{d}F(\phi, \alpha) = F'(\phi)(\alpha)$ for all $\alpha \in \Phi(\Omega)$.

Based on the analogy with finite-dimensional spaces, we define $\boldsymbol{\nabla}F(\phi)$, the gradient of F at ϕ, to be the field $K \in \Phi(\Omega' \times \Omega)$ satisfying $F'(\phi)(\alpha) = K\alpha$ for all α in $\Phi(\Omega)$. That is, $F'(\phi)$ is an integral operator with kernel $K = \boldsymbol{\nabla}F(\phi)$; note that

$F'(\phi)$ is an operator but $\boldsymbol{\nabla} F(\phi)$ is a field. The field analog of a directional derivative is then defined:

$$\boldsymbol{\nabla}_\alpha F(\phi) = [\boldsymbol{\nabla} F(\phi)]\alpha = F'(\phi)(\alpha).$$

Because of their importance, it is worth highlighting the gradients of functionals (real-valued operators on fields). According to the preceding definition, the gradient of a functional $F : \Phi(\Omega) \to \Phi^0$ will be a two dimensional field $\boldsymbol{\nabla} F(\phi) \in \Phi(\{0\}\times\Omega)$. (Recall $\Phi^0 = \Phi(\{0\})$.) However, when confusion is unlikely, it is more convenient to define $\boldsymbol{\nabla} F(\phi) = \gamma \in \Phi(\Omega)$, where γ is the representer of $F'(\phi)$. Then $F'(\phi)(\alpha) = \langle \gamma \mid \alpha \rangle = \langle \boldsymbol{\nabla} F(\phi) \mid \alpha \rangle$.

Higher order derivatives of field operators are defined in the obvious way, but it is important to note that each derivative is of "higher type" that the preceding. That is, we have seen that if $F : \Phi(\Omega) \to \Phi(\Omega')$, then $\mathrm{d}F : \Phi(\Omega)^2 \to \Phi(\Omega')$, where $\Phi(\Omega)^2 = \Phi(\Omega)\times\Phi(\Omega)$. Similarly, $\mathrm{d}^n F : \Phi(\Omega)^{n+1} \to \Phi(\Omega')$. Also, as $F' : \Phi(\Omega) \to \mathcal{L}(\Phi(\Omega), \Phi(\Omega'))$, so $F'' : \Phi(\Omega) \to \mathcal{L}(\Phi(\Omega), \mathcal{L}(\Phi(\Omega), \Phi(\Omega')))$ and, in general,

$$F^{(n)} : \Phi(\Omega) \to \overbrace{\mathcal{L}(\Phi(\Omega), \mathcal{L}(\Phi(\Omega), \cdots, \mathcal{L}(\Phi(\Omega),}^{n} \Phi(\Omega')) \cdots)).$$

Corresponding to higher-order derivatives are higher-order gradients:

$$\begin{aligned} \mathrm{d}F^n(\phi, \alpha_1, \ldots, \alpha_n) &= \boldsymbol{\nabla}^n F(\phi)\alpha_1 \cdots \alpha_n \\ &= \boldsymbol{\nabla}^n F(\phi)(\alpha_n \wedge \cdots \wedge \alpha_1) \\ &= \boldsymbol{\nabla}_{\alpha_n} \cdots \boldsymbol{\nabla}_{\alpha_1} F(\phi). \end{aligned}$$

For reference, we state the chain rules for Fréchet and Gâteaux derivatives:

$$(F \circ G)'(\phi)(\alpha) = F'[G(\phi)][G'(\phi)(\alpha)], \tag{3}$$

$$\mathrm{d}(F \circ G)(\phi, \alpha) = \mathrm{d}F[G(\phi), \mathrm{d}G(\phi, \alpha)]. \tag{4}$$

Just as a real function can be expanded in a Taylor series around a point to obtain a polynomial approximation, there is a corresponding theorem in functional analysis that allows the expansion of an operator around a fixed field. This suggests an approach to general-purpose computation based on *field polynomials* (MacLennan 1987), but there are also other approaches suggested by neural networks (see Sec. 5.10 below). We begin with a formal statement of the theorem.

Theorem 1 (Taylor) *Suppose that U is any open subset of $\Phi(\Omega)$ and that $F : \Phi(\Omega) \to \Phi(\Omega')$ is a field transformation that is C^n in U (that is, its first n derivatives exist). Let $\phi \in U$ and $\alpha \in \Phi(\Omega)$ such that $\phi + s\alpha \in U$ for all $s \in [0, 1]$. Then:*

$$F(\phi + \alpha) = \sum_{k=0}^{n-1} \frac{\mathrm{d}^k F(\phi, \overbrace{\alpha, \ldots, \alpha}^{k})}{k!} + R_n(\phi, \alpha),$$

where

$$R_n(\phi, \alpha) = \int_0^1 \frac{(1-s)^{n-1} \mathrm{d}^n F(\phi + s\alpha, \overbrace{\alpha, \ldots, \alpha}^{n})}{(n-1)!} \mathrm{d}s.$$

Here the "zeroth derivative" is defined in the obvious way: $\mathrm{d}^0 F(\phi) = F(\phi)$.

If the first n gradients exist (as they will for physically realizable fields and transformations), then the Taylor approximation can be written:

$$F(\phi + \alpha) = F(\phi) + \sum_{k=1}^{n} \frac{\boldsymbol{\nabla}_\alpha^k F(\phi)}{k!} + R_n(\phi, \alpha).$$

However, $\boldsymbol{\nabla}_\alpha^k F(\phi) = \boldsymbol{\nabla}^k F(\phi) \alpha^{(k)}$, where $\alpha^{(k)}$ is the k-fold outer product:

$$\alpha^{(k)} = \overbrace{\alpha \wedge \alpha \wedge \cdots \wedge \alpha}^{k}.$$

If we define the fields $\Gamma_k = \boldsymbol{\nabla}^k F(\phi)$, then we can see this approximation as a "field polynomial":

$$F(\phi + \alpha) \approx F(\phi) + \sum_{k=1}^{n} \frac{\Gamma_k \alpha^{(k)}}{k!}.$$

Such an approximation may be computed by a field analog of "Horner's rule," which is especially appropriate for computation in a series of layers similar to a neural network. Thus $F(\phi + \alpha) \approx G_0(\alpha)$, where

$$G_k(\alpha) = \Gamma_k + \frac{G_{k+1}(\alpha)}{k+1} \alpha,$$

for $k = 0, \ldots, n$, $\Gamma_0 = F(\phi)$, and $G_{n+1}(\alpha) = 0$.

4 Field Computation in the Brain

There are a number of contexts in mammalian brains in which information representations are usefully treated as fields, and information processing as field computation. These include neuronal cell bodies, patterns of axonal projection, and synapses. Of course, all of these are discrete structures, but in many cases the numbers are sufficiently large (e.g., 146×10^3 neurons / mm^2: Changeux 1985, p. 51) that the representations are usefully treated as fields; that is, they are *phenomenological fields*). (We omit discussing the intriguing possibility that the brain's electromagnetic field may affect conscious experience (McFadden 2002, Pockett 2000).)

4.1 Neuronal Fields

Computational maps, in which significant information is mapped to cortical location, are found throughout the brain (Knudsen, du Lac & Esterly 1987). For example, *tonotopic maps* in auditory cortex have systematic arrangements of neurons that respond to particular pitches, and *retinotopic maps* in visual cortex respond systematically to patches of color, edges, and other visual features projected onto the retina. Other *topographic maps* in somatosensory cortex and motor cortex systematically reflect sensations at particular locations in the body, or control motor activity at those locations, respectively. The number of identified maps is very large and there are probably many that have not been identified. And while some are quite large and can be investigated by fMRI and other noninvasive imaging techniques, other are less than a square millimeter in size (Knudsen, du Lac & Esterly 1987). However, even a 0.1mm^2 map may have tens of thousands of neurons, and thus be analyzed reasonably as a field.

In mathematical terms, let $\mathcal{X}$ be a space of features represented by a cortical map. These might be microfeatures of a sensory stimulus (e.g., oriented edges at particular retinal locations) or motor neurons (e.g., controlling muscle fibers in particular locations). These examples are peripheral features, but $\mathcal{X}$ might represent patterns of activity in nonperipheral groups of neurons (e.g., in other cortical maps). Let Ω be a two-dimensional manifold corresponding to a cortical map representing $\mathcal{X}$. There will a piecewise continuous function $\mu : \mathcal{X} \to \Omega$ so that $\mu(x)$ is the cortical location corresponding to feature $x \in \mathcal{X}$. The mapping μ may be only piecewise continuous since $\mathcal{X}$ may be of higher dimension that Ω. (This is the reason, for example, that we find stripes in striate cortex; it is a consequence of mapping a higher dimensional space into a lower one.)

Typically, the activity $\phi_{\mu(x)}$ at a cortical location $\mu(x)$ will reflect the degree of presence of the feature x in the map's input. Furthermore, the responses of neurons are often broadly tuned, therefore the response at a location $\mu(x')$ will generally be a decreasing function $r[d(x, x')]$ of the distance $d(x, x')$, where d is some appropriate metric on $\mathcal{X}$. Therefore an input feature x will generate a response field $\phi = \xi(x)$ given by

$$\phi_{\mu(x')} = r[d(x, x')].$$

If a number of features $x_1, \ldots, x_n$ are simultaneously present in the input, then the activity in the map may be a superposition of the activities due to the individual features:

$$\xi(x_1) + \cdots + \xi(x_n).$$

Furthermore, a sensory or other input, represented as a subset $\mathcal{X}' \subset \mathcal{X}$ of the feature space, generates a corresponding field,

$$\xi(\mathcal{X}') = \int_{\mathcal{X}'} \xi(x) \mathrm{d}x$$

(with an appropriate definition of integration for $\mathcal{X}$, which usually can be taken to be a measure space). (See Sec. 5.5 for more on computation on superpositions of inputs via topographic maps.)

The preceding discussion of cortical maps refers somewhat vaguely to the "activity" of neurons, which requires clarification. In cortical maps the represented microfeatures are correlated most closely with the location of the neuronal cell body, which often interacts with nearby neurons. Therefore, when a cortical map is treated mathematically as a field, there are several physical quantities that can be interpreted as the field's value ϕ_u at a particular cortical location u. Although the choice depends somewhat on the purpose of the analysis, the most common interpretation of $\phi_u(t)$ will be the instantaneous spiking frequency at time t of the neuron at location u. We will refer to $\phi(t) \in \Phi(\Omega)$ as the *neuronal field* (at time t) associated with the neurons u in the map Ω.

The relative phase of neural impulses is sometimes relevant to neural information processing (Hopfield 1995). For example, the relative phase with which action potentials arrive a neuron's dendrites can determine whether or not the induced postsynaptic potentials add. In these cases it may be convenient to treat neural activity as a complex-valued field, $\psi(t) \in \Phi_{\mathbb{C}}(\Omega)$, which can be written in polar form:

$$\psi(t) = \rho(t)e^{i\phi(t)}.$$

Then the magnitude (or modulus) field $\rho(t)$ may represent the impulse rate and the phase (or argument) field $\phi(t)$ may represent the relative phase of the impulses. That is, $\rho_u(t)$ is the rate of neuron u (at time t) and $\phi_u(t)$ is its phase. For example, in a *bursting neuron* (which generates impulses in clusters), $\rho(t)$ can represent the impulse rate within the clusters and $\phi(t)$ the relative phase of the clusters. More generally, in a complex-valued neuronal field, the phase part may represent microfeatures of stimulus, while the magnitude part represent pragmatic characteristics of the microfeatures, such as their importance, confidence, or urgency. (Such dual representations, comprising semantics and pragmatics, are discussed in Sec. 5.7.)

4.2 Synaptic and Dendritic Fields

The surface of each neuron's dendritic tree and soma (cell body) is complicated two-dimensional manifold Ω_{m}, and so the electrical field across the neuron's membrane is naturally treated as a two-dimensional potential field $\phi \in \Phi(\Omega_{\mathrm{m}})$. Synaptic inputs create electrical disturbances in this field, which, to a first approximation, propagate passively according to the cable equations (Anderson 1995*a*, pp. 25–31). However, there are also nonlinear effects due to voltage-gated ion channels etc. (Rumelhart, McClelland & the PDP Research Group 1986, p. 381). Therefore the membrane field obeys a nonlinear PDE (partial differential equation) dependent on a synaptic input field ϵ:

$$M(\epsilon, \phi, \dot{\phi}, \ddot{\phi}, \ldots) = 0.$$

The electrical field ϕ on the membrane includes the field ϕ_a at the axon hillock $a \in \Omega_{\mathrm{m}}$. This voltage determines the rate at which the neuron generates action potentials (APs, nerve impulses), which constitute the neuron's contribution to a neuronal field. The dependence of the impulse rate r on the membrane field, $r(t) = A_r[\phi(t)]$, which is approximately linear (that is, the rate is proportional to the depolarization, relative to the resting potential, at the axon hillock). To a first approximation, the dendritic tree implements an approximately linear (adaptive) analog filter on its input field (MacLennan 1993, 1994*a*). Some purposes require a more detailed analysis, which looks at the time-varying action potential $V(t)$, rather than at the instantaneous impulse rate, as a function of the membrane field, $V(t) = A_V[\phi(t)]$.

Many neurons have tens of thousands of synaptic inputs (Anderson 1995*a*, p. 304), and so these quantitative properties can be treated as a field over a domain Ω, which is a subset of the dendritic membrane. The post-synaptic potential ϵ_s at synapse s is a result of the synaptic efficacy σ_s and the pre-synaptic axonal impulse rate ζ_s. The synaptic efficacy is the composite effect of the number of receptors for the neurotransmitter released by the incoming axon, as well as of other factors, such as the dependence of neurotransmitter flux on the impulse rate. Some learning processes (e.g., long-term potentiation) alter the synaptic efficacy field σ.

However, because synaptic transmission involves the diffusion of neurotransmitter molecules across the synaptic cleft, the subsequent binding to and unbinding from receptors, and the opening and closing of ion channels, the post-synaptic potential is not a simple product, $\epsilon_s = \sigma_s \zeta_s$. Rather, the synaptic system filters the input field. To a first approximation we may analyze it as a linear system S:

$$\begin{pmatrix} \epsilon \\ \dot{\psi} \end{pmatrix} = \mathrm{S}(\sigma) \begin{pmatrix} \zeta \\ \psi \end{pmatrix},$$

where ψ represents the internal state of the synaptic system (concentrations of neurotransmitter in the clefts, receptor and ion channel states, etc.). The parameter σ shows the system's dependence on the synaptic efficacies. The preceding equation is an abbreviation for the following system (in which we suppress the σ parameter):

$$\begin{aligned} \epsilon &= S_{11}\zeta + S_{12}\psi, \\ \dot{\psi} &= S_{21}\zeta + S_{22}\psi, \end{aligned}$$

in which the products are Hilbert-Schmidt integral operators (that is, the S_{ij} are fields operating on the input and state fields).

4.3 Axonal Projection Fields

Bundles of axons form *projections* from one brain region to another; through the pattern of their connections they may effect certain field transformations (explained below). The input is typically a neuronal field $\phi \in \Phi(\Omega)$ defined over the source region

Ω. At their distal ends the axons branch and form synapses with the dendritic trees of the neurons in the destination region. Since each axon may form synapses with many destination neurons, and each neuron may receive synapses from many axons, it is convenient to treat all the synapses of the destination neurons as forming one large synaptic system S, where the synaptic efficacies σ_u range over all the synapses u in the destination region, $u \in \Omega'$. Correspondingly we can consider the field $\zeta \in \Phi(\Omega')$ of pre-synaptic inputs ζ_u to all of these synapses. The axons and their synapses define an *axonal projection system* P, which is, to a first approximation, a linear system:

$$\begin{pmatrix} \zeta \\ \dot{\alpha} \end{pmatrix} = \mathrm{P} \begin{pmatrix} \phi \\ \alpha \end{pmatrix},$$

where α represents the internal state of the axonal projection system.

The function of axons is to transmit nerve impulses over relatively long distances with no change of amplitude or waveform. However, there is a transmission delay, and different axons in a projection may introduce different delays. Thus an axonal projection may change the phase relationships of the input field, in addition to introducing an overall delay. On the basis of our analysis the axonal projection as a linear system, we can express the Laplace transform Z of the pre-synaptic field $\zeta(t)$ in terms of the transfer function H^{S} of the projection and the Laplace transform Φ of the input field $\phi(t)$:

$$\mathrm{Z}(s) = H^{\mathrm{S}}(s)\Phi(s)$$

(where s is the conjugate variable of time). Note that all the variables refer to fields, and so this equation means

$$\mathrm{Z}_u(s) = \int_\Omega H^{\mathrm{S}}_{uv}(s)\Phi_v(s)\mathrm{d}v,$$

where $H^{\mathrm{S}}_{uv}(s) \in \Phi_{\mathbb{C}}(\Omega' \times \Omega)$ is the (complex-valued) transfer function to synapse u from input neuron v. Since the effects of the axons are pure delays, the transfer function is imaginary:

$$H^{\mathrm{S}}_{uv}(s) = \exp(-i\Delta_{uv}s),$$

where Δ_{uv} is the delay imposed by the axon from neuron v to synapse u. Thus the delay field $\Delta \in \Phi(\Omega' \times \Omega)$ defines the effect of the axonal projection on the input field.

The system S comprising all the synapses of the destination neurons is also characterized by a transfer function $H^{\mathrm{S}}(s)$; that is, $\mathrm{E}(s) = H^{\mathrm{S}}(s)\mathrm{Z}(s)$, where $\mathrm{E}(s)$ is the Laplace transform of the post-synaptic field $\epsilon(t)$. Therefore the combined effect of the axonal projection and the synapses is $\mathrm{E}(s) = H^{\mathrm{SP}}(s)\Phi(s)$, where the composite transfer function is $H^{\mathrm{SP}}(s) = H^{\mathrm{S}}(s)H^{\mathrm{P}}(s)$. Note that this a field equation, which abbreviates

$$H^{\mathrm{SP}}_{uv}(s) = \int_{\Omega'} H^{\mathrm{S}}_{uw}(s)H^{\mathrm{P}}_{wv}\mathrm{d}w.$$

The transfer function $H^{\mathrm{SP}}_{uv}(s)$ has a corresponding *impulse response* $\eta^{\mathrm{SP}}_{uv}(t)$, which represents the post-synaptic response at u to a mathematical impulse (Dirac delta function) injected at v. (For Dirac delta functions, see Glossary and Sec. 5.1.4.) The impulse response characterizes the effect of signal transmission to u from v as follows:

$$\epsilon_u(t) = \int_{\Omega'} \eta^{\mathrm{SP}}_{uv}(t) \otimes \phi_v(t) \mathrm{d}v,$$

where "$\otimes$" represents convolution in the time domain. This may be abbreviated as a field equation, $\epsilon(t) = \eta^{\mathrm{SP}}(t) \otimes \phi(t)$.

Since axonal projections largely determine the *receptive fields* of the destination neurons, it will be worthwhile to consider the relation of the projection field to the neuronal field at the destination region. Therefore, let ψ_w represent the output signal of a destination neuron w in response to an input field ϕ. We may write

$$\psi_w(t) = F_w[\eta^{\mathrm{SP}}(t) \otimes \phi(t)],$$

where F_w represents the (possibly nonlinear) function computed by neuron w on the subset of the post-synaptic signal $\eta^{\mathrm{SP}}(t) \otimes \phi(t)$ in its dendritic tree. Therefore, the destination neuronal field is given by the field equation $\psi = F[\eta^{\mathrm{SP}} \otimes \phi]$. Many neurons behave as "leaky integrators" (Anderson 1995*a*, pp. 52–4), which are approximately linear, and in these cases the combined effect of the axonal projection, synaptic field, and destination neurons is a linear operator applied to the input signal, $\psi(t) = L\phi(t)$.

5 Examples of Field Computation

5.1 Neural-Network-Like Computation

Many neural network approaches to artificial intelligence can be adapted easily to field computation, effectively by taking the number of neurons in a layer to the continuum limit. For example, as discussed in Sec. 3.2, $\psi = \overline{s}(L\phi + \beta)$ (Eq. 2) is the field analog of one layer of a neural net, that is, a continuum neural net, with interconnection field L and bias field β.

5.1.1 Discrete Basis Function Networks

Radial basis function (RBF) networks are a familiar and useful class of artificial neural networks, which have similarities to neural networks in the brain (Light 1992, Powell 1985). Indeed, RBF networks are inspired by the observation that many sensory neurons are tuned to a point in sensory space and that their response falls off continuously with distance from that central point (recall Sec. 4.1). RBFs are usually defined over finite dimensional spaces, but the extension to fields is straight-forward. Therefore we will consider a set of functionals $r_1, r_2, \ldots$, where $r_j : \Phi(\Omega) \to [0, 1]$.

Typically we restrict our attention to finite sets of basis functionals, but we include the infinite case for generality. The intent is that each r_j is tuned to a different field input η_j, its "focal field," and that $r_j(\phi)$ represents the closeness of ϕ to the focal field η_j.

If all the RBFs have the same *receptive field profile*, that is, the same fall-off of response with increasing distance from the focal field, then we can write $r_j(\phi) = r(\|\phi - \eta_j\|)$, where the receptive field profile is defined by a $r : [0, \infty) \to [0, 1]$ that is monotonically decreasing with $r(0) = 1$ and $r(x) \longrightarrow 0$ as $x \longrightarrow \infty$.

As is well known, the inner product is frequently used as a measure of similarity. Expanding the difference in terms of the inner product yields:

$$\|\phi - \eta_j\|^2 = \|\phi\|^2 - 2\langle\phi \mid \eta_j\rangle + \|\eta_j\|^2.$$

The inverse relation between the inner product and distance is especially obvious if, as is often the case (see Sec. 5.7), the input and focal fields are normalized ($\|\phi\| = 1 = \|\eta_j\|$); then:

$$\|\phi - \eta_j\|^2 = 2 - 2\langle\phi \mid \eta_j\rangle.$$

Therefore, RBFs with an identical fall-off of response can be defined in terms of a fixed function $s : [-1, 1] \to [0, 1]$ applied to the inner product, $r_j(\phi) = s(\langle\phi \mid \eta_j\rangle)$, where the monotonically increasing function s equals 1 when $\phi = \eta_j$ and equals 0 when the fields are maximally different ($\phi = -\eta_j$). That is, for normalized fields $\langle\phi \mid \eta_j\rangle \in [-1, 1]$, and so $s(-1) = 0$, $s(1) = 1$.

Such RBFs are closely related to familiar artificial neurons (Eq. 1). Indeed, we may define $r_j(\phi) = s(\langle\eta_j \mid \phi\rangle + b_j)$, where $s : \mathbb{R} \to [0, 1]$ is a sigmoidal activation function and b_j is the bias term. Here the input ϕ to the neuron is a field, as is its receptive field profile η_j, which is the focal field defined by the neuron's interconnection field.

Generally, neurons are quite broadly tuned, and so individual RBFs do not characterize the input very precisely, but with an appropriate distribution of focal fields the collection of RBFs can characterize the input accurately, a process known as *coarse coding* (e.g., Rumelhart, McClelland & the PDP Research Group 1986, pp. 91–6, Sanger 1996). Therefore the discrete ensemble of RBFs compute a representation $\mathbf{p}(\phi)$ of the input given by $p_j(\phi) = r_j(\phi)$.

When information is represented in some way we must consider the adequacy of the representation for our information processing goals. In general, it is not necessary that a representation function $\mathbf{p}$ preserve all characteristics and distinctions of the input space; indeed often the function of representation is to extract the relevant features of the input for subsequent processing. Nevertheless it will be worthwhile to consider briefly RBF-like representations that do not lose any information. A Hilbert function space is isomorphic (indeed, isometric) to the space ℓ_2 of square-summable sequences; that is, there is a one-to-one correspondence between fields and the infinite sequences of their generalized Fourier coefficients. Therefore let $\beta_1, \beta_2, \ldots$ be any orthonormal (ON) basis of $\Phi(\Omega)$ and define $p_j(\phi) = \langle\beta_j \mid \phi\rangle$. Define $\mathbf{p} : \Phi(\Omega) \to \ell_2$ so that $\mathbf{p}(\phi)$

is the infinite sequence of generalized Fourier coefficients, $(p_1(\phi), p_2(\phi), \ldots)$. Mathematically, we can always find an m such that the first m coefficients approximate the fields as closely as we like; practically, physically realizable fields are band-limited, and so they have only a finite number of nonzero Fourier coefficients. Therefore, we may use $\mathbf{p}^m : \Phi(\Omega) \to \mathbb{R}^m$ to compute the m-dimensional representations (relative to an understood ON basis):

$$\mathbf{p}^m(\phi) = (p_1(\phi), p_2(\phi), \ldots, p_m(\phi))^{\mathrm{T}}.$$

5.1.2 Continua of Basis Functions

In the preceding section we looked at the field computation of a discrete, typically finite, set of basis functionals. This is appropriate when the basis elements are relatively few in number and there is no significant topological relation among them. In the brain, however, large masses of neurons typically have a significant topological relation (e.g., they may form a topographic map (Sec. 4.1), and so we are interested in cases in which each point in an output field ψ is a result of applying a different basis function to the input field. Suppose $\phi \in \Phi(\Omega)$ and $\psi \in \Phi(\Omega')$. For all $u \in \Omega'$ we want $\psi_u = R(u, \phi)$, where $R : \Omega' \times \Phi(\Omega) \to \Phi(\Omega')$. That is, R defines a family of functionals in which, for each u, $R(u, —)$ has a different focal field, which varies continuously with u.

For example, suppose we want ψ_u to be an inner-product comparison of ϕ with the focal field η^u: $\psi_u = s(\langle \eta^u \mid \phi \rangle)$. Since $\langle \eta^u \mid \phi \rangle = \int_\Omega \eta^u_v \phi_v \mathrm{d}v$, define the field $\mathrm{H} \in \Phi(\Omega' \times \Omega)$ by $\mathrm{H}_{uv} = \eta^u_v$. Then a point in the output field is given by $\psi_u = s[(\mathrm{H}\phi)_u]$, and the entire field is computed by:

$$\psi = \overline{s}(\mathrm{H}\phi). \tag{5}$$

This is, of course, the field analog of one layer of a neural net (Eq. 2), but with no bias field. In a similar way we can define a continuum of RBFs: $\psi_u = r(\|\phi - \eta^u\|)$.

5.1.3 Spatial Correlation and Convolution

A special case of Eq. 5 rises when all the focal fields η^u are the same shape but centered on different points $u \in \Omega$. That is, $\eta^u_v = \varrho(v - u)$, where $\varrho \in \Phi(\Omega)$ is the common shape of the focal fields (their *receptive field profile*). In this case,

$$\langle \eta^u \mid \phi \rangle = \int_\Omega \varrho(v - u)\phi(v)\mathrm{d}v.$$

This is simply the *cross-correlation* of ϱ and ϕ, which we may write $\varrho \star \phi$. In general,

$$(\psi \star \phi)_u = \int_\Omega \psi(v - u)\phi(v)\mathrm{d}v, \tag{6}$$

which gives the correlation of ψ and ϕ at a relative displacement u. Therefore in this case the RBF field is given by $\psi = \overline{s}(\varrho \star \phi)$. If the receptive field ϱ is symmetric, $\varrho(-x) = \varrho(x)$, then

$$\langle \eta^u \mid \phi \rangle = \int_\Omega \varrho(u - v)\phi(v)\mathrm{d}v,$$

which is $\varrho \otimes \phi$, the *convolution* of ϱ and ϕ. In general,

$$(\psi \otimes \phi)_u = \int_\Omega \psi(u - v)\phi(v)\mathrm{d}v. \tag{7}$$

Hence $\psi = \overline{s}(\varrho \otimes \phi)$ when ϱ is symmetric. Computation of these fields by means of convolution or correlation rather than by the integral operator (Eq. 5) may be more convenient on field computers that implement convolution or correlation directly.

5.1.4 Approximation of Spatial Integral and Differential Operators

Correlation and convolution (Eqs. 6, 7) can be used to implement many useful linear operators, in particular spatial integral and differential operators. Of course these linear operations can be implemented by a field product with the appropriate Hilbert-Schmidt kernel, but convolution and correlation make use of lower dimensional fields than the kernel.

For example, suppose we want to compute the indefinite spatial integral of a field $\phi \in \Phi(\mathbb{R})$. That is, we want to compute $\psi = \int \phi$ defined by $\psi_x = \int_{-\infty}^{x} \phi_y \mathrm{d}y$. This can be computed by the convolution $\psi = \upsilon \otimes \phi$ where υ is the *Heaviside* or *unit step field* on $\mathbb{R}$:

$$\upsilon_x = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}.$$

The Heaviside field is discontinuous, and therefore it may not be physically realizable, but obviously it may be approximated arbitrarily closely by a continuous field.

Spatial differentiation is important in image processing in nervous systems and artificial intelligence systems. In the one-dimesional case, for $\phi \in \Phi(\mathbb{R})$ we want $\phi' \in \Phi(\mathbb{R})$, where $\phi'_u = \mathrm{d}\phi_u/\mathrm{d}u$. To express this as a convolution we may begin by considering the *Dirac delta function* or *unit impulse function* δ, which is the derivative of the unit step function, $\delta(x) = \upsilon'(x)$. This is a *generalized function* or *distribution* with the following properties:

$$\begin{aligned} \delta(0) &= +\infty, \\ \delta(x) &= 0,\ x \neq 0, \\ \int_{-\infty}^{+\infty} \delta(x)\mathrm{d}x &= 1. \end{aligned}$$

Obviously such a function is not physically realizable (more on that shortly), but such functions exist as limit objects in Hilbert spaces. The Dirac delta satisfies the

following "sifting property":

$$\phi_x = \int_{-\infty}^{+\infty} \delta(x-y)\phi(y)\mathrm{d}y;$$

that is, the Dirac delta is an identity for convolution, $\phi = \delta \otimes \phi$. Now observe:

$$\begin{aligned} \phi'_x &= \mathrm{D}_x \int_{-\infty}^{+\infty} \delta(x-y)\phi(y)\mathrm{d}y \\ &= \int_{-\infty}^{+\infty} \delta'(x-y)\phi(y)\mathrm{d}y, \end{aligned}$$

where δ' is the derivative of the Dirac delta. It is called the *unit doublet* and has the property of being zero everywhere except infinitesimally to the left of the origin, where it is $+\infty$, and infinitesimally to the right of the origin, where it is $-\infty$. Thus the spatial derivative of a field can be computed by convolution with the unit doublet: $\phi' = \delta' \otimes \phi$.

Obviously, neither the unit impulse (Dirac delta) nor the unit doublet is physically realizable, but both may be approximated arbitrarily closely by physically realizable fields. For example, the delta function can be approximated by a sufficiently sharp Gaussian field γ (i.e., $\gamma_x = \sqrt{r/\pi}\exp(-rx^2)$ for sufficiently large r). Corresponding to the sifting property $\phi = \delta \otimes \phi$ we have Gaussian smoothing $\phi \approx \gamma \otimes \phi$, which is a typical effect of the limited bandwidth of physically realizable fields in cortex and other physical media. Similarly, the unit doublet can be approximated by a *derivative of Gaussian (DoG)* field γ', where $\gamma'_x = \mathrm{d}\gamma_x/\mathrm{d}x$. Thus, the spatial derivative can be approximated the convolution $\phi' \approx \gamma' \otimes \phi$. Indeed, in the nervous system we find neurons with approximately DoG receptive field profiles. (These derivative formulas are perhaps more intuitively expressed in terms of correlation, $\phi' = (-\delta') \star \phi \approx (-\gamma') \star \phi$, since this is more easily related to the difference, $\phi_{x+\epsilon} - \phi_{x-\epsilon}$.)

If ψ is a two-dimensional field, $\psi \in \Phi(\mathbb{R}^2)$, it is easy to show that the partial derivative along the first dimension can be computed by convolution with $\delta' \wedge \delta$, and along the second by convolution with $\delta \wedge \delta'$. The partial derivatives may be approximated by convolutions with $\gamma' \wedge \gamma$ and $\gamma \wedge \gamma'$. The divergence of a field can be computed by a two-dimensional convolution with the sum of these fields:

$$\nabla \cdot \psi = (\delta' \wedge \delta + \delta \wedge \delta') \otimes \psi \approx (\gamma' \wedge \gamma + \gamma \wedge \gamma') \otimes \psi.$$

Similarly the gradient is

$$\nabla\psi = [(\delta' \wedge \delta) \otimes \psi]\mathbf{i} + [(\delta \wedge \delta') \otimes \psi]\mathbf{j},$$

where $\mathbf{i} \in \Phi_{\mathbb{R}^2}(\mathbb{R}^2)$ is a constant vector field of unit vectors in the x direction, $\mathbf{i}_{(x,y)} = (1,0)$, and $\mathbf{j}$ is a similar field in the y direction. It is approximated by

$$\nabla\psi \approx [(\gamma' \wedge \gamma) \otimes \psi]\mathbf{i} + [(\gamma \wedge \gamma') \otimes \psi]\mathbf{j}. \tag{8}$$

To compute the Laplacian we need the second partial derivatives, but note that for a one-dimensional field $\phi'' = \delta' \otimes (\delta' \otimes \phi) = (\delta' \otimes \delta') \otimes \phi = \delta'' \otimes \phi$, where δ'' is the second derivative of the Dirac function (a "unit triplet"). Hence, for two-dimensional ψ

$$\nabla^2 \psi = (\delta'' \wedge \delta + \delta \wedge \delta'') \otimes \psi \approx (\gamma'' \wedge \gamma + \gamma \wedge \gamma'') \otimes \psi, \tag{9}$$

where γ'' is the second derivative of the Gaussian, a typical (inverted) "Mexican hat function" with the center-surround receptive-field profile often found in the nervous system. These formulas extend in the obvious way to higher-dimensional fields.

5.2 Change of Field Domain

We have seen that physically realizable linear operators are integral operators, and therefore can be computed by field products of the form $K\phi$. However, the kernel K might not be physically realizable if its dimension is too high. For example, suppose $L : \Phi(\Omega) \to \Phi(\Omega)$ is a linear operator on two dimensional visual images; that is, Ω is a bounded subset of two-dimensional Euclidean space. Its kernel K, satisfying $K\phi = L(\phi)$, will be a four-dimensional field $K \in \Phi(\Omega \times \Omega)$, and therefore physically unrealizable. Therefore we need means for realizing or approximating high-dimensional fields in three or fewer spatial dimensions.

The simplest way to accomplish this is to represent fields of higher dimensional spaces by corresponding fields over lower dimensional spaces. For example, to represent $\phi \in \Phi(\Omega)$ by $\psi \in \Phi(\Omega')$, suppose $\beta_1, \beta_2, \ldots$ is an ON basis for $\Phi(\Omega)$, as $\eta_1, \eta_2, \ldots$ is for $\Phi(\Omega')$. Then, let the generalized Fourier coefficients of ϕ be used as the coefficients to compute a corresponding ψ. Observe:

$$\psi = \sum_k \eta_k \langle \beta_k \mid \phi \rangle = \sum_k (\eta_k \wedge \beta_k)\phi.$$

(Of course, a finite sum is sufficient for physically realizable fields.) Therefore the change of basis can be implemented by the kernel $K = \sum_k \eta_k \wedge \beta_k$. By this means, any Hilbert-Schmidt operator on two-dimensional fields can be implemented by a physically realizable field product: represent the input by a one-dimensional field, generate the one-dimensional representation of the output by a product with a two-dimensional kernel, and convert this representation to the output field. Specifically, suppose $\phi \in \Phi(\Omega)$, $\psi \in \Phi(\Omega')$, and $L : \Phi(\Omega) \to \Phi(\Omega')$ is a Hilbert-Schmidt linear operator. The three-dimensional kernel $\mathrm{H} = \sum_k \eta_k \wedge \beta_k \in \Phi([0,1] \times \Omega)$ will be used to generate a one-dimensional representation of the two-dimensional input, $\mathrm{H}\phi \in \Phi([0,1])$. Similarly, the two-dimensional output will be generated by $\Theta = \sum_j \zeta_j \wedge \eta_j \in \Phi(\Omega' \times [0,1])$, where $\zeta_1, \zeta_2, \ldots$ is an ON basis for $\Phi(\Omega')$. It is easy to show that the required two-dimensional kernel $K \in \Phi([0,1]^2)$ such that $L = \Theta K \mathrm{H}$ is just

$$K = \sum_{jk} \langle \zeta_j \mid L\beta_k \rangle (\eta_j \wedge \eta_k).$$

We have seen (Sec. 5.1) that field computation can often be implemented by neural-network-style computation on finite-dimensional spaces. For example, a linear field transformation (of Hilbert-Schmidt type) can be factored through the *eigenfield basis*. Let $\epsilon_1, \epsilon_2, \ldots$ be the eigenfields of L with corresponding eigenvalues $e_1, e_2, \ldots$: $L\epsilon_k = e_k\epsilon_k$. The eigenfields can be chosen to be orthonormal (ON), and, since $\Phi(\Omega)$ is a Hilbert space, only a finite number of the eigenvalues are greater than any fixed bound, so ϕ can be approximated arbitrarily closely by a finite sum $\phi \approx \sum_{k=1}^{m} c_k\epsilon_k$, where $c_k = \langle \epsilon_k \mid \phi \rangle$; that is, ϕ is represented by the finite dimensional vector $\mathbf{c}$. The discrete set of coefficients $c_1, \ldots, c_m$ is not a field because there is no significant topological relationship among them; also, typically, m is relatively small.

The output ψ is computed by a finite sum, $\psi \approx \sum_{k=1}^{m} \epsilon_k e_k c_k$. In terms of neural computation, we have a finite set of neurons $k = 1, \ldots, m$ whose receptive field profiles are the eigenfields, so that they compute $e_k c_k = e_k \langle \epsilon_k \mid \phi \rangle$. The outputs of these neurons amplitude-modulate the generation of the individual eigenfields ϵ_k, whose superposition yields the output ψ.

It is not necessary to factor the operator through the eigenfield basis. To see this, suppose $L : \Phi(\Omega) \to \Phi(\Omega')$ and that the fields β_k are an ON basis for $\Phi(\Omega)$ and that the fields ζ_j are an ON basis for $\Phi(\Omega')$. Represent the input by a finite-dimensional vector $\mathbf{c}$, where $c_k = \langle \beta_k \mid \phi \rangle$. Then the output ψ can be represented by the finite dimensional vector $\mathbf{d}$, where $d_j = \langle \zeta_j \mid \psi \rangle$. (Since the input and output spaces are both Hilbert spaces, only a finite number of these coefficients are greater than any fixed bound.) It is easy to show $\mathbf{d} = \mathrm{M}\mathbf{c}$, where $M_{jk} = \langle \zeta_j \mid L\beta_k \rangle$ (the Hilbert-Schmidt theorem). In neural terms, a first layer of neurons with receptive field profiles β_k compute the discrete representation $c_k = \langle \beta_k \mid \phi \rangle$. Next, a layer of linear neurons computes the linear combinations $d_j = \sum_{k=1}^{m} M_{jk} c_k$ in order to control the amplitudes of the output basis fields in the output superposition $\psi \approx \sum_{j=1}^{n} d_j \zeta_j$. In this way, an arbitrary linear field transformation may be computed through a neural representation of relatively low dimension.

If a kernel has too high dimension to be physically realizable, it is not necessary to completely factor the product through a discrete space; rather, one or more dimensions can be replaced by a discrete set of basis functions and the others performed by field computation. To see the procedure, suppose we have a linear operator $L : \Phi(\Omega) \to \Phi(\Omega')$ with kernel $K \in \Phi(\Omega' \times \Omega)$, where $\Omega = \Omega_1 \times \Omega_2$ is of too great dimension. Let $\psi = K\phi$ and observe

$$\psi_u = \int_{\Omega} K_{uv}\phi_v \mathrm{d}v = \int_{\Omega_1}\int_{\Omega_2} K_{uxy}\phi_x(y)\mathrm{d}y\mathrm{d}x,$$

where we consider $\phi_v = \phi_{xy}$ as a function of y, $\phi_x(y)$. Expand ϕ_x in terms of an ON basis of $\Phi(\Omega_2)$, $\beta_1, \beta_2, \ldots$:

$$\phi_x = \sum_k \langle \phi_x \mid \beta_k \rangle \beta_k.$$

Note that

$$\langle \phi_x \mid \beta_k \rangle = \int_{\Omega_2} \phi_{xy} \beta_k(y) \mathrm{d}y = (\phi\beta_k)_x,$$

where $\phi\beta_k \in \Phi(\Omega_1)$. Rearranging the order of summation and integration,

$$\psi_u = \sum_k \int_{\Omega_1} \int_{\Omega_2} K_{uxy} \beta_k(y) (\phi\beta_k)_x \mathrm{d}y \mathrm{d}x = \sum_k [K\beta_k(\phi\beta_k)]_u.$$

Hence, $\psi = \sum_k K\beta_k(\phi\beta_k)$. Let $J_k = K\beta_k$ to obtain a lower-dimensional field computation:

$$L(\phi) = \sum_k J_k(\phi\beta_k).$$

Note that $J_k \in \Phi(\Omega' \times \Omega_1)$ and all the other fields are of lower dimension than $K \in \Phi(\Omega' \times \Omega)$. As usual, for physically realizable fields, a finite summation is sufficient.

We can discretize $\Phi(\Omega_1)$ by a similar process, which also can be extended straightforwardly to cases where several dimensions must be discretized. Normally we will discretize the dimension that will have the fewest generalized Fourier coefficients, given the bandwidth of the input fields.

The foregoing example discretized one dimension of the input space, but it is also possible to discretize dimensions of the output space. Therefore suppose $L : \Phi(\Omega) \to \Phi(\Omega')$ with kernel $K \in \Phi(\Omega' \times \Omega)$, where $\Omega' = \Omega_1 \times \Omega_2$ is of too great dimension. Suppose $\zeta_1, \zeta_2, \ldots$ are an ON basis for $\Phi(\Omega_1)$. Consider $\psi_u = \psi_{xy}$ as a function of x, expand, and rearrange:

$$\begin{aligned}
\psi_{xy} &= \sum_k \zeta_k(x) \int_{\Omega_1} \zeta_k(x') \psi_{x'y} \mathrm{d}x' \\
&= \sum_k \zeta_k(x) \int_{\Omega} \int_{\Omega_1} \zeta_k(x') K_{x'yv} \mathrm{d}x' \phi_v \mathrm{d}v \\
&= \sum_k \zeta_k(x) [(\zeta_k K)\phi]_y.
\end{aligned}$$

Hence $\psi = \sum_k \zeta_k \wedge [(\zeta_k K)\phi]$. Let $J_k = \zeta_k K \in \Phi(\Omega_2 \times \Omega)$ and we can express the computation with lower dimensional fields:

$$L(\phi) = \sum_k \zeta_k \wedge J_k\phi.$$

Other approaches to reducing the dimension of fields are described elsewhere (MacLennan 1990).

The converse procedure, using field computation to implement a matrix vector product, is also useful, since a field computer may have better facilities for field computation than for computing with vectors. Therefore suppose M is an $m \times n$ matrix, $\mathbf{c} \in \mathbb{R}^n$, and that we want to compute $\mathbf{d} = \mathrm{M}\mathbf{c}$ by a field product $\psi = K\phi$.

The input vector will be represented by $\phi \in \Phi(\Omega)$, where we choose a field space $\Phi(\Omega)$ for which the first n ON basis elements $\beta_1, \ldots, \beta_n$ are physically realizable. The field representation is given by $\phi = \sum_{k=1}^{n} c_k \beta_k$. Analogously, the output is represented by a field $\psi \in \Phi(\Omega')$ given by $\psi = \sum_{j=1}^{m} d_k \zeta_k$, for ON basis fields $\zeta_1, \ldots, \zeta_m$. The required kernel $K \in \Phi(\Omega' \times \Omega)$ is given by

$$K = \sum_{j=1}^{m} \sum_{k=1}^{n} M_{ij}(\zeta_j \wedge \beta_k).$$

To see this, observe:

$$\begin{aligned} K\phi &= \sum_{jk} M_{jk}(\zeta_j \wedge \beta_k)\phi \\ &= \sum_{jk} M_{jk}\zeta_j \langle \beta_k \mid \phi \rangle \\ &= \sum_{j} \zeta_j \sum_{k} M_{jk} c_k \\ &= \sum_{j} \zeta_j d_j. \end{aligned}$$

5.3 Diffusion Processes

Diffusion processes are useful in both natural and artificial intelligence. For example, it has been applied to path planning through a maze (Steinbeck, Tóth & Showalter 1995) and to optimization and constraint-satisfaction problems, such as occur in image processing and motion estimation (Miller, Roysam, Smith & O'Sullivan 1991, Ting & Iltis 1994). Natural systems, such as developing embryos and colonies of organisms, use diffusion as a means of massively parallel search and communication.

A simple diffusion equation has the form $\dot{\phi} = d\nabla^2\phi$ with $d > 0$. On a continuous-time field computer that provides the Laplacian operator (∇^2) diffusion can be implemented directly by this equation. With sequential computation, the field will be iteratively updated in discrete steps:

$$\phi := \phi + d\nabla^2\phi.$$

If the Laplacian is not provided as a primitive operation, then its effect can be approximated by a spatial convolution with a suitable field ϱ (Sec. 5.1.3). In sequential computation we may use $\phi := (1-d)\phi + d\varrho \otimes \phi$, where ϱ is an appropriate two-dimensional Gaussian or similarly shaped field. In continuous time, we may use $\dot{\phi} = d\varrho \otimes \phi$, where $\varrho = \gamma'' \wedge \gamma + \gamma \wedge \gamma''$ (Eq. 9, Sec. 5.1.4), where γ is an appropriate one-dimensional Gaussian and γ'' is its second derivative (or similarly shaped fields).

Reaction-diffusion systems combine diffusion in two or more fields with local nonlinear reactions among the fields. A typical reaction-diffusion system over fields

$\phi^1, \ldots, \phi^n \in \Phi(\Omega)$ has the form:

$$\begin{aligned}
\dot{\phi^1} &= \overline{F_1}(\phi^1, \ldots, \phi^n) + d_1 \nabla^2 \phi^1, \\
\dot{\phi^2} &= \overline{F_2}(\phi^1, \ldots, \phi^n) + d_2 \nabla^2 \phi^2, \\
&\vdots \\
\dot{\phi^n} &= \overline{F_n}(\phi^1, \ldots, \phi^n) + d_n \nabla^2 \phi^n,
\end{aligned}$$

where the $d_k > 0$, and the local reactions $\overline{F_k}$ apply at each point $u \in \Omega$ of the fields: $F_k(\phi^1_u, \ldots, \phi^n_u)$. With obvious extension of the notation, this can be written as a differential equation on a vector field:

$$\dot{\boldsymbol{\phi}} = \overline{\mathbf{F}}(\boldsymbol{\phi}) + \mathrm{D}\nabla^2 \boldsymbol{\phi},$$

where $\mathrm{D} = \mathrm{diag}(d_1, \ldots, d_n)$ is a diagonal matrix of diffusion rates.

Embryological development and many other biological processes of self-organization are controlled by local reaction to multiple diffusing chemicals (e.g., Bar-Yam 1997, ch. 7, Solé & Goodwin 2000, ch. 3); these are examples of natural field computation, a subject pioneered by Turing (1952). For example, simple *activator-inhibitor systems* can generate *Turing patterns*, which are reminiscent of animal skin and hair-coat pigmentation patterns (e.g., Bar-Yam 1997, ch. 7). In the simplest case, these involve an activator (α) and an inhibitor (β), which diffuse at different rates, and a nonlinear interaction which increases both when $\alpha > \beta$, and decreases them otherwise. For example (Bar-Yam 1997, p. 668):

$$\begin{aligned}
\dot{\alpha} &= \frac{k_1 \alpha^2}{\beta(1 + k_5 \alpha^2)} - k_2 \alpha + d_\alpha \nabla^2 \alpha, \\
\dot{\beta} &= k_3 \alpha^2 - k_4 \beta + d_\beta \nabla^2 \beta.
\end{aligned}$$

Reaction-diffusion systems have been applied experimentally in several image-processing applications, where they have been used to restore broken contours, detect edges, and improve contrast (Adamatzky 2001, pp. 26–31). In general, diffusion accomplishes (high-frequency) noise filtering and the reaction is used for contrast enhancement.

A Adamatzky and his colleagues have used chemical implementation of reaction-diffusion systems to construct Voronoi diagrams around points and other two-dimensional objects (Adamatzky, De Lacy Costello & Asai 2005, ch. 2). Voronoi diagrams have been applied to collision-free path planning, nearest-neighbor pattern classification, and many other problems (Adamatzky, De Lacy Costello & Asai 2005, pp. 32–3). They also demonstrated a chemical field computer on a mobile robot to implement a reaction-diffusion path planning (Adamatzky, De Lacy Costello & Asai 2005, ch. 4).

Excitable media are an important class of reaction-diffusion system, which are found, for example, in the brain, cardiac tissue, slime mold aggregation, and many

other natural systems. In the simplest cases these comprise an *excitation field* ϵ and a *recovery field* ρ coupled by local nonlinear reactions:

$$\begin{aligned} \dot{\epsilon} &= \overline{F}(\epsilon, \rho) + d_\epsilon \nabla^2 \epsilon, \\ \dot{\rho} &= \overline{G}(\epsilon, \rho) + d_\rho \nabla^2 \rho. \end{aligned}$$

Typically $G(e, r)$ is positive for large e and negative for large r, while along the *nullcline* $F(e, r) = 0$, r has a roughly cubic dependence on e, with $F(e, r) < 0$ for large values of r and > 0 for small ones. The intersection of the nullclines defines the system's stable state, and small perturbations return to the stable state. However excitation above a threshold will cause the excitation to increase to a maximum, after which the system becomes first refractory (unexcitable), then partially excitable with an elevated threshold, and finally back to its excitable, resting state. Excitation spreads to adjacent regions, but the refractory property assures that propagation takes the form of a unidirectional wave of constant amplitude. Characteristic circular and spiral waves appear in two-dimensional media. Excitable media are useful for rapid, efficient communication. For example, masses of slime mold amoebas (*Dictyostelium discoideum*) act as an excitable medium in which the propagating waves accelerate aggregation of the amoebas into a mound (Solé & Goodwin 2000, pp. 21–4).

Many self-organizing systems and structures in biological systems involve reaction-diffusion processes, chemical gradients, excitable media, and other instances of field computation.

For example, Deneubourg (1977) has described the construction of equally-spaced pillars in termite nests in terms of three interrelated two-dimensional fields: ϕ, the concentration of cement pheromone in the air, σ, the amount of deposited cement with active pheromone, and τ the density of termites carrying cement (see also Bonabeau, Dorigo & Theraulaz 1999, pp. 188–93, Camazine, Deneubourg, Franks, Sneyd & Bonabeau 2001, pp. 399–400, and Solé & Goodwin 2000, pp. 151–7). The amount of deposited cement with pheromone increases as it is deposited by the termites and decreases as the pheromone evaporates into the air: $\dot{\sigma} = k_1\tau - k_2\sigma$. The pheromone in the air is increased by this evaporation, but also decays and diffuses at specified rates: $\dot{\phi} = k_2\sigma - k_4\phi + d_\phi\nabla^2\phi$. Laden termites enter the system at a uniform rate r, deposit their cement (k_1), wander a certain amount (modeled by diffusion at rate d_τ), but also exhibit *chemotaxis*, that is, motion up the gradient of pheromone concentration:

$$\dot{\tau} = r - k_1\tau + d_\tau\nabla^2\tau - k_5\nabla\cdot(\tau\overline{\times}\nabla\phi),$$

where $\overline{\times}$ represents the point-wise (local) product, $(\phi\overline{\times}\psi)_u = \phi_u\psi_u$. See Figure 1 for this model expressed as a field computation.

In addition to reaction-diffusion systems, chemical gradients, chemotaxis, and other field processes are essential to self-organization in morphogenesis, which can be understood in terms of field computation (Davies 2005).

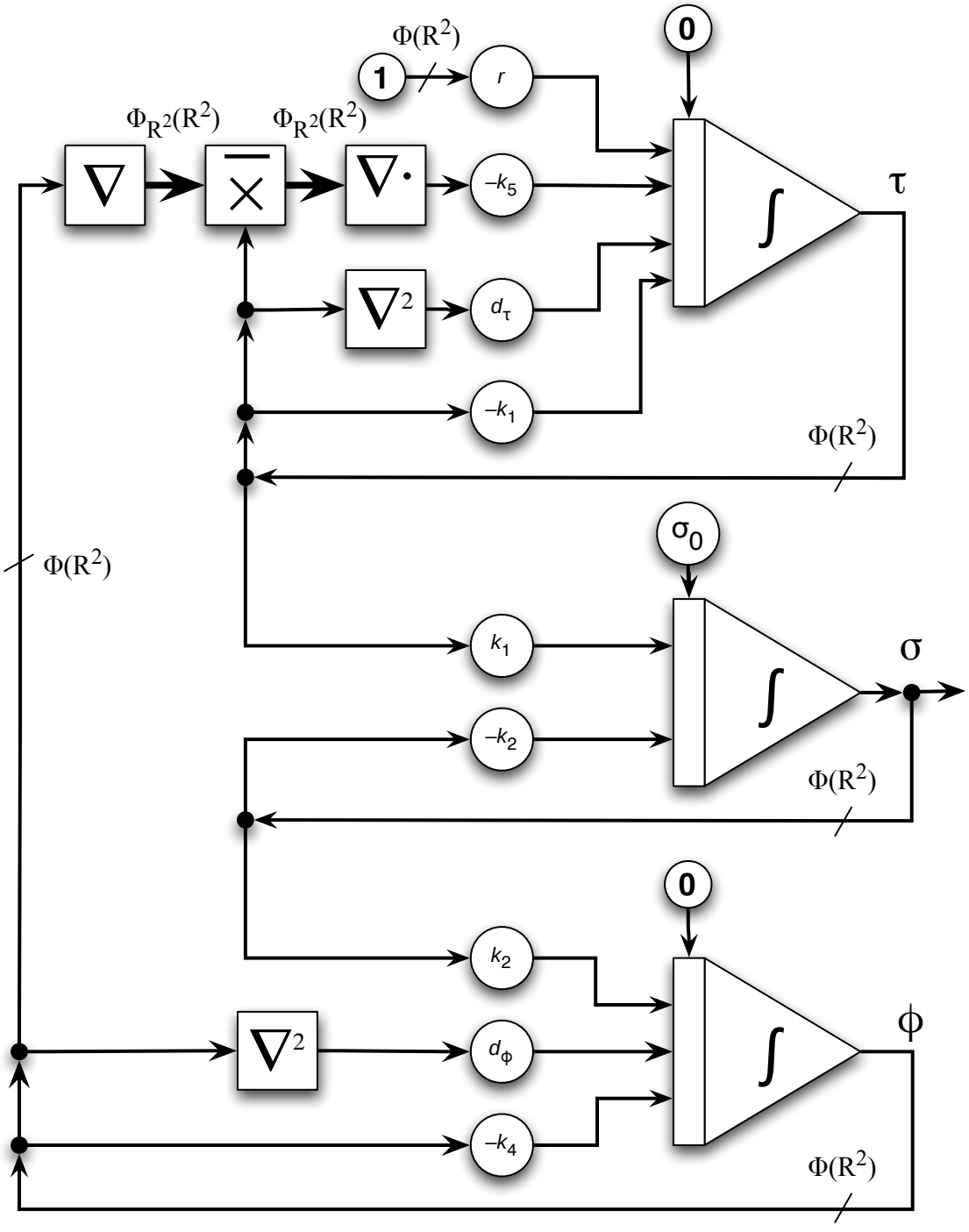

Figure 1: Field computation of Deneubourg's model of pillar construction by termites

5.4 Motion in Direction Fields

For an example of field computation in motor control, we may consider Georgopoulos (1995) explanation of the population coding of direction. In this case the feature space $\mathcal{D}$ represents directions in three-dimensional space, which we may identify with normalized three-dimensional vectors $\mathbf{d} \in \mathcal{D}$. Each neuron $u \in \Omega$ has a preferred direction $\boldsymbol{\eta}_u \in \mathcal{D}$ to which it responds most strongly, and it is natural to define u as the location in the map corresponding to this direction, $u = \mu(\boldsymbol{\eta}_u)$. However, Georgopoulos has shown that the direction is represented (more accurately and robustly) by a population code, in which the direction is represented by a neuronal field. Specifically, the activity ϕ_u of a neuron (above a base level) is proportional to the cosine of the angle between its preferred direction $\boldsymbol{\eta}_u$ and the direction $\mathbf{d}$ to be encoded. In particular, since the cosine of the angle between normalized vectors is equal to their scalar product, $\phi_u \propto \mathbf{d} \cdot \boldsymbol{\eta}_u$. A neurally plausible way of generating such a field is with a layer of radial basis functions (Sec. 5.1.2), $\phi_u = r(\|\mathbf{d} - \boldsymbol{\eta}_u\|)$, where $r(x) = 1 - x^2/2$; then $\phi_u = \mathbf{d} \cdot \boldsymbol{\eta}_u$ (MacLennan 1997).

Field computation is also used to update direction fields in the brain. For example, a remembered two-dimensional location, relative to the retina, must be updated when the eye moves (Droulez & Berthoz 1991*a*, Droulez & Berthoz 1991*b*). In particular, if the direction field ϕ has a peak representing the remembered direction, and the eye moves in the direction $\mathbf{v}$, then this peak has to move in the direction $-\mathbf{v}$ in compensation. More specifically, if $\mathbf{v}$ is a two-dimensional vector defining the direction of eye motion, then the change in the direction field is given by the differential field equation, $\dot{\phi} = \mathbf{v} \cdot \nabla\phi$, where the gradient is a two-dimensional vector field (retinal coordinates). (That is, $\partial\phi(\mathbf{d}, t)/\partial t = \mathbf{v} \cdot \nabla_{\mathbf{d}}\phi(\mathbf{d}, t)$.) To see this, note that *behind* the moving peak $\nabla\phi$ and $-\mathbf{v}$ point in the same direction, and therefore $(-\mathbf{v}) \cdot \nabla\phi$ is positive; hence $\dot{\phi}$ is negative. Conversely, $\dot{\phi}$ is positive in front of the peak. Each component of the gradient may be approximated by convolution with a derivative-of-Gaussian (DoG) field, in accord with Eq. 8, which can be computed by neurons with DoG receptive field profiles. (Additional detail can be found elsewhere (MacLennan 1997).)

Anderson (1995*b*) describes how transformations between retinal coordinates and head- or body-centered coordinates can be understood as transformations between field representations in area 7a of the posterior parietal cortex. For example, a minimum in a field may represent the destination of a motion (such as a saccade) in head-centered space, and then the gradient represents paths from other locations to that destination (MacLennan 1997). Further, the effects of motor neurons often correspond to vector fields (Bizzi & Mussa-Ivaldi 1995, Goodman & Anderson 1989).

5.5 Nonlinear Computation via Topographic Maps

As discussed in Secs. 4.1 and 5.4, the brain often represents scalar or vector quantities by topographic or computational maps, in which fields are defined over the range of

possible values and a particular value is represented by a field with a peak of activity at the corresponding location. That is, a value $x \in \Omega$ is represented by a field $\phi_x \in \Phi(\Omega)$ that is distinctly peaked at x. For mathematical convenience we can idealize ϕ_x as a Dirac delta function (unit impulse) centered at x: δ_x, where $\delta_x(u) = \delta(u - x)$. That is, δ_x is an idealized topographic representation of x.

For every function $f : \Omega \to \Omega'$, with $y = f(x)$, there is a corresponding linear transformation of a topographic representation of its input, $\delta_x \in \Phi(\Omega)$, into a topographic representation of its output, $\delta_y \in \Phi(\Omega')$. It is easy to show that the kernel $K \in \Phi(\Omega' \times \Omega)$ of this operation is

$$K = \int_\Omega \delta_{f(x)} \wedge \delta_x \mathrm{d}x,$$

which is essentially a *graph* of the function f. That is, we can compute an arbitrary, possibly *nonlinear* function $y = f(x)$ by a *linear* operation on the corresponding computational maps, $\delta_y = K\delta_x$.

To avoid the use of Dirac delta functions, we can expand them into generalized Fourier series; for example, $\delta_x = \sum_k \beta_k \langle \beta_k \mid \delta_x \rangle = \sum_k \beta_k \beta_k(x)$. This expansion yields

$$\begin{aligned} K &= \int_\Omega \left(\sum_j \zeta_j \zeta_j[f(x)] \right) \wedge \left(\sum_k \beta_k \beta_k(x) \right) \mathrm{d}x \\ &= \sum_{j,k} \zeta_j \wedge \beta_k \int_\Omega \zeta_j[f(x)] \beta_k(x) \mathrm{d}x \\ &= \sum_{j,k} \zeta_j \wedge \beta_k \langle \zeta_j \circ f \mid \beta_k \rangle, \end{aligned}$$

where $\zeta_j \circ f$ is the composition of ζ_j and f: $(\zeta_j \circ f)(x) = \zeta_j[f(x)]$. A physically realizable approximation to K is obtained by limiting the summations to finite sets of physically realizable basis functions. (This has the effect of blurring the graph of f.)

Computation on topographic maps has a number attractive advantages. These are simple mathematical consequences of the linearity of topographic computation, but it will be informative to look at their applications in neural information processing. For example, transformation of input superpositions compute superpositions of the corresponding outputs in parallel: $K(\delta_x + \delta_{x'}) = \delta_{f(x)} + \delta_{f(x')}$ (recall Sec. 4.1).

Since an input value is encoded by the position of the peak of a field rather than by its amplitude, the amplitude can be used for pragmatic characteristics of the input, such as its importance or certainty (see Sec. 5.7). These pragmatic characteristics are preserved by topographic computation, $K(p\delta_x) = p\delta_{f(x)}$. Therefore if we have two (or more) inputs $x, x' \in \Omega$ with corresponding pragmatic scale factors $p, p' \in \mathbb{R}$, then the corresponding outputs carry the same factors, $K(p\delta_x + p'\delta_{x'}) = p\delta_{f(x)} + p'\delta_{f(x')}$. For example, if the inputs are weighted by confidence or importance, then the corresponding outputs will be similarly weighted. Further, if several inputs generate the

same output, then their pragmatic scale factors will sum; for example if $f(x) = f(x')$, then $K(p\delta_x + p'\delta_{x'}) = (p + p')\delta_{f(x)}$. Thus, a number of inputs that are individually relatively unimportant (or uncertain) could contribute to a single output that is relatively important (or certain).

Finite superpositions of inputs are easily extended to the continuum case. For example, suppose that ϕ_x is the pragmatic scale factor associated with x, for all $x \in \Omega$ (for example, ϕ_x might be the probability of input x). We can think of the field ϕ as a continuum of weighted delta functions, $\phi = \int_\Omega \phi_x \delta_x \mathrm{d}x$. Applying the kernel to this field yields a corresponding continuum of weighted outputs, $K\phi = \int_\Omega \phi_x \delta_{f(x)} \mathrm{d}x \in \Phi(\Omega')$, where each point of the output field gives the total of the pragmatic scale factors (e.g., probabilities) of the inputs leading to the corresponding output value:

$$(K\phi)_y = \int_{\{x|y=f(x)\}} \phi_x \mathrm{d}x.$$

Therefore, by topographic computation, a transformation of an input probability distribution yields the corresponding output probability distribution.

We have remarked that the brain often uses *coarse coding*, in which a population of broadly-tuned neurons collectively represent a value with high precision (Sec. 5.1.1). If ϕ is the coarse coding of input x, then its maximum will be at x and its amplitude will decrease with distance from x, $\phi_u = r(\|u - x\|)$. Similarly, $K\phi$ will be a coarse coding of the output $f(x)$ induced by the coarse coding of the input. As discussed in Sec. 5.1.3, if all the neurons have the same receptive field profile ϱ, then the effect of coarse coding is a convolution or correlation of ϱ with the input map.

5.6 Gabor Wavelets and Coherent States

In 1946 D Gabor presented a theory of information based on application to arbitrary signals of the Heisenberg-Weyl derivation of the quantum mechanical Uncertainty Principle (Gabor 1946). Although he derived it for functions of time, it is easily generalizable to fields (square-integrable functions) over any finite-dimensional Euclidean space (reviewed elsewhere (MacLennan 1991)). Therefore, for $\Omega \subset \mathbb{R}^n$, let $\psi \in \Phi(\Omega)$ be an arbitrary (possible complex-valued) field (assumed, as usual, to have a finite norm, that is, to be square-integrable; see Sec. 3.1). To characterize this field's locality in space, we can measure its spread (or uncertainty) along each of the n spatial dimensions x_k by the root mean square deviation of x_k (assumed to have 0 mean):

$$\Delta x_k = \|x_k \psi(\mathbf{x})\| = \sqrt{\int_\Omega \psi^*_{\mathbf{x}} x_k^2 \psi_{\mathbf{x}} \mathrm{d}\mathbf{x}},$$

where $\mathbf{x} = (x_1, \ldots, x_n)^{\mathrm{T}} \in \mathbb{R}^n$. Consider also the Fourier transform $\Psi(\mathbf{u})$ of $\psi(\mathbf{x})$, the spread or uncertainty of which, in the frequency domain, can be quantified in a

similar way:

$$\Delta u_k = \|(u_k - \bar{u})\Psi(\mathbf{u})\| = \sqrt{\int_\Omega \Psi^*_{\mathbf{u}} u_k^2 \Psi_{\mathbf{u}} \mathrm{d}\mathbf{u}}.$$

It is straight-forward to show that the joint localization in any two conjugate variables (i.e., x_k in the space domain and u_k in the spatial-frequency domain) is limited by the *Gabor Uncertainty Principle*: $\Delta x_k \Delta u_k \geq 1/4\pi$.

This principle limits the information carrying capacity of any physically-realizable signal, so it is natural to ask if any function achieves the theoretical minimum, $\Delta x_k \Delta u_k = 1/4\pi$. Gabor showed that this minimum is achieved by what we may call the *Gabor elementary fields*, which have the form:

$$\Gamma_{\mathbf{pu}}(\mathbf{x}) = \exp\left[-\pi \|\mathrm{A}(\mathbf{x} - \mathbf{p})\|^2\right] \exp[2\pi i \mathbf{u} \cdot (\mathbf{x} - \mathbf{p})].$$

The second, imaginary exponential defines a plane wave originating at $\mathbf{p}$ with a frequency and direction determined by the *wave vector* $\mathbf{u}$. The first exponential defines a Gaussian envelope centered at $\mathbf{p}$ with a shape determined by the diagonal *aspect matrix* $\mathrm{A} = \mathrm{diag}(\alpha_1, \ldots, \alpha_n)$, which determines the spread of the function along each of the space and frequency axes:

$$\Delta x_k = \frac{\alpha_k}{2\sqrt{\pi}}, \qquad \Delta u_k = \frac{\alpha_k^{-1}}{2\sqrt{\pi}}.$$

Gaussian-modulated complex exponentials of this form correspond to the *coherent states* of quantum mechanics.

Each Gabor elementary field defines a cell in $2n$-dimensional "Gabor space" with volume $(4\pi)^{-n}$. He explained that these correspond to elementary units of information, which he called *logons*, since a field of finite spatial extent and bandwidth occupies a finite region in Gabor space, which determines its *logon content*. It may be computed by

$$N = \prod_{k=1}^{n} \frac{X_k}{\Delta x_k} \frac{U_k}{\Delta u_k} = (4\pi)^n \prod_{k=1}^{n} X_k U_k,$$

where X_k is the width of the field along the kth axis, and U_k its bandwidth on that axis, that is, a field's logon content is $(4\pi)^n$ times its Gabor-space volume.

The set of Gabor elementary functions are complete, and so any finite-energy function can be expanded into a series (Heil & Walnut 1989, pp. 656–7): $\psi = \sum_{k=1}^{N} c_k \Gamma_k$, where $\Gamma_1, \ldots, \Gamma_N$ are the Gabor fields corresponding to the cells occupied by ψ, and the c_k are complex coefficients. These N complex coefficients are the information conveyed by ψ, each corresponding to a logon or degree of freedom in the signal.

The Gabor elementary functions are not orthogonal, and so the coefficients cannot be computed by the inner product, $\langle \Gamma_k \mid \psi \rangle$. (They do form a *tight frame*, a very useful but weaker condition, under some conditions (Daubechies, Grossman & Meyer 1986, p. 1275); see MacLennan (1991) for additional discussion of the non-orthogonality

issue.) On the other hand, it is easy to find the coefficients by minimization of the approximation error (Daugman 1993). Let $\hat{\psi}(\mathbf{c}) = \sum_{k=1}^{N} c_k \Gamma_k$ and define the error $\mathcal{E} = \|\hat{\psi}(\mathbf{c}) - \psi\|^2$. This is a standard least-squares problem (cf. Sec. 5.10), which can be solved by matrix calculation or by gradient descent on the error surface. It is easy to show that $\partial\mathcal{E}/\partial c_k = 2\langle \Gamma_k \mid \hat{\psi}(\mathbf{c}) - \psi \rangle$, and therefore gradient descent is given by $\dot{c}_k = r\langle \Gamma_k \mid \psi - \hat{\psi}(\mathbf{c}) \rangle$ for some rate $r > 0$.

There is considerable evidence (reviewed elsewhere (MacLennan 1991)) that approximate Gabor representations are used in primary visual cortex, and there is also evidence that Gabor representations are used for generating motor signals (Pribram 1991, pp. 139–44, Pribram, Sharafat & Beekman 1984).

5.7 Information Fields

Hopfield (1995) observed that in some cases a neural impulse train can be understood as transmitting two signals: (1) the information content, encoded in the *phase* of the impulses relative to some global or local "clock," and (2) some other pragmatic characteristic of the information (such as importance, urgency, or confidence), encoded in the *rate* of the impulses. Such a combination of phase-encoded semantics and rate-encoded pragmatics may be common in the nervous system. Already in his *Laws of Thought* (1854), George Boole recognized idempotency as characteristic of information: repeating a message does not change its meaning, but it may affect its pragmatic import. The distinction is implicit in our typographic conventions; consider:

YES NO

YES NO

The horizontal distinction is semantic, but the vertical is pragmatic. More generally, following a distinction that has been made in quantum mechanics (Bohm & Hiley 1993, pp. 35–6), we may say that the *form* of the signal *guides* the resulting action, but its *magnitude* determines the *amount* of action.

Similarly in field computation it may be useful to represent information by a field's shape and pragmatics by its magnitude; that is, pragmatics depends on the total amount of "stuff," semantics on it disposition (also a holistic property). The magnitude of such an *information field* is given by its norm $\|\psi\|$, where we normally mean the inner-product norm of the Hilbert space, $\|\psi\|^2 = \langle \psi \mid \psi \rangle$ (which we can think of as "energy"), but other norms may be appropriate, depending on the relevant sense of the "amount" of action. The semantics of such fields is determined by their form, which we may identify with the normalization of the field, $N(\psi) = \psi/\|\psi\|$ (for nonzero fields). Idempotency is expressed by the identity $N(z\psi) = N(\psi)$ for all $z \neq 0$.

Therefore, it is reasonable that the entropy of a field depends on its shape, but

not its magnitude:

$$S(\psi) = \int_\Omega \frac{\psi_u}{\|\psi\|} \log \frac{\psi_u}{\|\psi\|} \mathrm{d}u = \int_\Omega N(\psi)_u \log N(\psi)_u \mathrm{d}u = \langle N(\psi) \mid \overline{\log} N(\psi) \rangle.$$

It is perhaps unsurprising that similar issues arise in quantum mechanics and field computation, for they are both formulated in the language of Hilbert spaces. For example, a quantum mechanical state ψ is taken to be undetermined with respect to magnitude, so that $z\psi$ is the same state as ψ for any nonzero complex number z (Dirac 1958, p. 17). Therefore, the state is conventionally taken to the normalized, $\|\psi\| = 1$, so that its square is a probability density function, $\rho_x = |\psi_x|^2$.

Independence of magnitude is also characteristic of the quantum potential, which led Bohm & Hiley (1993) to characterize this field as *active information*. For example, if we write the wave function in polar form, $\psi_x = R_x e^{iS_x/h}$, then the motion of a single particle is given by (Bohm & Hiley 1993, pp. 28–9):

$$\frac{\partial S_x}{\partial t} + \frac{(\nabla S_x)^2}{2m} + V_x + Q_x = 0,$$

where the quantum potential is defined:

$$Q_x = -\frac{\hbar^2}{2m} \frac{\nabla^2 R_x}{R_x}.$$

Since the Laplacian $\nabla^2 R_x$ is scaled by R_x, the quantum potential depends only on the *local form* of the wavefunction ψ, not on its magnitude. From this perspective, the particle moves under its own energy, but the quantum potential controls the energy.

5.8 Field Representations of Discrete Symbols

Quantum field theory treats discrete particles as quantized excitations of a field. This observation suggests analogous means by which field computation can represent and manipulate discrete symbols and structures, such as those employed in symbolic AI. It also provides potential models for neural representation of words and categories, especially in computational maps, which may illuminate how discrete symbol processing interacts with continuous image processing. From this perspective, discrete symbol manipulation is an emergent property of continuous field computation, which may help to explain the flexibility of human symbolic processes, such as language use and reasoning (MacLennan 1994*a*, 1994*b*, 1995).

Mathematically, discrete symbols have the *discrete topology*, which is defined by the *discrete metric*, for which the distance between any two distinct objects is 1: $d(x, x) = 0$ and $d(x, y) = 1$ for $x \neq y$. Therefore we will consider various field representations of symbols that have this property. For example, discrete symbols could be represented by localized, non-overlapping patterns of activity in a computational

map. In particular, symbols could be represented by Dirac delta functions, for which $\langle \delta_x \mid \delta_x \rangle = 1$ and $\langle \delta_x \mid \delta_y \rangle = 0$ for $x \neq y$. Here we may let $d(x, y) = 1 - \langle \delta_x \mid \delta_y \rangle$. More realistically, symbols could be represented by physically realizable normalized fields ϕ_x with little or no overlap between the representations of different symbols: $\langle \phi_x \mid \phi_y \rangle \approx 0$ for $x \neq y$. Indeed, any sufficiently large set of orthonormal fields may be used to represent discrete symbols. Fields may seem like an inefficient way to represent discrete symbols, and so it is worth observing that with at least 146 000 neurons per square millimeter, a one hundred thousand-word vocabulary could be represented in a few square millimeters of cortex.

Since the meaning of these fields is conveyed by the location of activity peak in the map, that is, by the shape of the field rather than its amplitude, the field's amplitude can be used for pragmatic scale factors, as previously discussed (Sec. 5.5). This could be used, for example, to convey the confidence or probability of a word or verbal category, or another pragmatic factor, such as loudness (cf. Sec. 5.7).

Wave packets (coherent states, Gabor elementary functions) are localized patterns of oscillation resulting from the superposition of a number of nonlocal oscillators with a Gaussian distribution of frequencies (MacLennan 1991). The relative phase of these oscillators determines the position of the wave packet within its field of activity. Therefore different phase relationships may determine field representations for different discrete symbols. The amplitude of the wave packet could represent pragmatic information, and frequency could be used for other purposes, for example for *symbol binding*, with bound symbols having the same frequency. Continuous phase control could be used to control the motion of wave packets in other representations, such as direction fields (Sec. 5.4).

5.9 Gradient Processes

Many optimization algorithms and adaptive processes are implemented by gradient ascent or gradient descent. Because of its physical analogies, it is more convenient to think of optimization as decreasing a *cost function* rather than increasing some *figure of merit*. For example, the function might represent the difficulty of a motor plan or the incoherence in an interpretation of sensory data (such as stereo disparity).

Therefore suppose that $U : \Phi(\Omega) \to \mathbb{R}$ is a functional that defines the undesirability of a field; the goal is to vary ϕ so that $U(\phi)$ decreases down a path of "steepest descent." (By analogy with physical systems, we may call U a *potential function* and think of gradient descent as a *relaxation process* that decreases the potential.) The change in the potential U is given by the chain rule for field transformations (Eq. 3):

$$\begin{aligned} \dot{U}(t) &= (U \circ \phi)'(t, 1) \\ &= U'[\phi(t)][\phi'(t)(1)] \\ &= \langle \boldsymbol{\nabla} U[\phi(t)] \mid \dot{\phi}(t) \rangle. \end{aligned}$$

More briefly, suppressing the dependence on time, $\dot{U} = \langle \boldsymbol{\nabla} U(\phi) \mid \dot{\phi} \rangle$. To guarantee

$\dot{U} \leq 0$ we let $\dot{\phi} = -r\boldsymbol{\nabla}U(\phi)$ with a rate $r > 0$ for gradient descent. Then,

$$\dot{U} = \langle \boldsymbol{\nabla}U(\phi) \mid \dot{\phi} \rangle = \langle \boldsymbol{\nabla}U(\phi) \mid -r\boldsymbol{\nabla}U(\phi) \rangle = -r\|\boldsymbol{\nabla}U(\phi)\|^2 \leq 0.$$

Therefore, gradient descent decreases U so long as the gradient is nonzero. (More generally, of course, so long as the trajectory satisfies $\langle \boldsymbol{\nabla}U(\phi) \mid \dot{\phi} \rangle < 0$ the potential will decrease.)

Often the potential takes the form of a *quadratic functional*:

$$U(\phi) = \phi K \phi + L\phi + c,$$

where $K \in \Phi(\Omega \times \Omega)$, $\phi K \phi = \int_\Omega \int_\Omega \phi_u K_{uv} \phi_v \mathrm{d}u\mathrm{d}v$, L is a linear functional, and $c \in \mathbb{R}$. We require the *coupling field* K to be symmetric: $K_{uv} = K_{vu}$ for all $u, v \in \Omega$; typically it reflects the importance of correlated activity between any two locations u and v in ϕ. By the Riesz Representation Theorem (Sec. 3.2) this quadratic functional may be written

$$U(\phi) = \phi K \phi + \langle \rho \mid \phi \rangle + c,$$

where $\rho \in \Phi(\Omega)$. The field gradient of such a functional is especially simple:

$$\boldsymbol{\nabla}U(\phi) = 2K\phi + \rho.$$

In many cases $\rho = \mathbf{0}$ and then gradient descent is a linear process: $\dot{\phi} = -rK\phi$.

This process can be understood as follows. Notice that $-K_{uv}$ decreases with the *coupling* between locations u and v in a field and reflects the inverse variation of the potential with the coherence of the activity at those sites (i.e., the potential measures lack of coherence). That is, if $K_{uv} > 0$ then the potential will be lower to the extent that activity at u *covaries* with activity at v (since then $-\phi_u K_{uv} \phi_v \leq 0$), and if $K_{uv} < 0$, the potential will be lower to the extent they *contravary*. Therefore, the gradient descent process $\dot{\phi} = -rK\phi$ changes ϕ_u to maximally decrease the potential in accord with the covariances and contravariances with other areas as defined by K: $\dot{\phi}_u = -r \int_\Omega K_{uv} \phi_v \mathrm{d}v$. The gradient descent will stop when it produces a field ϕ^* for which $-rK\phi^* = \mathbf{0}$, that is, a field in the *null space* of K (the set of all $\phi \in \Phi(\Omega)$ such that $K\phi = \mathbf{0}$.

5.10 Universal Approximation

A system of *universal computation* provides a limited range of facilities that can be programmed or otherwise set up to implement any computation in a large and interesting class. The most familiar example is the Universal Turing Machine (UTM), which can be programmed to emulate any Turing machine, and therefore can implement any (Church-Turing) computable function. While this model of universal computation has been important in the theory of digital computation, other models may be more relevant in for other computing paradigms (MacLennan 2003, 2004, 2007, 2008).

Models of universal computation are important for both theory and practice. First, they allow the theoretical power of a computing paradigm to be established. For example, what cannot be computed by a UTM cannot be computed by a Turing machine or by any computer equivalent to a Turing machine. Conversely, if a function in Church-Turing computable, then it can be computed on a UTM or any equivalent machine (such as a programmable, general-purpose digital computer). Second, a model of universal computation for a computing paradigm provides a starting point for designing a general-purpose computer for that paradigm. Of course, there are many engineering problems that must be solved to design a practical general-purpose computer, but a model of universal computation establishes a theoretical foundation.

In the context of field computing there are several approaches to universal computation. One approach to universal field computation is based on a kind of field polynomial approximation based on the Taylor series for field transformations (Sec. 3.3) (MacLennan 1987, 1990). Another approach relies on a variety of "universal approximation theorems" for real functions, which are themselves generalizations of Fourier-series approximation (Haykin 1999, pp. 208–9, 249–50, 264–5, 274–8, 290–4). To explain this approach we will begin with the problem of interpolating a field transformation $F : \Phi(\Omega) \to \Phi(\Omega')$ specified by the samples $F(\phi^k) = \psi^k$, $k = 1, \ldots, P$. Further, we require the interpolating function to have the form

$$\hat{\psi} = \sum_{j=1}^{H} r_j(\phi)\alpha_j,$$

for some H, where the $r_j : \Phi(\Omega) \to \mathbb{R}$ are fixed nonlinear functionals (real-valued field transformation), and the $\alpha_j \in \Phi(\Omega')$ are determined by the samples so as to minimize the sum-of-squares error defined by $\mathcal{E} = \sum_{k=1}^{P} \|\hat{\psi}^k - \psi^k\|^2$, where $\hat{\psi}^k = \sum_{j=1}^{H} r_j(\phi^k)\alpha_j$. (A regularization term can be added if desired (Haykin 1999, ch. 5).)

A field, as an element of a Hilbert space, has the same norm as the (infinite) sequence of its generalized Fourier coefficiants (with respect to some ON basis). Let $\zeta_1, \zeta_2, \ldots$ be a basis for $\Phi(\Omega')$, and we can compute the Fourier coefficients of $\hat{\psi}^k - \psi^k$ as follows:

$$\begin{aligned} \langle \zeta_i \mid \hat{\psi}^k - \psi^k \rangle &= \left\langle \zeta_i \middle| \sum_{j=1}^{H} r_j(\phi^k)\alpha_j - \psi^k \right\rangle \\ &= \left[\sum_{j=1}^{H} r_j(\phi^k) \langle \zeta_i \mid \alpha_j \rangle \right] - \langle \zeta_i \mid \psi^k \rangle \end{aligned}$$

Let $R_{kj} = r_j(\phi^k)$, $A_{ji} = \langle \zeta_i \mid \alpha_j \rangle$, and $Y_{ki} = \langle \zeta_i \mid \psi^k \rangle$. Then, $\langle \zeta_i \mid \hat{\psi}^k - \psi^k \rangle = \sum_{j=1}^{H} R_{kj}A_{ji} - Y_{ki}$. The fields may approximated arbitrarily closely by the first N Fourier coefficients, in which case R, A, and Y are ordinary matrices. Then $\|\hat{\psi}^k -$

$\psi^k\|^2 \approx \sum_{i=1}^N E_{ki}^2$, where $\mathrm{E} = \mathrm{RA} - \mathrm{Y}$. Therefore the approximate total error is $\hat{\mathcal{E}} = \sum_{k=1}^P \sum_{i=1}^N E_{ki}^2$, or $\hat{\mathcal{E}} = \|\mathrm{E}\|_F^2$ (the squared Frobenius norm).

This is a standard least-squares minimization problem, and, as is well known (Leon 1986, pp. 371–3), the error is minimized by $\mathrm{A} = \mathrm{R}^+\mathrm{Y}$, where R^+ is the *Moore-Penrose pseudoinverse* of the interpolation matrix R: $\mathrm{R}^+ = (\mathrm{R}^\mathrm{T}\mathrm{R})^{-1}\mathrm{R}^\mathrm{T}$. From A we can compute the required fields to approximate F: $\alpha_j = \sum_{i=1}^N A_{ji}\zeta_i$.

For universality, we require that the approximation error can be made arbitrarily small, which depends on the choice of the basis functionals r_j, as can be learned from multivariable interpolation theory. Therefore, we represent the input fields by their first M generalized Fourier coefficients, an approximation that can be made as accurate as we like. Let $\beta_1, \beta_2, \ldots$ be an ON basis for $\Phi(\Omega)$ and let $\mathbf{p}^M : \Phi(\Omega) \to \mathbb{R}^M$ compute this finite-dimensional representation: $p_j^M(\phi) = \langle \beta_j \mid \phi \rangle$. We will approximate $r_j(\phi) \approx s_j[\mathbf{p}^M(\phi)]$, for appropriate functions $s_j : \mathbb{R}^M \to \mathbb{R}$, $j = 1, \ldots, H$. That is, we are approximating the field transformation F by

$$F(\phi) \approx \sum_{j=1}^H s_j[\mathbf{p}^M(\phi)]\alpha_j.$$

Now let $S_{kj} = s_j[\mathbf{p}^M(\phi^k)]$, and we have corresponding finite-dimensional interpolation conditions $\mathrm{Y} = \mathrm{SA}$ with the best least-square solution $\mathrm{A} = \mathrm{S}^+\mathrm{Y}$.

Various universal approximation theorems tell us that, given an appropriate choice of basis functions $s_1, \ldots, s_H$, any continuous function $\mathbf{f} : \mathbb{R}^M \to \mathbb{R}^N$ can be approximated arbitrarily closely by a linear combination of these functions (Haykin 1999, pp. 208–9). That is, the error $\hat{\mathcal{E}} = \|\mathrm{SA} - \mathrm{Y}\|_F^2$ can be made as small as we like. Therefore, appropriate choices for the s_j imply corresponding choices for the basis functionals r_j.

For example, one universal class of basis functions has the form $s_j(\mathbf{x}) = s(\mathbf{w}_j \cdot \mathbf{x} + b_j)$, for any nonconstant, bounded, monotone-increasing continuous function s (Haykin 1999, p. 208). This form is common in artificial neural networks, where $\mathbf{w}_j$ is a vector of neuron j's input weights (connection strengths) and b_j is its bias. To find the corresponding basis functional, $r_j(\phi) = s_j[\mathbf{p}^M(\phi)]$, observe

$$\begin{aligned}
\mathbf{w}_j \cdot \mathbf{p}^M(\phi) + b_j &= \sum_{k=1}^M w_{jk} p_k^M(\phi) + b_j \\
&= \sum_{k=1}^M w_{jk} \langle \beta_k \mid \phi \rangle + b_j \\
&= \left\langle \sum_{k=1}^M w_{jk}\beta_k \,\middle|\, \phi \right\rangle + b_j.
\end{aligned}$$

Therefore, let $\varpi_j = \sum_{k=1}^M w_{jk}\beta_k$, and we see that a universal class of functionals has the form:

$$r_j(\phi) = s(\langle \varpi_j \mid \phi \rangle + b_j). \tag{10}$$

Thus, in this field analog of an artificial neuron, the input field ϕ is matched to the neuron's interconnection field ϖ_j.

Another universal class is the *radial basis functions*, $s_j(\mathbf{x}) = r(\|\mathbf{x} - \mathbf{c}^j\|)$, where the radial function r is monotonically decreasing, and the centers $\mathbf{c}^j$ are either fixed or dependent on the function to be approximated. A corresponding universal class of field functions has the form:

$$r_j(\phi) = r(\|\phi - \eta_j\|), \tag{11}$$

where each field $\eta_j = \sum_i c_i^j \zeta_i$ causes the maximal response of the corresponding basis function r_j. Furthermore, if we set $H = P$ and $\eta_j = \phi^j$, then the matrix R is invertible for a wide variety of radial functions r (Haykin 1999, pp. 264–5).

Thus familiar methods of universal approximation can be transferred to field computation, which reveals simple classes of field transformations that are universal. This implies that universal field computers can be designed around a small number of simple functions (e.g., field summation, inner product, monotonic real functions).

6 Field Computers

6.1 Structure

As previously explained (Sec. 1), fields do not have to be physically continuous in either variation or spatial extension (that is, in range or domain), so long as the discretization is sufficiently fine that a continuum is a practical approximation. Therefore, field computation can be implemented with ordinary serial or parallel digital computing systems (as it has been in the past). However, field computation has a distinctively different approach to information representation and processing; computation tends to be shallow (in terms of operations applied), but very wide, "massively parallel" in the literal sense of computing with an effectively continuous *mass* of processors. Therefore field computation provides opportunities for the exploitation of novel computing media that may not be suitable for digital computation. For example, as the brain illustrates how relatively slow, low precision analog computing devices can be used to implement intelligent information processing via field computation, so electronic field computers may exploit massive assemblages of low-precision analog devices, which may be imprecisely fabricated, located, and interconnected. Other possibilities are optical computing in which fields are represented by optical wavefronts, molecular computation based on films of bacteriorhodopsin or similar materials, chemical computers based on reaction-diffusion systems, and "free space computing" based on the interactions of charge carriers and electrical fields in homogeneous semiconductors (Sec. 6.3).

Field computation is a kind of analog computation, and so there are two principal time domains in which field computation can take place, sequential time and continuous time (MacLennan 2007, 2008). In *sequential* computation, operations take place

in discrete steps in an order prescribed by a program. Therefore, sequential field computation is similar to ordinary digital computation, except that the individual program steps may perform massively parallel analog field operations. For example, a field assignment statement, such as:

$$\psi := \phi + \psi;$$

updates that field variable ψ to contain the sum of ϕ and the previous value of ψ.

In *continuous-time computation* the fields vary continuously in time, generally according to differential equations in which time is the independent variable; this has been the mode of operation of most analog computers in the past. In this case, a simple dependence, such as $\psi = \phi + \chi$, is assumed to have an implicit time parameter, $\psi(t) = \phi(t) + \chi(t)$, which represents the real time of computation. Since continuous-time programs are often expressed by differential equations, these computers usually provide hardware for definite integration of functions with respect to time:

$$\psi(t) = \psi_0 + \int_0^t F[\phi(\tau)]\mathrm{d}\tau. \tag{12}$$

Continuous-time programs are expressed by circuit diagrams (variable-dependency diagrams) rather than by textual programs such as used in digital computer programming (see Figure 1 for an example).

Although special-purpose analog and digital computers are appropriate for many purposes, already in the first half of the twentieth century the value of general-purpose (programmable) digital and analog computers had been recognized (MacLennan 2007, 2008). Therefore it will be worthwhile to consider briefly the sort of facilities we may expect to find in a general-purpose field computer (whether operating in sequential or continuous time).

We have seen that the following facilities are sufficient for universal computation (Sec. 5.10): multiplication of fields by scalars, local (point-wise) addition of fields ($\psi_u = \phi_u + \chi_u$), and some means of computing appropriate basis functionals. Neural-net style functionals (Eq. 10) require inner product and any non-constant, bounded, monotone-increasing scalar function (i.e., a sigmoid function). Radial basis functionals (Eq. 11) require the norm (which can be computed with the inner product) and any non-constant, bounded, monotone-decreasing scalar function. (Point-wise subtraction can be implemented, of course, by scalar multiplication and point-wise addition.) These are modest requirements, and we can expect practical field computers to have additional facilities.

In addition, continuous-time field computers will implement definite integration with respect to time (Eq. 12), which is used to implement field processes defined by differential equations. The equations are implemented in terms of the operations required for universal computation or in terms of others, discussed next.

Additional useful operations for general-purpose field computing include matrix-vector style field products (Hilbert-Schmidt integral operators), outer product, convolution, cross-correlation, normalization, local (point-wise) product and quotient

($\psi_u = \phi_u \chi_u$, $\psi_u = \phi_u / \chi_u$), and various other local operations ($\overline{\log}$, $\overline{\exp}$, etc.). Operations on vector fields can be implemented by scalar field operations on the vector components (Cartesian or polar); in this manner, vector fields of any finite dimension can be processed. If vector fields and operations on them are provided by the hardware, then it is useful if these operations include conversions between scalar and vector fields (e.g., between vector fields and their Cartesian or polar coordinate fields). Other useful vector field operations include point-wise scalar products between vector fields ($\boldsymbol{\psi}_u = \boldsymbol{\phi}_u \cdot \boldsymbol{\chi}_u$), gradient ($\nabla$), Laplacian ($\nabla^2$), divergence ($\nabla \cdot$), and point-wise scalar-vector multiplication ($\boldsymbol{\psi}_u = \phi_u \boldsymbol{\chi}_u$). Scalar analog computation is a degenerate case of field computation (since scalars correspond to fields in $\Phi(\{0\})$), and so practical general-purpose field computers will include the facilities typical of analog computers (MacLennan 2007, 2008).

6.2 The Extended Analog Computer

One interesting proposal for a general-purpose field computer is the *Extended Analog Computer* (EAC) of LA Rubel, which was a consequence of his conviction that the brain is an analog computer (Rubel 1985). However, Rubel and others had shown that the existing model of a general-purpose analog computer (GPAC), the abstract *differential analyzer* defined by CE Shannon, had relatively severe theoretical limitations, and so it did not seem adequate as a model of the brain (MacLennan 2007, 2008)(Lipshitz & Rubel 1987, Pour-El 1974, Rubel 1988, Shannon 1941, Shannon 1993). Like Shannon's differential analyzer, the EAC is an abstract machine intended for theoretical investigation of the power of analog computation, not a proposal for a practical computer (Rubel 1993); nevertheless, some actual computing devices have been based on it.

The EAC is structured in a series of levels, each building on those below it, taking outputs from the lower layers and applying analog operations to them to produce its own outputs. The inputs to the lowest layer are a finite number of "settings," which can be thought of real-numbers (e.g., set by a continuously adjustable knob). This layer is able to combine the inputs with real constants to compute polynomials over which it can integrate to generate differentially algebraic functions; this layer is effectively equivalent to Shannon's GPAC. Each layer provides a number of analog devices, including "boundary-value-problem boxes," which can solve systems of PDEs subject to boundary conditions and other constraints. That is, these conceptual devices solve field computation problems. Although for his purposes Rubel was not interested in implementation, he did remark that PDE solvers might be implemented by physical processes that obeyed the same class of PDEs as the problem (e.g., using physical diffusion to solve diffusion problems). This of course is precisely the old field analogy method, which was also used in network analyzers (recall Sec. 2). Rubel was able to show that the EAC is able to solve an extremely large class of problems, but the extent of its power has not been determined (MacLennan 2007, 2008).

6.3 Field Computing Hardware

Research in field computing hardware is ongoing and a comprehensive survey is beyond the scope of this article; a few examples must suffice.

Although the EAC was intended as a conceptual machine (for investigating the limits of analog computing), JW Mills has demonstrated several hardware devices inspired by it (Mills 1996, Mills, Himebaugh, Kopecky, Parker, Shue & Weilemann 2006). In these the diffusion of electrons in bulk silicon or conductive gels is used to solve diffusion equations subject to given boundary conditions, an technique he describes as "computing with empty space." This approach, in which a physical system satisfying certain PDEs is used to solve problems involving similar PDEs, is a contemporary version of the "field analogy method" developed by Kirchhoff and others (Sec. 2).

Adamatzky and his colleagues have investigated chemical field computers for implementing reaction-diffusion equations (Adamatzky 2001, Adamatzky, De Lacy Costello & Asai 2005); see Sec. 5.3. These use variants of the Belousov-Zhabotinsky Reaction and similar chemical reactions. Although the chemical reactions proceed relatively slowly, they are massively parallel: at the molecular level ("molar parallelism"). Also, Adamatzky, De Lacy Costello & Asai (2005, chs. 6–8) have designed both analog and digital electronic reaction-diffusion computers.

M Perŭs and his colleagues have investigated the use of quantum holography to implement field analogues of neural-network algorithms (Loo, Peruš & Bischof 2004, Peruš 1998).

Several investigators have explored optical implementations of field computers. For example, Skinner, Behrman, Cruz-Cabrera & Steck 1995 used self-lensing media, which respond nonlinearly to applied irradience, to implement feed-forward neural networks trained by back-propagation. Tõkés et al. (Tõkés, Orzó & Ayoub 2003, Tõkés, Orzó, Váró, Dér, Ormos & Roska 2001) have been developing an optical field computer using bacteriorhodopsin as a medium.

7 Future Directions

In the future field computation can be expected to provide an increasingly important analytical and intuitive framework for understanding massively parallel analog computation in natural and artificial intelligence.

First, field computation will provide a theoretical framework for understanding information processing in the brain in terms of cortical maps and, more generally, at a level between anatomical structures and individual neurons or small neural circuits. This will require improved understanding of information processing in terms of field computation, which will benefit from cognitive neuroscience research, but also contribute new computational concepts to it. Increased understanding of neural field computation will improve our ability to design very large artificial neural net-

works, which will be more attractive as massively parallel neurocomputing hardware is developed.

Traditionally, artificial intelligence has approached knowledge representation from the perspective of discrete, language-like structures, which are difficult to reconcile with the massively parallel analog representations found in the cortex. Therefore field computation will provide an alternative framework for understanding knowledge representation and inference, which will be more compatible with neuroscience but also provide a basis for understanding cognitive phenomena such as context sensitivity, perception, sensorimotor coordination, image-based cognition, analogical and metaphorical thinking, and nonverbal intelligences (kinesthetic, emotional, aesthetic, etc.).

As we have seen, concepts from field computation may be applied to understanding the collective intelligence of large groups of organisms. This approach permits separating the abstract computational principles from the specifics of their realization by particular organisms, and therefore permits their application to other organisms or artificial systems. For example, principles of field computation governing the self organization of groups of organisms are applicable to distributed robotics; in particular, they will provide a foundation for controlling very large population of microrobots or nanobots.

Embryological morphogenesis is naturally expressed in terms of field computation, since the differentiation and self-organization of an (initially homogeneous) cell mass is governed by continuous distributions of continuous quantity. Therefore, field computation provides a vehicle for rising above the specifics of particular signaling molecules, mechanisms of cell migration, etc. in order to understand development in abstract or formal terms. Understanding morphogenesis in terms of field computation will facilitate applying its principles to other systems in which matter self-organizes into complex structures. In particular, field computation will suggest means for programming the reorganization of matter for nanotechnological applications and for describing the behavior of adaptive "smart" materials.

As we approach the end of Moore's Law (Moore 1965), future improvements in computing performance will depend on developing new computing paradigms not based in sequential digital computation (see also MacLennan 2007, 2008). Improvements in both speed and density can be achieved by matching data representations and computational operations to the physical processes that realize them, which are primarily continuous and parallel in operation. Indeed, many of these processes are described in terms of fields or involve physical fields (i.e., phenomenological or physical fields). Therefore field computation points toward many physical processes that might be used for computation and provides a framework for understanding how best to use them. Thus we anticipate that field computation will play an important role in post-Moore's Law computing.

References

Adamatzky, A. 2001. *Computing in Nonlinear Media and Automata Collectives*. Bristol: Institute of Physics Publishing.

Adamatzky, A, B De Lacy Costello & T Asai. 2005. *Reaction-Diffusion Computers*. Amsterdam: Elsevier.

Anderson, JA. 1995*a*. *An Introduction to Neural Networks*. Cambridge, MA: MIT Press.

Anderson, RA. 1995*b*. Coordinate Transformations and Motor Planning in Posterior Parietal Cortex. In *The Cognitive Neurosciences*, ed. MS Gazzaniga. MIT Press pp. 519–32.

Bar-Yam, Y. 1997. *Dynamics of Complex Systems*. Reading, MA: Perseus Books.

Bizzi, E & FA Mussa-Ivaldi. 1995. Toward a Neurobiology of Coordinate Transformation. In *The Cognitive Neurosciences*, ed. MS Gazzaniga. MIT Press pp. 495–506.

Bohm, D & BJ Hiley. 1993. *The Undivided Universe: An Ontological Interpretation of Quantum Theory*. Routledge.

Bonabeau, E, M Dorigo & G Theraulaz. 1999. *Swarm Intelligence: From Natural to Artificial Systems*. Santa Fe Institute Studies in the Sciences of Complexity New York: Oxford University Press.

Brachman, G & L Narici. 1966. *Functional Analysis*. New York: Academic Press.

Camazine, S, J-L Deneubourg, NR Franks, G Sneyd, J Theraulaz & E Bonabeau. 2001. *Self-organization in Biological Systems*. Princeton.

Changeux, J-P. 1985. *Neuronal Man: The Biology of Mind*. Oxford: Oxford University Press. tr. by L. Garey.

Daubechies, I, A Grossman & Y Meyer. 1986. "Painless Non-orthogonal Expansions." *Journal of Mathematical Physics* 27:1271–83.

Daugman, JG. 1993. An Information-Theoretic View of Analog Representation in Striate Cortex. In *Computational Neuroscience*, ed. EL Schwartz. Cambridge: MIT Press pp. 403–423.

Davies, JA. 2005. *Mechanisms of Morphogensis*. Amsterdam: Elsevier.

Deneubourg, JL. 1977. "Application de l'ordre par Fluctuation à la Description de Certaines étapes de la Construction du nid chez les Termites." *Insectes Sociaux* 24:117–30.

Dirac, PAM. 1958. *The Principles of Quantum Mechanics*. fourth ed. Oxford: Oxford University Press.

Droulez, J & A Berthoz. 1991*a*. The Concept of Dynamic Memory in Sensorimotor Control. In *Motor Control: Concepts and Issues*, ed. D. R. Humphrey & H.-J. Freund. Wiley pp. 137–161.

Droulez, J & A Berthoz. 1991*b*. "A Neural Network Model of Sensoritopic Maps with Predictive Short-Term Memory Properties." *Proc. National Acad. Science USA* 88:9653–9657.

Feldman, JA & DH Ballard. 1982. "Connectionist Models and their Properties." *Cognitive Science* 6(3):205–54.

Gabor, D. 1946. "Theory of Communication." *Journal of the Institution of Electrical Engineers* 93, Part III:429–57.

Georgopoulos, AP. 1995. Motor Cortex and Cognitive Processing. In *The Cognitive Neurosciences*. MIT Press pp. 507–517.

Goodman, SJ & RA Anderson. 1989. "Microstimulation of a Neural-Network Model for Visually Guided Saccades." *Journal of Cognitive Neuroscience* 1:317–26.

Haykin, S. 1999. *Neural Networks: A Comprehensive Foundation*. 2nd ed. Upper Saddle River: Prentice-Hall.

Heil, CE & DF Walnut. 1989. "Continuous and Discrete Wavelet Transforms." *SIAM Review* 31(4):628–66.

Hopfield, JJ. 1995. "Pattern Recognition Computation Using Action Potential Timing for Stimulus Response." *Nature* 376:33–6.

Kirchhoff, G. 1845. "Ueber den Durchgang eines elektrischen Stromes durch eine Ebene, insbesondere durch eine kreisförmige." *Annalen der Physik und Chemie* 140/64(4):497–514.

Knudsen, EJ, S du Lac & SD Esterly. 1987. "Computational Maps in the Brain." *Ann. Rev. of Neuroscience* 10:41–65.

Leon, SJ. 1986. *Linear Algebra with Applications*. 2nd ed. New York: Macmillan.

Light, WA. 1992. Ridge Functions, Sigmoidal Functions and Neural Networks. In *Approximation Theory VII*, ed. EW Cheney, CK Chui & LL Schumaker. Boston: Academic Press pp. 163–206.

Lipshitz, L & LA Rubel. 1987. "A Differentially Algebraic Replacment Theorem." *Proceedings of the American Mathematical Society* 99(2):367–72.

Loo, CK, M Peruš & H Bischof. 2004. "Associative Memory Based Image and Object Recognition by Quantum Holography." *Open Systems & Information Dynamics* 11(3):277–89.

MacLennan, BJ. 1987. Technology-independent Design of Neurocomputers: The Universal Field Computer. In *Proceedings of the IEEE First International Conference on Neural Networks*, ed. M. Caudill & C.Butler. Vol. 3 IEEE Press pp. 39–49.

MacLennan, BJ. 1990. Field Computation: A Theoretical Framework for Massively Parallel Analog Computation, Parts I–IV. Technical Report CS-90-100 Department of Computer Science, University of Tennessee, Knoxville. Available from `www.cs.utk.edu/~mclennan`.

MacLennan, BJ. 1991. Gabor Representations of Spatiotemporal Visual Images. Technical Report CS-91-144 Department of Computer Science, University of Tennessee, Knoxville. Available from `www.cs.utk.edu/~mclennan`.

MacLennan, BJ. 1993. Information Processing in the Dendritic Net. In *Rethinking Neural Networks: Quantum Fields and Biological Data*, ed. Karl H. Pribram. Hillsdale, NJ: Lawrence Erlbaum pp. 161–197.

MacLennan, BJ. 1994*a*. Continuous Computation and the Emergence of the Discrete. In *Origins: Brain & Self-Organization*, ed. Karl H. Pribram. Hillsdale, NJ: Lawrence Erlbaum pp. 121–151.

MacLennan, BJ. 1994*b*. Image and Symbol: Continuous Computation and the Emergence of the Discrete. In *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration*, ed. Vasant Honavar & Leonard Uhr. New York: Academic Press pp. 207–24.

MacLennan, BJ. 1995. Continuous Formal Systems: A Unifying Model in Language and Cognition. In *Proceedings of the IEEE Workshop on Architectures for Semiotic Modeling and Situation Analysis in Large Complex Systems*. Monterey, CA: pp. 161–172.

MacLennan, BJ. 1997. Field Computation in Motor Control. In *Self-Organization, Computational Maps and Motor Control*, ed. Pietro G. Morasso & Vittorio Sanguineti. Elsevier pp. 37–73.

MacLennan, BJ. 2003. "Transcending Turing Computability." *Minds and Machines* 13:3–22.

MacLennan, BJ. 2004. "Natural Computation and Non-Turing Models Of Computation." *Theoretical Computer Science* 317:115–145.

MacLennan, BJ. 2007. A Review of Analog Computing. Technical Report UT-CS-07-601 Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville.

MacLennan, BJ. 2008. Analog Computation. In *Encyclopedia of Complexity and System Science*. Springer.

Mathematical Society of Japan. 1980. *Encyclopedic Dictionary of Mathematics*. Cambridge: MIT Press.

McClelland, JL, DE Rumelhart & the PDP Research Group. 1986. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 2: Psychological and Biological Models*. Cambridge, MA: MIT Press.

McFadden, J. 2002. "Synchronous Firing and its Influence on the Brain's Electromagnetic Field: Evidence for an Electromagnetic Field Theory of Consciousness." *Journal of Consciousness Studies* 9(4):23–50.

Miller, MI, B Roysam, KR Smith & JA O'Sullivan. 1991. "Representing and Computing Regular Languages on Massively Parallel Networks." *IEEE Transactions on Neural Networks* 2:56–72.

Mills, JW. 1996. The Continuous Retina: Image Processing with a Single-Sensor Artificial Neural Field Network. In *Proceedings IEEE Conference on Neural Networks*. IEEE Press.

Mills, JW, B Himebaugh, B Kopecky, M Parker, C Shue & C Weilemann. 2006. "Empty Space" Computes: The Evolution of an Unconventional Supercomputer. In *Proceedings of the 3rd Conference on Computing Frontiers*. New York: ACM Press pp. 115–26.

Moore, GE. 1965. "Cramming More Components Onto Integrated Circuits." *Electronics* 38(8):114–117.

Peruš, M. 1998. A Quantum Information-Processing "Algorithm" Based on Neural Nets. In *Joint Connference on Information Sciences*, ed. P Wang, G Georgiou, C Janikow & Y Yao. Vol. II Association for Intelligent Machinery pp. 197–200.

Pockett, S. 2000. *The Nature of Consciousness: A Hypothesis*. San Jose: Writers Club Press.

Pour-El, MB. 1974. "Abstract Computability and its Relation to the General Purpose Analog Computer (Some Connections Between Logic, Differential Equations and Analog Computers)." *Transactions of the American Mathematical Society* 199:1–29.

Powell, MJD. 1985. Radial Basis Functions for Multivariable Interpolation: A Review. In *IMA Conference on Algorithms for the Approximation of Functions and Data*. Shrivenham, UK: RMCS pp. 143–67.

Pribram, KH. 1991. *Brain and Perception: Holonomy and Structural in Figural Processing*. Hillsdale, NJ: Lawrence Erlbaum.

Pribram, KH, A Sharafat & GJ Beekman. 1984. Frequency Encoding in Motor Systems. In *Human Motor Actions: Bernstein Reassessed*, ed. HTA Whiting. Elsevier pp. 121–56.

Rubel, LA. 1985. "The Brain as an Analog Computer." *Journal of Theoretical Neurobiology* 4:73–81.

Rubel, LA. 1988. "Some Mathematical Limitations of the General-Purpose Analog Computer." *Advances in Applied Mathematics* 9:22–34.

Rubel, LA. 1993. "The Extended Analog Computer." *Advances in Applied Mathematics* 14:39–50.

Rumelhart, DE, JL McClelland & the PDP Research Group. 1986. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*. Cambridge, MA: MIT Press.

Sanger, TD. 1996. "Probability Density Estimation for the Interpretation of Neural Population Codes." *Journal of Neurophysiology* 76:2790–3.

Shannon, CE. 1941. "Mathematical Theory of the Differential Analyzer." *Journal of Mathematics and Physics of the Massachusetts Institute Technology* 20:337–354.

Shannon, CE. 1993. Mathematical Theory of the Differential Analyzer. In *Claude Elwood Shannon: Collected Papers*, ed. N. J. A. Sloane & Aaron D. Wyner. New York: IEEE Press pp. 496–513.

Skinner, SR, EC Behrman, AA Cruz-Cabrera & JE Steck. 1995. "Neural Network Implementation Using Self-Lensing Media." *Applied Optics* 34:4129–35.

Small, JS. 2001. *The Analogue Alternative: The electronic analogue computer in Britain and the USA, 1930–1975*. London & New York: Routledge.

Solé, R & B Goodwin. 2000. *Signs of Life: How Complexity Pervades Biology*. New York: Basic Books.

Soroka, WW. 1954. *Analog Methods in Computation and Simulation*. New York: McGraw-Hill.

Steinbeck, O, A Tóth & K Showalter. 1995. "Navigating Complex Labyrinths: Optimal Paths from Chemical Waves." *Science* 267:868–71.

Ting, P-Y & RA Iltis. 1994. "Diffusion Network Architectures for Implementation of Gibbs Samplers with Applications to Assignment Problems." *IEEE Transactions on Neural Networks* 5:622–38.

Tõkés, Sz, L Orzó & A Ayoub. 2003. Two-wavelength POAC (Programmable Optoelectronic Analogic Computer) using Bacteriorhodopsin as Dynamic Holographic Material. In *Proceedings of ECCTD '03 Conference*. Vol. 3 Krakow: pp. 97–100.

Tõkés, Sz, L Orzó, Gy Váró, A Dér, P Ormos & T Roska. 2001. Programmable Analogic Cellular Optical Computer using Bacteriorhodopsin as Analog Rewritable Image Memory. In *Bioelectronic Applications of Photochromic Pigments*, ed. A Dér & L Keszthelyi. Amsterdam, The Netherlands: IOS Press pp. 54–73.

Truitt, TD & AE Rogers. 1960. *Basics of Analog Computers*. New York: John F. Rider.

Turing, AM. 1952. "The Chemical Basis of Morphogenesis." *Philosophical Transactions of the Royal Society* B 237:37–72.

Books and Reviews

1. Bachman, G, and L Narici (1966) *Functional analysis*. Academic Press, New York.

2. Berberian, SK (1961) *Introduction to Hilbert space*. Oxford, New York.

3. MacLennan, BJ (1991) *Field computation: A theoretical framework for massively parallel analog computation, parts I–IV*. Technical Report CS-90-100, Dept. of Computer Science, University of Tennessee, Knoxville. Available from `http:www.cs.utk.edu/~mclennan`.

4. MacLennan, BJ *Foundations of Field Computation*. In preparation. Available from `http:www.cs.utk.edu/~mclennan`.