Field Computation
A Framework for Quantum Inspired Computing

Bruce J. MacLennan

Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville 37996, TN USA

Abstract

Fields are spatially continuous distributions of quantity, and field computation operates on fields in parallel. Examples of fields include neural activity across macroscopic regions of cortex and quantum wave functions, but also other massively parallel analog representations. By means of the mathematics of Hilbert spaces, field computation provides a common language for quantum computation and neural information processing, and thus for quantum inspired computational intelligence. This chapter presents the mathematical foundations of field computation, basic field transformations, examples of field computation and their relation to neural networks, dimension reduction for the sake of physical realizability, cortical information processing in terms of fields, connections to the uncertainty principle, and minimal operations for universal field computation and general purpose field computation.

Keywords: analog computation, computational maps, cortical processing, field computation, Hilbert space, neural networks, quantum computation, quantum inspired computation

1. Introduction

This chapter develops the theory of *field computation*, a model of computation that uses the mathematics of quantum mechanics to describe information representation and processing in massively parallel neural networks in both cortical tissue and neuromorphic computers. Field computation can be implemented in a variety of technologies, including massively parallel analog and digital electronics, full wavefront optical computing, and continuous-value quantum computing [1]. This chapter presents the basic concepts, definitions, and theorems of field computation, and discusses its applications in neuroscience, artificial intelligence, and quantum computing. (Prior and related work is cited throughout this chapter where it is relevant to the discussion.)

Informally, a *field* can be defined as a spatially continuous distribution of continuous quantity. Examples include real- and complex-valued scalar and vector fields, but we also consider fields with values in other continuous algebraic fields. In terms of physical realization, field computation includes both *physical fields*, which are mathematically continuous (e.g., electromagnetic and gravitational fields, quantum mechanical wave functions), and *phenomenological fields*, which are discrete, but sufficiently dense to be treated mathematically as though continuous (e.g., fluids, macroscopic objects, cortical tissue, massive arrays of discrete electrical components). Recent neuron counts reveal primate cortical densities ranging from 56,000 to 127,000 neurons per square millimeter [2]. Therefore, even a 0.1 mm^2 patch of human cerebral cortex might contain enough neurons to be treated as a field. A mathematical theory of cortical processing was the original motivation of the work described here [3], but the theory is more widely applicable, and in particular provides a mathematical foundation for quantum inspired artificial intelligence and quantum computing. The quantum wave function is a field, in the sense defined above, and a suitable medium for field computation, but fields can be implemented classically as well and used as a medium for quantum inspired computation.

Field computation may be defined as computation in which the state is represented by one or more physical or phenomenological fields [4]. In fact, it has a long history, for while most analog computers were used to integrate systems of ordinary differential equations, already in the nineteenth and early twentieth centuries systems of partial differential equations were being solved using the *field analogy method* [4, 5]. In these computers, fields were represented in continuous media such as electrolytic tanks or rubber sheets, or in dense arrays of discrete components such as capacitors.

2. Fields

Fields are described mathematically as continuous functions over (typically one-, two-, or three-dimensional) extended domains. If ϕ is a field over a domain Ω, then its value at a point $u \in \Omega$ can be written $\phi(u)$, but we usually use the subscript notation ϕ_u, which helps us to think of fields as continuously indexed vectors. Occasionally, it will be convenient to use Dirac's notation $\langle u \mid \phi \rangle$ for the value of ϕ at u. The value of a time-varying field ϕ at point u and time t is written $\phi_u(t)$, and the value of the field as a whole at a particular time t is written $\phi(t)$. Fields take their values in some algebraic field, usually the real or complex numbers, but also finite-dimensional vectors of real or complex numbers. (In general, we use upper and lower case Greek letters for fields, lower case Roman letters for scalars and vectors, and upper case Roman letters for functions, including field transformations.)

Physically realizable fields have additional properties. For example, they have a bounded range of variation: $|\phi_u| \leq b$ for some bound $b \in \mathbb{R}$ and all $u \in \Omega$. Typically their domains are finite in size, but sometimes we use fields that are extended in the time dimension. Physically realizable fields are usually *band-limited* because the physical media cannot sustain unlimited rates of spatial variation, and phenomenological (non-physical) fields have some finite discrete grain size. We write $\Phi_K(\Omega)$ for a space of physically realizable K-valued fields over the domain Ω; when the fields' codomain is clear from context, as is usually the case, we write $\Phi(\Omega)$.

Generally we treat fields as elements of a separable Hilbert space, in particular as square-integrable functions over Ω, and we treat a field space $\Phi_K(\Omega)$ as an appropriate subspace of the Hilbert space $\mathcal{H}(\Omega, K)$ (leaving the Lebesgue measure unspecified). Therefore, the mathematics of field computation is the same as the mathematics of quantum mechanics, and thus field computation is an example of quantum inspired computation. However, while field computation includes the usual apparatus of quantum computation (e.g., qubits and quantum gates), it goes beyond them by focusing on the greater information representation capacity of the spatially extended wave function, which is analogous to cortical information representations. Field computation also applies to large spatial arrays of discrete quantum systems such as quantum dots. Therefore field computation expands the familiar range of quantum computation and brings it closer to neural information processing.

Since a Hilbert space is an inner product space, we define the inner product of fields in the usual way. If ϕ and ψ are two real- or complex-valued fields with a common domain, $\phi, \psi \in \Phi(\Omega)$, then their inner product is:

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

where ϕ_u^* is the complex conjugate of ϕ_u. This inner product is extended to vector-valued fields, $\boldsymbol{\phi}, \boldsymbol{\psi} \in \Phi_{\mathbb{C}^n}(\Omega)$, as follows:

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

where $\boldsymbol{\phi}_u^\dagger$ is the conjugate transpose (adjoint) of the column vector $\boldsymbol{\phi}_u$. Given the inner-product, we define the norm $\|\phi\|$ in the usual way:

$$\|\phi\| = \sqrt{\langle \phi \mid \phi \rangle}.$$

The elements of a Hilbert space must have a finite norm (i.e., to be square-integrable): $\|\phi\| < \infty$ for all $\phi \in \Phi(\Omega)$. A quantum wave function is, of course, normalized: $\|\phi\| = 1$.

Occasionally we use the constant-valued fields $\mathbf{0}$ and $\mathbf{1}$, which are defined $\mathbf{0}_u = 0$ and $\mathbf{1}_u = 1$ for all $u \in \Omega$. We write $\mathbf{0}_\Omega$, $\mathbf{1}_\Omega$, etc. when it is necessary to be explicit about the field's domain. Conversely, since some field operations return or operate on a single real or complex number, it is useful to treat these numbers as degenerate fields, that is, as functions whose domains are singleton sets. For example, we define $\Phi^0 = \Phi_{\mathbb{C}}(\{0\})$ to be the field space of complex numbers. Because $\mathbb{C}$ and Φ^0 are isomorphic ($\mathbb{C} \cong \Phi^0$), we treat them as equivalent when confusion is unlikely.

3. Field Computation

Field computation is the application of *field transformations* to one or more fields in discrete or continuous steps. Since fields are functions in a Hilbert space, field transformations are operators on Hilbert spaces, which is also the mathematics of quantum mechanics. In short, since *quantum inspired computation* is computation over Hilbert spaces, field computation includes quantum inspired computation [6]. Quantum transformations are, of course, unitary, but field computation is not limited to quantum implementations, and so it permits the use of non-unitary and even non-linear operations. Moreover, any quantum process can be simulated with classical field computation (by classical computation in the Hilbert space), and therefore any local quantum computation can be implemented on a classical field computer, but it can be computationally very expensive to do so. Nevertheless, field computation provides an overarching framework for designing and programming hybrid classical-quantum computer systems.

In this section we define the principal operations of field computation. In a field computer these operations operate in parallel on entire fields, and thus they are constant-time operations. (Space complexity is addressed in terms of the required spatial bandwidth of the fields involved in the computation, or in terms of the size of the basis.)

The simplest field operations are *local transformations*, which apply the same function at each point of a field. That is, if the input field is $\phi \in \Phi_J(\Omega)$ and the function is $f : J \to K$, then the output field $\psi \in \Phi_K(\Omega)$ is defined $\psi_u = f(\phi_u)$. We use the notation $\overline{f} : \Phi_J(\Omega) \to \Phi_K(\Omega)$ for the local transformation, and therefore write $\psi = \overline{f}(\phi)$, but write $\psi = f(\phi)$ when confusion is unlikely.

Matrix-vector products are fundamental to the description of neural computation, and their extensions to the continuous domain are similarly fundamental, for they are simple linear operators. Let $\phi \in \Phi(\Omega)$ and $L \in \Phi(\Omega' \times \Omega)$ be square-integrable fields (which are continuous analogs of a vector and a matrix). We define the *field product* $L\phi = \psi \in \Phi(\Omega')$ by:

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

Although we call it a "product," it is not in general associative. Rather, it is an *integral operator of Hilbert-Schmidt type* with a *kernel* L. Because physically realizable fields are band-limited, physically realizable linear operators have a Hilbert-Schmidt kernel [7]. Unitary field products can be approximated by small set of fixed quantum operations [8, §4.5].

Let $L : \Phi(\Omega) \to \mathbb{C}$ be a continuous functional (scalar-valued linear operator). Then the Riesz Representation Theorem [e.g., 9, Sec. 12.4], says that L has a *representer*, which is a field $\rho \in \Phi(\Omega)$ such that $L\phi = \langle \rho \mid \phi \rangle$. If we use Dirac's bracket notation for fields, $|\rho\rangle = \rho$, then $L = \langle \rho|$, which is the dual of the field $|\rho\rangle$. This theorem applies to all field functionals, because linear operators are continuous if and only if they are bounded, and realizable field transformations are bounded (that is, there is a b such that $\|L\phi\| \leq b\|\phi\|$ for all $\phi \in \Phi(\Omega)$).

We extend the field product to multilinear operators in the obvious way. Let $\phi^k \in \Phi(\Omega_k)$, for $k = 1, \ldots, n$, and $M \in \Phi(\Omega' \times \Omega_n \times \cdots \Omega_2 \times \Omega_1)$. Then we define $M\phi^1\phi^2 \cdots \phi^n = \psi \in \Phi(\Omega')$:

$$\psi_u = \int_{\Omega_n} \cdots \int_{\Omega_2} \int_{\Omega_1} M_{uv_n \cdots v_2 v_1} \phi^1_{v_1} \phi^2_{v_2} \cdots \phi^n_{v_n} \mathrm{d}v_1 \mathrm{d}v_2 \cdots \mathrm{d}v_n.$$

In the absence of parentheses, these products associate to the left.

As in quantum mechanics, an *outer product* is also useful in field computation, but we define it a little differently so that it is more generally useful. Let $\phi \in \Phi(\Omega)$ and $\psi \in \Phi(\Omega')$; then the outer product $\phi \wedge \psi \in \Phi(\Omega \times \Omega')$ is defined $(\phi \wedge \psi)_{(u,v)} = \phi_u \psi_v$, for all $u \in \Omega$, $v \in \Omega'$. Since the space $\Phi(\Omega \times \Omega')$ is isomorphic to the space $\Phi(\Omega) \otimes \Phi(\Omega')$ of tensor products $\phi \otimes \psi$, we can treat the outer and tensor products as effectively identical, and in this chapter use the tensor product, which is more familiar from quantum mechanics. Note also that the field outer product $\phi \wedge \psi$ is the kernel of the dyad or Dirac outer product $|\phi\rangle\langle\psi^*|$. (The tensor product operator will never be elided, as is common in quantum mechanics, because that could be confused with a field product.)

The following relationships hold among the inner, outer (tensor), and field products for all $\phi, \chi \in \Phi(\Omega)$ and $\psi \in \Phi(\Omega')$:

$$\phi(\chi \otimes \psi) = \langle \phi^* \mid \chi \rangle \psi = \psi \langle \chi^* \mid \phi \rangle = (\psi \otimes \chi)\phi.$$

(Note that $\langle \phi^* \mid \chi \rangle = \langle \chi^* \mid \phi \rangle$ is a scalar.)

Two closely-related, useful field operators, which often have direct physical implementations, are cross-correlation and convolution. The *cross-correlation* of two fields over the same domain is defined:

$$(\psi \star \phi)_u = \int_\Omega \psi^*_{v-u} \phi_v \mathrm{d}v, \tag{1}$$

Therefore, $(\psi \star \phi)_u$ is an inner-product-based comparison of ϕ with ψ displaced by u; that is, $(\psi \star \phi)_u = \langle \psi_{-u} \mid \phi \rangle$, where ψ_{-u} is ψ displaced by u. Thus, cross-correlation compares the two fields (in an inner-product sense) in all possible relative positions and returns a field of those comparisons. The *convolution* of the two fields is similar:

$$(\psi * \phi)_u = \int_\Omega \psi_{u-v} \phi_v \mathrm{d}v. \tag{2}$$

It has simpler algebraic properties than the cross-correlation. Moreover, convolutional neural networks have proved very valuable in practical applications. With field computation hardware (e.g., optical implementations), convolution and cross-correlation are typically constant-time operations.

4. Derivatives of Field Transformations

It is useful to take the derivatives of field transformations, which are equivalent to the derivatives of operators over function spaces [10, §251G]. There are two alternative notions of differentiation of operators on Banach spaces, and therefore also on Hilbert spaces: the Fréchet and the Gâteaux derivatives. They are the same for field transformations due to the requirements of physical realizability [7]. We begin with the Fréchet differential, which is an operator in $\mathcal{L}(\Phi(\Omega), \Phi(\Omega'))$, the space of bounded linear operators from $\Phi(\Omega)$ to $\Phi(\Omega')$.

Let $F : \Phi(\Omega) \to \Phi(\Omega')$ be any field transformation and let U be an open subset of $\Phi(\Omega)$. Then $D \in \mathcal{L}(\Phi(\Omega), \Phi(\Omega'))$ is 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

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

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

For the same F and U, we define $\mathrm{d}F : \Phi(\Omega) \times \Phi(\Omega) \to \Phi(\Omega')$ to be the *Gâteaux derivative* of F if for all $\alpha \in U$ the following limit exists:

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

In general, if the Fréchet derivative exists, then it is identical to the Gâteaux, i.e., $\mathrm{d}F(\phi, \alpha) = F'(\phi)(\alpha)$ for all $\alpha \in \Phi(\Omega)$.

Since a Fréchet differential is a linear operator, it has a kernel field $K \in \Phi(\Omega' \times \Omega)$ satisfying $K\alpha = F'(\phi)(\alpha)$ for all α in $\Phi(\Omega)$. Therefore we define the *operator gradient* to be this kernel, $\boldsymbol{\nabla} F(\phi) = K$. The analog in field computation of a directional derivative is then defined:

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

Recalling that $\Phi^0 = \Phi(\{0\}) \cong \mathbb{C}$, we can see that the gradient of a functional $F : \Phi(\Omega) \to \Phi^0$ is the two-dimensional field $\boldsymbol{\nabla} F(\phi) \in \Phi(\{0\} \times \Omega)$. With slight abuse of notation we can define $\boldsymbol{\nabla} F(\phi) = \rho \in \Phi(\Omega)$, where ρ is the representer of $F'(\phi)$. Then, $F'(\phi)(\alpha) = \langle \rho \mid \alpha \rangle = \langle \boldsymbol{\nabla} F(\phi) \mid \alpha \rangle$.

Successively higher-order operator derivatives are operators of successively higher types. For example, if $F : \Phi(\Omega) \to \Phi(\Omega')$, then $\mathrm{d}F : \Phi(\Omega)^2 \to \Phi(\Omega')$, where $\Phi(\Omega)^2 = \Phi(\Omega) \times \Phi(\Omega)$. In general, $\mathrm{d}^n F : \Phi(\Omega)^{n+1} \to \Phi(\Omega')$. Similarly, as $F' : \Phi(\Omega) \to \mathcal{L}(\Phi(\Omega), \Phi(\Omega'))$, so $F'' : \Phi(\Omega) \to \mathcal{L}(\Phi(\Omega), \mathcal{L}(\Phi(\Omega), \Phi(\Omega')))$. In general,

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

Higher-order differentials are computed by products of "directional gradients":

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

The chain rules for Fréchet and Gâteaux derivatives completes this introduction to the derivatives of field transformations. Let $F : \Phi(\Omega') \to \Phi(\Omega'')$ and $G : \Phi(\Omega) \to \Phi(\Omega')$; then:

$$(F \circ G)'(\phi)(\alpha) = F'[G(\phi)][G'(\phi)(\alpha)], \quad (3)$$

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

In field computation we are often interested in the approximation of nonlinear field transformations, and for this purpose we can use a well-known analogue of Taylor's theorem for Hilbert spaces (in fact, for any Banach space), which allows us to express a nonlinear operator as a sum of multilinear operators.

Theorem 1 (Taylor). *Let U be any open subset of $\Phi(\Omega)$. Suppose that $F : \Phi(\Omega) \to \Phi(\Omega')$ is a field transformation and that its first n derivatives exist in U. Let $\phi \in U$ be the field around which we will expand F. If $\alpha \in \Phi(\Omega)$ is 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 the remainder term is

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

For notational convenience, we have defined $\mathrm{d}^0 F(\phi) = F(\phi)$.

The Taylor expansion can be expressed in terms of directional gradients as follows:

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

where we define $\boldsymbol{\nabla}_\alpha^k F(\phi) = \boldsymbol{\nabla}^k F(\phi) \alpha^{\otimes k}$, and where $\alpha^{\otimes k}$ is the k-fold tensor product:

$$\alpha^{\otimes k} = \overbrace{\alpha \otimes \alpha \otimes \cdots \otimes \alpha}^{k}.$$

This approximation is obviously analogous to a polynomial expansion, and we can strengthen the analogy by expressing the derivatives by products with coefficient fields. Therefore, define the fields $\Gamma_k = \boldsymbol{\nabla}^k F(\phi)$, and the approximation takes the form:

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

To avoid redundant computation of tensor powers of α, we can compute the approximation recursively by an analog of "Horner's rule": $F(\phi + \alpha) \approx G_0(\alpha)$, where

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

for $k = 0, \ldots, n-1$, $\Gamma_0 = F(\phi)$, and $G_n(\alpha) = 0$. This formulation facilitates computation by a series of neural-network-like layers.

5. Examples of Field Computation

In this section we present several examples of field computation with applications in artificial intelligence, neural modeling, and quantum computation. We begin with non-iterative computations, and in particular with the simplest case, a feed-forward pipeline of field transformations. In this case we apply a sequence of transformations $F_1, \ldots, F_n$ to an input field ϕ to yield an output field ψ: $\psi = F_n(\cdots F_1(\phi) \cdots)$. We can think of the output field ψ as a function of a particular input field ϕ, or if the input field is a signal changing in discrete or continuous time, $\phi(t)$, then the field transformation operates as a memoryless filter to produce the output signal $\psi(t)$: $\psi(t) = F_n(\cdots F_1(\phi(t)) \cdots)$.

5.1. Neural Network Style Computation

Multilayer feed-forward neural networks are good examples of this style of neural computation. In typical artificial neural networks, the activity y_i of neuron i in one layer is a simple function of the activities, $x_1, \ldots, x_n$, of the neurons in another layer. In general,

$$y_i = s\left(\sum_{j=1}^{N} W_{ij} x_j + b_i\right), \tag{5}$$

where W_{ij} is the weight (connection strength) to neuron i from neuron j, b_i is a *bias* (negative threshold), and $s : \mathbb{R} \to \mathbb{R}$ is a *sigmoid function*, that is, a non-decreasing, bounded continuous function.

Neural network style computation can be applied to continuous fields, such as images, or to dense arrays of neurons treated as phenomenological fields. In these cases we may speak of *neural fabrics* or *cortical arrays*. Suppose ϕ is the field representing the activity over an input neural region Ω and ψ is a field representing the activity over an output region Ω'. Then the activity ψ_u of neuron u ($u \in \Omega'$) is defined by the activities ϕ_v of the neurons v ($v \in \Omega$) by the integral equation:

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

where $L \in \Phi(\Omega' \times \Omega)$ is an *interconnection kernel* or *projection field* (representing the synaptic weights) and $\beta \in \Phi(\Omega')$ is a *bias field* (representing neural thresholds). With the addition of a sigmoid activation function we have $\psi = \overline{s}(L\phi + \beta)$.

Clearly, an N-layer deep neural network is implemented in the same way. Let $\phi_0 \in \Phi(\Omega_0)$ be the input field and $\phi_N \in \Phi(\Omega_N)$ be the output field. For $k = 1, \ldots, N$, let $L_k \in \Phi(\Omega_k \times \Omega_{k-1})$ be the interconnection kernel (projection field) from layer $k-1$ to layer k, and let $\beta_k \in \Phi(\Omega_k)$ are the bias field for layer k. Then $\phi_k = \overline{s}(L_k \phi_{k-1} + \beta_k)$.

5.2. Discrete Basis Function Networks

Many cortical neurons are broadly tuned, which means that they respond maximally to a certain input pattern or sensory stimulus, but their response falls off gradually for other inputs (its *receptive field profile*). This results in a representation of an input signal in terms of the joint activity of a population of broadly tuned neurons. If the number of neurons in the representation is small, we may talk of a discrete set of (possibly nonorthogonal) basis fields. One important class of these are *radial basis function networks* (RBF networks) [11, 12].

Suppose we have a set of radial basis functions, $r_1, r_2, \ldots, r_N$, where $r_j : \Phi(\Omega) \to [0, 1]$. This means that each function responds maximally to a particular input field η_j, its optimal input. Its value $r_j(\phi)$ represents how close ϕ is to its optimal input η_j. In many cases the RBFs have an identical receptive field profile $r : [0, \infty) \to [0, 1]$, which maps distance into neural activity, $r_j(\phi) = r(\|\phi - \eta_j\|)$. We assume that r is monotonically decreasing with $r(0) = 1$ and $r(x) \to 0$ as $x \to \infty$.

The inner product functions as a measure of similarity, as we can see by expanding the squared difference:

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

In many cases the input and focal fields are real-valued and normalized ($\|\phi\| = 1 = \|\eta_j\|$; see Sec. 7.1), in which case there is a simple inverse relation between squared distance and inner product: $\|\phi - \eta_j\|^2 = 2 - 2\langle \phi \mid \eta_j \rangle$. (Quantum states are of course normalized by definition.)

Since for real-valued normalized fields $\langle \phi \mid \eta_j \rangle \in [-1, 1]$, we can define RBFs with an identical response curve in terms of a fixed monotonically increasing function $s : [-1, 1] \to [0, 1]$ applied to the inner product, $r_j(\phi) = s(\langle \phi \mid \eta_j \rangle)$, which equals 1 when $\phi = \eta_j$ and equals 0 when the fields are maximally different ($\phi = -\eta_j$). This is a special case of neural-network style computation, in which s is the sigmoidal activation function (cf., Eq. 5). For complex-valued normalized fields, we use $r_j(\phi) = s(\langle \phi \mid \eta_j \rangle - \langle \phi \mid \eta_j^* \rangle)$, which makes use of conjugate weight fields η_j and η_j^*. Radial basis functions model *coarse coding* in the nervous system in which populations of broadly-tuned neurons collectively represent an input pattern [e.g., 13, 14, Vol. 1, pp. 91–6]. We can define this population code explicitly as a finite-dimensional vector function $\mathbf{p}(\phi)$ given by $p_j(\phi) = r_j(\phi)$.

Hilbert function spaces are isomorphic to ℓ_2, the space of square-summable sequences; in particular, there is a one-to-one correspondence between fields and the (square summable) infinite sequences of their generalized Fourier coefficients. Therefore, let $|1\rangle, |2\rangle, \ldots$ be an orthonormal (ON) basis of $\Phi(\Omega)$ and define $\mathbf{p} : \Phi(\Omega) \to \ell_2$ by the projections $p_j(\phi) = \langle j \mid \phi\rangle$. In practice, we can use a finite-dimensional projection, since there is a dimension m that will achieve any desired degree of accuracy. We will use $\mathbf{p}^m : \Phi(\Omega) \to \mathbb{C}^m$ for an m-dimensional representation:

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

Equivalently, this is the (unnormalized) result of a quantum mechanical projection into the subspace spanned by $|1\rangle, |2\rangle, \ldots, |m\rangle$, which is obtained with probability $\|\mathbf{p}^m(\phi)\|^2$ by measuring in this subspace.

5.3. *Continua of Basis Functions*

One goal of field computation is to describe cortical maps with large numbers of neurons. In these cases, we may think of a spatially organized *continuum* of basis functions rather than a discrete array of them. Suppose that the input space is $\Phi(\Omega)$ and the output space is $\Phi(\Omega')$. At each point $u \in \Omega'$ we want to apply to the input a different functional $R(u, —)$, with a different focal field. Therefore, let $\phi \in \Phi(\Omega)$ and $\psi \in \Phi(\Omega')$, and suppose $R : \Omega' \times \Phi(\Omega) \to \Phi(\Omega')$. The output field is then given by $\psi_u = R(u, \phi)$. More specifically, suppose that ψ_u is computed by neural-network-style computation with a continuum of weight fields η^u, that is, $\psi_u = s(\langle \eta^u \mid \phi\rangle)$. The inner product is computed $\langle \eta^u \mid \phi\rangle = \int_\Omega (\eta^u_v)^* \phi_v \mathrm{d}v$. Therefore, define the field $\mathrm{H} \in \Phi(\Omega' \times \Omega)$ by $\mathrm{H}_{uv} = (\eta^u_v)^*$. The output field is computed by the following field computation: $\psi = \overline{s}(\mathrm{H}\phi)$.

An important special case occurs when each field in the continuum has the same shape, but centered at a different point $u \in \Omega$; that is, the fields η^u are defined $\eta^u_v = \varrho(v - u)$ and therefore the kernel is $\mathrm{H}_{uv} = \varrho^*(v - u)$. It is apparent that $\mathrm{H}\phi = \varrho \star \phi$, the cross-correlation of the receptive field with the input field (cf. Eq. 1). If the receptive field ϱ is even, $\varrho(-x) = \varrho(x)$, then $\psi = \overline{s}(\varrho^* * \phi)$, which is a simple convolutional neural network (cf. Eq. 2). In these cases, if the kernel H is unitary, then $\mathrm{H}\phi$ can be computed by quantum operations.

5.4. Approximations of Spatial Integration and Differentiation

In this section we describe field computation approximations of spatial integrals and derivatives by convolutions. For a simple example, suppose that $\phi \in \Phi([-1,1])$ is a one-dimensional field. The integral $\psi = \int \phi$, which is defined $\psi_x = \int_0^x \phi_y \mathrm{d}y$, can be computed by a convolution $\psi = \upsilon * \phi$ with the *Heaviside* or *unit step field* on $\mathbb{R}$:

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

Next consider the spatial derivative $\phi'_u = \mathrm{d}\phi_u/\mathrm{d}u$ for $\phi \in \Phi([-1,1])$. To derive its kernel we need the *Dirac delta function* or *unit impulse function*, which the probability density of an ideal (dimensionless) particle located at the origin. It is defined by the properties:

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

It is the derivative of the unit step function, $\delta(x) = \upsilon'(x)$, and satisfies the "sifting equation":

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

Therefore it is an identity for the convolution operation: $\phi = \delta * \phi$.

Next, we express the derivative as a convolution:

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

where δ' is the *unit doublet*, the derivative of the Dirac delta function. This function is zero everywhere except infinitesimally below the origin, where it equals $+\infty$, and infinitesimally above it, where it equals $-\infty$. Therefore, the derivative is computed by the convolution $\phi' = \delta' * \phi$.

Generalized functions, such as the unit impulse and unit doublet, are not physically realizable fields, but they can be approximated as closely as required, for example by impulses or doublets of narrow but nonzero width. Alternatively, we can approximate the delta function with a sufficiently sharp Gaussian field $\gamma_x = \sqrt{r/\pi}\exp(-rx^2)$. Using it instead of δ produces Gaussian-smoothed operations. Thus, Gaussian-smoothed sifting, $\phi \approx \gamma * \phi$, implements Gausian smoothing. The Gaussian-smoothed spatial derivative is computed by the convolution with the *derivative of Gaussian* field: $\phi' \approx \gamma' * \phi$, where $\gamma'_x = \mathrm{d}\gamma_x/\mathrm{d}x$.

Spatial and temporal differentiation must be used with caution, since they amplify high-frequency noise; in particular we have to be careful applying spatial differentiation to phenomenological fields with an underlying discrete physical structure, which constitutes high-frequency noise. Therefore, in practice differentiation is often composed with a low-pass filter, such as Gaussian smoothing, to eliminate this noise.

These same kernel techniques can be extended to partial spatial derivatives. For example, suppose that $\psi \in \Phi(\mathbb{R}^2)$ is a two-dimensional field. Convolution with $\delta' \otimes \delta$ will take the derivative along the first dimension, and convolution with $\delta \otimes \delta'$ will take it along the second. To see this:

$$\begin{aligned}
[(\delta' \otimes \delta) * \Psi](x, y) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (\delta' \otimes \delta)(x-u, y-v)\Psi(u, v)\mathrm{d}v\mathrm{d}u \\
&= \int_{-\infty}^{\infty} \delta'(x-u) \int_{-\infty}^{\infty} \delta(y-v)\Psi(u, v)\mathrm{d}v\mathrm{d}u \\
&= \int_{-\infty}^{\infty} \delta'(x-u)\Psi(u, y)\mathrm{d}u \\
&= \partial\Psi(x, y)/\partial x.
\end{aligned}$$

Obviously, the kernels $\gamma' \otimes \gamma$ and $\gamma \otimes \gamma'$ can be used to approximate these partial derivatives, and the technique extends to higher dimensions.

These methods of approximating spatial derivatives can be used to compute the gradient or divergence of a field. Since the gradient is a vector field, we define $\mathbf{i} \in \Phi_{\mathbb{R}^2}(\mathbb{R}^2)$ to be a constant vector field of unit vectors in the x direction,

$\mathbf{i}_{(x,y)} = (1,0)$, and similarly, $\mathbf{j}_{(x,y)} = (0,1)$. Then the gradient field is defined:

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

A Gaussian-smoothed approximation is:

$$\nabla\psi \approx [(\gamma' \otimes \gamma) * \psi]\mathbf{i} + [(\gamma \otimes \gamma') * \psi]\mathbf{j}. \tag{6}$$

Computing the divergence of a vector field depends on how the vector field is represented. For example, a vector field $\Psi \in \Phi_{\mathbb{R}^2}(\mathbb{R}^2)$ can be represented by a pair of scalar fields $\chi, \psi \in \Phi_{\mathbb{R}}(\mathbb{R}^2)$ that are its cartesian components, $\Psi = \chi\mathbf{i}+\psi\mathbf{j}$. The divergence is the sum of the partials of the individual components:

$$\nabla \cdot \Psi = (\delta' \otimes \delta) * \chi + (\delta \otimes \delta') * \psi \approx (\gamma' \otimes \gamma) * \chi + (\gamma \otimes \gamma') * \psi.$$

If the vector field is represented directly (i.e., as a physical vector field), then it can be separated into its scalar cartesian components and the divergence computed as above.

The Laplacian can be computed directly from the second derivatives. Consider the second derivative of a one-dimensional field: $\phi'' = \delta' * (\delta' * \phi) = (\delta' * \delta') * \phi = \delta'' * \phi$, where δ'' is the second derivative of the unit impulse (a "unit triplet"). Therefore, the Laplacian of a two-dimensional field is:

$$\nabla^2\psi = (\delta'' \otimes \delta + \delta \otimes \delta'') * \psi \approx (\gamma'' \otimes \gamma + \gamma \otimes \gamma'') * \psi, \tag{7}$$

where γ'' is the second derivative of the Gaussian. The latter, a so-called "Mexican hat function" (inverted in this case), has the center-surround receptive-field profile exhibited by many neurons.

5.5. Iterative Field Computation

Like conventional computations, many field computations are iterative. In the simplest case, field operations can be performed sequentially, as on a conventional computer, but operating on all field elements in parallel. Field programs look like ordinary digital computer programs, except that the variables hold fields and the operators are field operations. For example, an iterative diffusion algorithm might contain an assignment statement such as this, which updates a field variable ϕ and uses a field operator ∇^2 (with $d > 0$):

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

5.6. Field Differential Equations

More commonly, as in conventional analog computers, field computers operate in continuous time by integrating partial differential equations. For example, a simple first-order field-valued differential equation is written $\dot{\phi} = F(\phi)$, which is field-computation notation for $\partial\phi_u(t)/\partial t = F_u[\phi(t)]$. The computation is performed by continuous-time field integrators, which solve the initial-value problem $\phi(T) = \phi(0) + \int_0^T F[\phi(t)]\mathrm{d}t$.

Gradient ascent provides a useful example of field computation; it can be implemented by sequential or continuous iteration. Suppose we are beginning with an initial field ϕ^0 (say, a corrupted image) and we are given a functional $F : \Phi(\Omega) \to \mathbb{R}$, a "figure of merit," which expresses the "goodness" of an image. Then the image can be improved by gradient ascent either sequentially $[\phi := \phi + r\boldsymbol{\nabla}F(\phi)]$ or continuously $[\dot{\phi} = r\boldsymbol{\nabla}F(\phi)]$ by use of the field gradient $\boldsymbol{\nabla}$ (Sec. 4).

Diffusion has many applications in artificial intelligence, for example, optimization and constraint-satisfaction in motion estimation and image processing [15, 16], and breadth-first path planning [17], but diffusion is expensive to simulate on a sequential computer. A diffusion equation, such as $\dot{\phi} = d\nabla^2\phi$, can be implemented directly on a continuous-time field computer, in which the Laplacian operator is computed in constant time; sequential-time iteration ($\phi := \phi + d\nabla^2\phi$) is no different. As we have seen (Sec. 5.4, Eq. 7), the Laplacian operator can be approximated by convolution with a simple kernel, if it is not a primitive operation.

5.7. Reaction-diffusion Computation

Another important use of diffusion is in reaction-diffusion computing [18, 19]. In these systems, fields are realized in (generally two-dimensional) distributions of chemical concentration, but they also can be implemented in other physical systems. Chemical reaction-diffusion systems have been applied to image processing [18, pp. 26–31] and to collision-free and shortest-path planning [19, ch. 2]. In general, we have n chemical concentration fields $\phi^1, \ldots, \phi^n \in \Phi(\Omega)$, each represented by a different chemical species. The system is defined by a system of PDEs such as these:

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

where the d_k are positive diffusion rates, and the local reactions $\overline{F_k}$ occur at each point $u \in \Omega$ of the fields: $F_k(\phi^1_u, \ldots, \phi^n_u)$. This system can be expressed compactly in terms of vector fields: $\dot{\boldsymbol{\phi}} = \overline{\mathbf{F}}(\boldsymbol{\phi}) + \mathrm{D}\nabla^2\boldsymbol{\phi}$, where $\mathrm{D} = \mathrm{diag}(d_1, \ldots, d_n)$ is a diagonal matrix of the diffusion constants. Field computation can be used to simulate reaction-diffusion systems modeling the behavior of massive swarms of agents [4].

6. Change of Field Domain

Mathematically, fields can be defined over any compact spaces, but physically realizable fields will be extended over at most three spatial dimensions and perhaps one time dimension. Quantum systems are an exception, for separate quantum systems with states in $\Phi(\Omega_1), \ldots, \Phi(\Omega_n)$ can be assembled into a composite system with the state space $\Phi(\Omega_1) \otimes \cdots \otimes \Phi(\Omega_n)$, which is isomorphic to $\Phi(\Omega_1 \times \cdots \times \Omega_n)$. There is no theoretical limit to the dimension of such systems.

For classical systems, however, we may need to reduce the spatial dimension, and these techniques may be applicable to quantum systems as well. To understand the problem, suppose we are using field computation to process images represented as two-dimensional fields in $\Phi(\Omega)$, where Ω is a bounded subset of two-dimensional Euclidean space. In general, a linear operation for processing these images will have a kernel in $\Phi(\Omega \times \Omega)$, a space of four-dimensional fields, which are in general physically unrealizable.

One solution to this problem is, for example, to represent the two-dimensional images by one-dimensional fields on which we can operate with a two-dimensional kernel. To accomplish this re-representation, we use generalized Fourier decompositions of the fields. For example, suppose $|1\rangle, |2\rangle, \ldots$ is an ON basis for $\Phi(\Omega)$, a space of 2D fields, and suppose $|1'\rangle, |2'\rangle, \ldots$ is an ON basis for $\Phi(\Omega')$, a space of 1D fields. A 1D representation $\psi \in \Phi(\Omega')$ of an arbitrary $\phi \in \Phi(\Omega)$ can be computed:

$$|\psi\rangle = \sum_k |k'\rangle\langle k \mid \phi\rangle = \left(\sum_k |k'\rangle\langle k| \right) |\phi\rangle.$$

This invertible transformation is implemented by the 3D kernel $K = \sum_k |k'\rangle\langle k|$.

Since the original and reduced fields have the same generalized Fourier coefficients, the fields have (by the Parseval equality) the same norms, and so the transformation is isometric and preserves inner products. Obviously, $KK^\dagger = I$ and $K^\dagger K = I$ (but with different identity operators). Therefore, K is unitary and can be implemented by quantum gates.

Of course, reducing the dimension of the operand fields won't do much good unless we can also reduce the dimension of the kernel that is to operate on them. To see how to do this, suppose $L : \Phi(\Omega) \rightarrow \Phi(\Omega')$ is a Hilbert-Schmidt linear operator. Use the kernel $\mathrm{H} = \sum_k |k'\rangle\langle k| \in \Phi([0,1] \times \Omega)$ to reduce the 2D input to a 1D field in $\Phi([0,1])$ with ON basis $|1'\rangle, |2'\rangle, \ldots$. Similarly, use the kernel $\Theta = \sum_j |j''\rangle\langle j'| \in \Phi(\Omega' \times [0,1])$, where $|1''\rangle, |2''\rangle, \ldots$ is an ON basis for $\Phi(\Omega')$, to transform the 1D output representation into the 2D output space. The actual computation has a 2D kernel, for it transforms a 1D input representation to a 1D output representation. The 2D kernel $K \in \Phi([0,1]^2)$ to perform the computation on the 1D fields is just the sum of the matrix elements times the corresponding dyads:

$$K = \sum_{j,k} \langle j'' \mid L \mid k\rangle \; |j'\rangle\langle k'|.$$

Putting it all together, the 2D-to-2D operator can be executed by means of 1D fields by a composition of three field computations using at most 3D kernels:

$L = \Theta K \mathrm{H}$. Of course, nothing comes for free. The spatial bandwidth of the 1D fields will be approximately the square of the bandwidth of the 2D fields. That is, the dimension (in terms of basis elements) of the 1D field is the product of the dimensions (in basis elements) of the 2D field's two spatial dimensions. The same approach applies to fields of any dimension, provided the generalized Fourier coefficients of the input can be extracted and provided the output can be constructed from its coefficients.

In some cases it may be necessary to factor a field computation through a finite-dimensional space. This may occur in the brain when a relatively small number of neurons have no significant spatial relationship (i.e., the spatial relationships among them do not convey information). Suppose, for example, that $L : \Phi(\Omega) \rightarrow \Phi(\Omega')$, that $|k\rangle$, $k = 1, \ldots, m$, constitute an ON basis for $\Phi(\Omega)$, and that $|j'\rangle$, $j = 1, \ldots, n$, constitute an ON basis for $\Phi(\Omega')$. (Physically realizable fields are finite-dimensional.) Consider $|\psi\rangle = L|\phi\rangle$. Let $\mathbf{c}$ be the vector of generalized Fourier coefficients of the input, $c_k = \langle k \mid \phi\rangle$, and let $\mathbf{d}$ be the coefficients of the output, $d_j = \langle j' \mid \psi\rangle$. The Hilbert-Schmidt theorem shows that $\mathbf{d} = \mathrm{M}\mathbf{c}$, where $M_{jk} = \langle j' \mid L \mid k\rangle$. To put this in neural network terms, we have an input layer of m neurons with receptive fields $|k\rangle$, $k = 1, \ldots, m$. They are connected through weights M_{jk} to an second layer of n neurons with projection fields $|j'\rangle$, $j = 1, \ldots, n$, which are summed to produce the output, $\psi = \sum_{j=1}^{n} d_j |j'\rangle$. The numbers m and n could be quite large, but we don't treat these neural layers as fields because they do not have significant spatial structure.

A partial reduction of dimension is sometimes useful. In these cases some of the spatial dimensions are reduced through a discrete set of basis functions, as described above, but the others remain unreduced. Let $L : \Phi(\Omega) \rightarrow \Phi(\Omega')$ be a linear operator with kernel $K \in \Phi(\Omega' \times \Omega)$. We suppose that $\Omega = \Omega_1 \times \Omega_2$ has physically unrealizable dimension. Letting $\psi = K\phi$ and treating $\phi_v = \phi_{xy}$ as a function of y, $\phi_x(y)$, we expand the product:

$$\psi_u = \int_\Omega K_{uv} \phi_v \mathrm{d}v = \int_{\Omega_1} \int_{\Omega_2} K_{uxy} \phi_x(y) \mathrm{d}y \mathrm{d}x, \tag{8}$$

Next expand ϕ_x in a generalized Fourier series in terms of $|1\rangle, |2\rangle, \ldots$, an ON basis of $\Phi(\Omega_2)$: $\phi_x = \sum_k |k\rangle\langle k \mid \phi_x\rangle$. With some abuse of notation, the Fourier coefficients are given by:

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

where the field product $\phi\langle k| \in \Phi(\Omega_1)$. Substitute the formula for the Fourier coefficients into the series, and the result into Eq. 8 to get:

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

That is, $\psi = \sum_k K|k\rangle(\phi\langle k|)$. (The parentheses are required because the field product is not associative.) Therefore, let $J_k = K|k\rangle$ to obtain a field computation $L(\phi) = \sum_k J_k(\phi\langle k|)$, which makes use of kernels $J_k \in \Phi(\Omega'\times\Omega_1)$ of lower dimension than $K \in \Phi(\Omega'\times\Omega)$.

In this reduction we discretized through $\Phi(\Omega_2)$, but $\Phi(\Omega_1)$ can be discretized in the same way, and moreover several dimensions can be discretized. Usually, we discretize those dimensions that have the fewest non-negligible Fourier coefficients (smallest bandwidth).

The preceding example reduced the kernel dimension by discretizing one dimension of the input space, but sometimes it is preferable to discretize an output dimension. For example, suppose $L : \Phi(\Omega) \to \Phi(\Omega')$ with kernel $K \in \Phi(\Omega'\times\Omega)$, where $\Omega' = \Omega_1\times\Omega_2$. Let $|1\rangle, |2\rangle, \ldots$ be an ON basis for $\Phi(\Omega_1)$. Consider an output value $\psi_u = \psi_{xy}$ as a function of x, expand, and rearrange as before:

$$\begin{aligned}
\psi_{xy} &= \sum_k |k\rangle_x \int_{\Omega_1} \langle k|_{x'}\psi_{x'y}\mathrm{d}x' \\
&= \sum_k |k\rangle_x \int_{\Omega}\int_{\Omega_1} \langle k|_{x'}K_{x'yv}\mathrm{d}x'|\phi\rangle_v \mathrm{d}v \\
&= \sum_k |k\rangle_x(\langle k \mid K \mid \phi\rangle)_y \\
&= \sum_k (|k\rangle \otimes \langle k \mid K \mid \phi\rangle)_{xy}.
\end{aligned}$$

Therefore, $\psi = \sum_k |k\rangle \otimes \langle k \mid K \mid \phi\rangle$. Define the kernel $J_k = \langle k|K \in \Phi(\Omega_2\times\Omega)$, and we can compute the operator with lower dimensional fields: $L(\phi) = \sum_k |k\rangle\otimes J_k|\phi\rangle$.

In the above examples, we have been concerned with reducing the dimension of fields so that they are physically realizable, and we have seen that one way to do this is to represent a continuous field by a discrete set of generalized Fourier coefficients. However, it is sometimes useful to go in the opposite direction, using a constant-time field product to implement a finite-dimensional matrix-vector product.

Let M be an $m \times n$ matrix, let $\mathbf{c} \in \mathbb{R}^n$ be the input vector, and suppose that our intention is to compute $\mathbf{d} = \mathrm{M}\mathbf{c}$ by a field product $|\psi\rangle = K|\phi\rangle$. That is, we will represent the input vector by a field $|\phi\rangle \in \Phi(\Omega)$ in a field space $\Phi(\Omega)$ containing n physically realizable ON fields $|1\rangle, \ldots, |n\rangle$. The input field is defined $|\phi\rangle = \sum_{k=1}^{n} c_k |k\rangle$, and the output field $\psi \in \Phi(\Omega')$ is defined $|\psi\rangle = \sum_{j=1}^{m} d_j |j'\rangle$, for physically realizable ON fields $|1'\rangle, \ldots, |m'\rangle$ of $\Phi(\Omega')$. The matrix-vector multiplication is implemented by a kernel $K \in \Phi(\Omega' \times \Omega)$ that is given by $K = \sum_{j=1}^{m} \sum_{k=1}^{n} M_{jk} |j'\rangle\langle k|$. To see this, observe:

$$\begin{aligned}
K|\phi\rangle &= \left(\sum_{j,k} M_{jk} |j'\rangle\langle k| \right) |\phi\rangle \\
&= \sum_{j,k} M_{jk} |j'\rangle \langle k \mid \phi\rangle \\
&= \sum_{j} |j'\rangle \sum_{k} M_{jk} c_k \\
&= \sum_{j} |j'\rangle d_j \\
&= |\psi\rangle .
\end{aligned}$$

7. Cortical Field Computation

Field computation has proved useful in describing neural systems underlying motor control and in formulating *dynamic field models* of many cognitive processes [20, 21]. Moreover, field computation on cortical maps shows how neural systems can implement nonlinear operations on multiple inputs in superposition, analogous to quantum computation [22]. The complex coefficients can encode additional pragmatic information, such as probability, confidence, or importance.

7.1. Information Fields

It is often useful to distinguish the meaning of a message, which is defined by syntax and semantics, from its pragmatics, which is conveyed by other aspects of this signal. For example, the loudness of an utterance does not affect its meaning, but it could affect the likelihood of its reception or its effect on a hearer's behavior. Similarly, repeating a signal does not affect its information content, but it might affect the reliability of its transmission. Often, we can say that it is the "shape" of a signal that conveys information, and it is its "size" and other bulk properties (e.g., color, pitch, brightness) that convey pragmatics. For example, Hopfield [23] remarked that in some neural systems the phase of impulses conveyed the information, but the frequency of the impulses conveyed pragmatic characteristics, such as urgency, importance, or certainty. Therefore, we may define an *information field* to be a normalized field, or alternatively a field whose magnitude doesn't matter (i.e., a *ray* in a projective Hilbert space), just like a quantum state vector. We can think of an ordinary field ϕ as comprising two components, $\phi = p\psi$, a pragmatic magnitude $p = \|\phi\|$, and an information-bearing form, $\psi = \phi/p$. Similarly, in quantum mechanical systems Bohm and Hiley [24, pp. 35–6] distinguish a *form*, which *guides* the action, and a *magnitude*, which determines the *amount* of action.

7.2. Non-linear Computation by Topographic Maps

Much of the information in the brain is organized in *topographic maps* in which sensory properties or other information is systematically mapped to spatial location. For example, there are *tonotopic maps* that systematically map pitch and *retinotopic maps* that reflect the organization of the retina. Information processing in topographic maps is naturally described by field computation, and topographic mapping facilitates computing nonlinear transformations of the information.

In these maps, activity in a region represents the information by its location within the region. That is, a particular value $x \in \Omega$ is represented by a corresponding field in $\Phi(\Omega)$ that is distinctly peaked at x. Mathematically, this field can be approximated by a Dirac delta function $\delta(u-x)$ located at x; as in quantum mechanics, we will write it $|x\rangle$. It corresponds to the wave function of an idealized particle with its probability density concentrated at x.

Now consider an arbitrary function $f : \Omega \to \Omega'$. Its value for a particular x, $f(x)$, is represented by a delta function with its peak at the location corresponding to this value, that is, $|f(x)\rangle$. In neural terms, each neuron representing an $x \in \Omega$ is connected to the neuron $y \in \Omega'$ representing $y = f(x)$. Investigating this in field computation terms will reveal some of the advantages of computing with topographic maps [22].

Given an arbitrary $f : \Omega \to \Omega'$ we will use a corresponding kernel $K \in \Phi(\Omega' \times \Omega)$ so that for all $x \in \Omega$, $|f(x)\rangle = K|x\rangle$. Since the delta functions are ON, this is simply:

$$K = \int_\Omega |f(x)\rangle\langle x| \mathrm{d}x, \tag{9}$$

which is, in essence, the *graph* of the function f; the computation is a sort of table lookup.

The delta functions can be approximated by the point-to-point neural connections described above, but they are not physically realizable in general. Therefore, it will be worthwhile to re-express this computation in terms of realizable kernels. To accomplish this, expand the deltas in terms of an ON basis; for example, $|x\rangle = \sum_k |k\rangle\langle k \mid x\rangle$ and therefore $\langle x| = \sum_k \langle x \mid k\rangle\langle k|$. Similarly, $|f(x)\rangle = \sum_j |j'\rangle\langle j' \mid f(x)\rangle$. Substitute the latter two Fourier series into Eq. 9 to get:

$$\begin{aligned}
K &= \int_\Omega \left(\sum_j |j'\rangle\langle j' \mid f(x)\rangle \right) \otimes \left(\sum_k \langle x \mid k\rangle\langle k| \right) \mathrm{d}x \\
&= \sum_{j,k} |j'\rangle \otimes |k\rangle \int_\Omega \langle j' \mid f(x)\rangle\langle x \mid k\rangle \mathrm{d}x \\
&= \sum_{j,k} |j'\rangle \otimes |k\rangle\langle j' \circ f \mid k\rangle,
\end{aligned}$$

where $|j' \circ f\rangle$ is the composition of $|j'\rangle$ and f: $|j' \circ f\rangle_x = |j'\rangle_{f(x)}$.

Computing with topographic maps has several interesting properties, one of which is that it uses a linear operator to compute a possibly nonlinear function. Several useful properties follow from linearity. For example, the map is able to operate on a superposition of two or more simultaneous inputs to produce a superposition of their outputs: $K(|x\rangle + |x'\rangle) = |f(x)\rangle + |f(x')\rangle$. Since inputs and outputs are represented by the locations of the peaks rather than their heights, the magnitude of the impulse (or its quantum probability amplitude) can be used to represent a pragmatic property of the input value, such as its certainty or importance (see Sec. 7.1). Topographic computation passes these pragmatic characteristics from the inputs to the outputs: $K(p|x\rangle) = p|f(x)\rangle$. If we have two (or more) inputs $x, x' \in \Omega$ with corresponding pragmatic weights $p, p' \in \mathbb{R}$, perhaps reflecting confidence or importance, then the outputs carry the same weights: $K(p|x\rangle + p'|x'\rangle) = p|f(x)\rangle + p'|f(x')\rangle$. If multiple simultaneous inputs, each with its own weight, happen to lead to the same output, then their weights will combine; for example, if $f(x) = f(x')$, then $K(p|x\rangle + p'|x'\rangle) = (p+p')|f(x)\rangle$. In this way individually uncertain or unimportant inputs may jointly lead to an output of much greater certainty or importance.

These observations extend naturally to the continuum case. Suppose the real field $\varpi \in \Phi(\Omega)$ represents the probability or other pragmatic characteristics of the information, so that ϖ_x is the pragmatic weight of $x \in \Omega$. If we think of the input field as a weighted superposition of delta functions, $|\varpi\rangle = \int_\Omega \varpi_x |x\rangle \mathrm{d}x$, then we can see that the output will reflect these weights: $K|\varpi\rangle = \int_\Omega \varpi_x |f(x)\rangle \mathrm{d}x \in \Phi(\Omega')$. The pragmatic weight of each output point will be the sum of the weights of all inputs leading to that output:

$$\langle y \mid K \mid \varpi\rangle = (K\varpi)_y = \int_{\{x|y=f(x)\}} \varpi_x \mathrm{d}x = \int_{f^{-1}(y)} \varpi_x \mathrm{d}x.$$

If f is not bijective, then K will not be unitary, and therefore not quantum computable. In this case we can us instead $K = \int_\Omega (|f(x)\rangle \otimes |x\rangle)(\langle 0| \otimes \langle x|)\mathrm{d}x$, where $|0\rangle \in \Phi(\Omega')$.

7.3. Field Representations of Discrete Symbols

Quantum inspired Hilbert-space models are illuminating aspects of human categorical cognition and may lead to improved use of concepts in artificial intelligence systems. For example, Pothos and Busemeyer [25] have presented compelling evidence that quantum probability provides a better model than classical probability for many cognitive processes. We have argued that this is because these cognitive processes are implemented in cortex by field computation, since it is described by the same mathematics as quantum probability [26]. Aerts, Gabora, and Sozzo [27] have used the mathematics of quantum mechanics to explain how humans combine concepts and use them in context.

Most concepts have indefinite boundaries and are context sensitive; they are modeled well by connectionist theories, but the *words* that denote these concepts are discrete, analogous to particles. For the most part words have a discrete topology; that is, as lexical entities (not strings of letters or phonemes) they are either the same or completely different: $d(x, x) = 0$ and $d(x, y) = 1$ for $x \neq y$. Therefore, words (and other particular things that must be kept distinct) are naturally represented by orthonormal fields, which have exactly this property. Such orthogonal representations emerge through self-organizing processes from the essentially continuous underlying adaptive and learning mechanisms in the brain.

7.4. Gabor Representation and the Uncertainty Principle

The well-known Heisenberg uncertainty principle applies to non-quantum systems, as shown by Gabor [28], who applied the Heisenberg-Weyl derivation to arbitrary square-integrable signals of finite duration and bandwidth; it is thus applicable to field computation. It quantifies the degrees of freedom of physically realizable fields.

Suppose that ψ is a field over n-dimensional space, $\psi \in \Phi(\Omega)$. The spread or uncertainty of the field along each spatial dimension x_k can be measured 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 $\mathbf{x} = (x_1, \ldots, x_n)^{\mathrm{T}} \in \Omega \subset \mathbb{R}^n$. Let $\Psi(\mathbf{u})$ be the Fourier transform of $\psi(\mathbf{x})$, and define in a similar way the uncertainty or spread in the frequency domain:

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

Gabor proved that $\Delta x_k \Delta u_k \geq 1/4\pi$, which may be called the *Gabor Uncertainty Principle* [for an informal introduction, see 29].

The minimum joint spread in the space and spatial frequency domains, $\Delta x_k \Delta u_k = 1/4\pi$, is achieved by the *Gabor elementary fields*, which are Gaussian-modulated complex exponentials (i.e., *wave packets*):

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

This is a family of fields parameterized by $\mathbf{p}$, which is the center of the Gaussian envelope, and $\mathbf{u}$, which is a *wave vector* defining the frequency and phase of the complex exponential. All members of the family share a common Gaussian envelope whose shape is defined by a diagonal *aspect matrix* $\mathrm{A} = \mathrm{diag}(\alpha_1, \ldots, \alpha_n)$. It defines the common spread of the wave packets in the space and frequency domains: $\Delta x_k = \alpha_k/(2\sqrt{\pi})$ and $\Delta u_k = \alpha_k^{-1}/(2\sqrt{\pi})$.

We can characterize a field (function) by its amplitude at various points in *Gabor-space*, that is, at different combinations of spatial location and frequency $(\mathbf{x}, \mathbf{u})$. But the Gabor Uncertainty Principle shows that these amplitudes cannot be localized more closely than cells of size $\prod_{k=1}^n \alpha_k/(2\sqrt{\pi})\ \alpha_k^{-1}/(2\sqrt{\pi}) = (4\pi)^{-n}$. Gabor referred to these elementary units of (complex-valued) information as *logons*, and the number of them in a field of finite extent and bandwidth as its *logon content.* It measures the maximum information (degrees of freedom) that can be represented by such a function (and Gabor's original purpose was to analyze the information-carrying capacity of the trans-Atlantic cable). If X_k is a field's extent along the kth axis, and U_k its bandwidth on that axis, then the field's Gabor-space volume is $V_{\mathrm{G}} = \prod_{k=1}^{n} X_k U_k$ and its logon content is $L = (4\pi)^n V_{\mathrm{G}}$.

Gabor showed that any square-integrable function with a finite Gabor-space volume could be expanded into a superposition of Gabor elementary functions [30, pp. 656–7]: $\psi = \sum_{k=1}^{L} c_k g_k$, where for convenience I have indexed the Gabor functions sequentially. That is, any physically realizable field can analyzed as a superposition of wave packets, and the L complex coefficients c_k constitute the information content of the field. Since the Gabor elementary functions are not orthogonal, these coefficients cannot be computed directly by an inner product, $\langle g_k \mid \psi \rangle$. With appropriate choice of parameters, however, the Gabor elementary functions constitute a *tight frame*, which allows the coefficients to be approximated by the inner product [31, p. 1275]. Alternatively, they can be computed directly by least-squares approximation (see Sec. 8) or by gradient descent on the error of approximation, $\mathcal{E} = \|\hat{\psi}(\mathbf{c}) - \psi\|^2$, where $\hat{\psi}(\mathbf{c}) = \sum_{k=1}^{L} c_k g_k$. Since $\partial\mathcal{E}/\partial c_k = 2\langle g_k \mid \hat{\psi}(\mathbf{c}) - \psi \rangle$, gradient descent is $\dot{c}_k \propto \langle g_k \mid \psi - \hat{\psi}(\mathbf{c}) \rangle$.

8. Universal Field Computation

An important question, with both theoretical and practical implications, is whether there can be a *universal field computer*, that is, a field computer that can be programmed to do any field computation in a large, significant class of computations. This question can be answered from the perspective of the ordinary theory of computation, but it does not address the key questions in an informative way. Since, in mathematical terms, fields are continuous functions, it is more natural to address universal field computation from the perspective of approximation theory in Hilbert spaces.

For example, as explained in Sec. 4 (Thm. 1), there is an analogue of Taylor's theorem for Hilbert spaces, and this theorem allows us to expand field transformations in sums of multilinear operators, which are analogous to the polynomials of the familiar Taylor theorem [3, 4, 22, 32]. This shows how to compute a wide class of field transformations with a small set of operations: field product, field addition, and field scaling. The required gradient kernels may be of high spatial dimension, but we have discussed ways of reducing dimension (Sec. 6).

Another approach to the question comes from various universal approximation theorems for real- and complex-valued functions [e.g., 33, pp. 166–168, 219–220, 236–239, 323–326]. They can be extended to physically realizable fields, which have a finite number of non-zero Fourier coefficients. These theorems show how to approximate field transformations with a limited repertoire of basic field operations.

For example, suppose that we want to interpolate a real-valued field transformation $F : \Phi(\Omega) \to \Phi(\Omega')$ given by the input-output samples (ϕ^k, ψ^k), where $F(\phi^k) = \psi^k$, $k = 1, \ldots, P$ with an interpolating function of the form

$$\hat{\psi} = \sum_{j=1}^{H} r_j(\phi)\alpha_j, \tag{10}$$

for some H. The operators $r_j : \Phi(\Omega) \to \mathbb{R}$ are fixed nonlinear functionals (e.g., radial basis functions) that weight the output projection fields $\alpha_j \in \Phi(\Omega')$. These fields can be computed by using the least-squares algorithm to minimize the approximation error, $\mathcal{E} = \sum_{k=1}^{P} \|\hat{\psi}^k - \psi^k\|^2$, where $\hat{\psi}^k = \sum_{j=1}^{H} r_j(\phi^k)\alpha_j$, as explained next.

By Parseval's identity, $\|\hat{\psi}^k - \psi^k\|^2$ is equal to the sum of the squares of the generalized Fourier coefficients of $\hat{\psi}^k - \psi^k$. Therefore, let $|1\rangle, |2\rangle, \ldots$ be a basis for $\Phi(\Omega')$, and compute them:

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

Since physically realizable fields have finite bandwidth, and for any fields we can pick a number N of Fourier coefficients to approximate them as close as we like, we can reduce this to a finite-dimensional least-squares problem. Define a $P \times H$ matrix R of the basis functional values for each input sample:

$R_{kj} = r_j(\phi^k)$. Let A be an $H \times N$ matrix (to be determined) of the Fourier coefficients of the projection fields, $A_{ji} = \langle i \mid \alpha_j \rangle$, and let Y be a $P \times N$ matrix of the Fourier coefficients of the output samples, $Y_{ki} = \langle i \mid \psi^k \rangle$. In these terms,

$$\langle i \mid \hat{\psi}^k - \psi^k \rangle = \sum_{j=1}^{H} R_{kj} A_{ji} - Y_{ki}.$$

Therefore let the error matrix $\mathrm{E} = \mathrm{RA} - \mathrm{Y}$ so that $\|\hat{\psi}^k - \psi^k\|^2 = \sum_{i=1}^{N} E_{ki}^2$, and the total error is the squared Frobenius norm: $\hat{\mathcal{E}} = \sum_{k,i} E_{ki}^2 = \|\mathrm{E}\|_F^2$.

This is an ordinary least-squares problem, and the error is minimized by $\mathrm{A} = \mathrm{R}^+\mathrm{Y}$, where R^+ is the *Moore-Penrose pseudoinverse*, $\mathrm{R}^+ = (\mathrm{R}^\mathrm{T}\mathrm{R})^{-1}\mathrm{R}^\mathrm{T}$ [e.g., 34, pp. 371–3]. The solution matrix A contains the Fourier coefficients of the output projection fields that minimize the error: $\alpha_j = \sum_{i=1}^{N} A_{ji} |i\rangle$.

Thus, we can determine the projection fields that minimize the error, but for universal approximation we need to be able to make the error as small as we like. This depends on the number and shape of the basis functionals r_j. One suitable class of functions are the radial basis functions, $r_j(\phi) = r(\|\phi - \eta_j\|)$, where each focal field η_j causes the maximal response of the corresponding basis function r_j. In fact, we can choose $H = P$ and $\eta_j = \phi^j$, and the matrix R will be invertible for many radial functions r [33, pp. 238–9]. Therefore, these radial basis functionals (for which convolution might be used), together with field scaling and summation (in order to evaluate Eq. 10) are sufficient to approximate any realizable field transformation.

These radial basis function approximations are like continuum neural networks, and other neural network-like field computations are also universal, such as basis functionals of the form $r_j(\phi) = s(\langle \omega_j \mid \phi \rangle + b_j)$. The weight fields ω_j and the bias values b_j are determined by an approximation algorithm [33, pp. 166–168]. In this case the operations required for universality are field scaling and summation for Eq. 10, and inner product, scalar addition, and the sigmoid s for computing the basis functionals.

9. General Purpose Field Computers

It remains to say a little about general purpose field computers. These can take many forms. The most familiar arrangement is to supplement an ordinary digital computer with hardware for performing a repertoire of field operations. Fields are stored in appropriate registers (1D, 2D, or 3D) like numbers in ordinary computers. Computation proceeds in sequential steps with loops, conditionals, and the other familiar apparatus of computer programming.

The most natural implementation of field computation (historically common, but less familiar nowadays) is the continuous-time analog computer. Programs are, in effect, partial differential equations, and the field computer integrates them with respect to time. In addition to initial-value and boundary-value problems, such computers can process continuous input signals and generate continuous output signals for control applications. This is closest to the function of the nervous system, which depends heavily on field computation.

In Section 8 we have seen that a few operations are sufficient for universal computation: field summation ($\phi + \psi$), field scaling ($z\phi$), inner product ($\langle \phi \mid \psi \rangle$), field product ($K\phi$), and simple radial basis functionals. More specifically, perceptron-net-style approximation requires the inner product and any non-constant, bounded, monotone-increasing scalar function (i.e., a sigmoid function), and radial basis approximation requires the norm (which can be computed with the inner product) and any non-constant, bounded, monotone-decreasing scalar function. Reduction of dimension may require the tensor/outer product ($\phi \otimes \psi$). Other useful operations, which are easy to implement in some technologies, include convolution ($\phi * \psi$), cross-correlation ($\phi \star \psi$), various local operators (e.g., $\overline{\log}$, $\overline{\exp}$, etc.), and vector field operations (∇, $\nabla\cdot$, ∇^2). Other vector field operations can be implemented in terms of their cartesian or polar components. Scalar analog computation is implemented by field computation on zero-dimensional fields in Φ^0.

10. Conclusions and Future Work

Hilbert spaces provide the mathematical foundations of quantum mechanics, and quantum inspired computation can be identified with computation in Hilbert spaces. In particular, the concept of a *field*, a continuous spatial distribution of quantity, can be used as a model both of information representations in neural cortex and of quantum wave functions. Field computation thus provides a unifying framework for quantum inspired cognition, both natural and artificial. A relatively small number of field operations, which have natural physical implementations, are sufficient for universal field computation and correspond to simple neural networks.

Research in field computation is progressing in many directions simultaneously. For example, continuing progress in neuroscience is illuminating cortical information representation and processing. These processes, many of which depend on cortical maps with high neuron densities, can be expressed naturally as field transformations and therefore implemented on current and future field computers. As we strive to reach and to surpass human-scale artificial intelligence, the most direct route will be to implement these processes by means of field-computing hardware. Quantum computation is also a kind of field computation, and therefore field computation provides a unifying framework for neural computation and quantum computation. In particular, field computation provides a path by which techniques from neural computation (over normalized vectors) can be brought into quantum computation, implementing computation over dense neural arrays by physical operations on quantum wave functions. Finally, field computation is a theory of massively parallel (or, more correctly, continuously parallel) analog computation, which can be implemented in a variety of current and future technologies. Implementation on conventional digital computers and graphical processing units has been and will continue to be a convenient medium for field computation. Other computational media, however, promise more direct implementations. These include nonlinear optical computation, continuous variable quantum computation, chemical computation, and massively parallel analog computation. Quantum inspired field computation is thus a theoretical foundation for both natural and artificial intelligence.

References

[1] S. Lloyd, S. L. Braunstein, Quantum computation over continuous variables, Phys. Rev. Lett. 82 (1999) 1784–1787. `doi:10.1103/PhysRevLett.82.1784`.
URL `http://link.aps.org/doi/10.1103/PhysRevLett.82.1784`

[2] C. E. Collins, D. C. Airey, N. A. Young, D. B. Leitch, J. H. Kaas, Neuron densities vary across and within cortical areas in primates, Proceedings of the National Academy of Sciences 107 (36) (2010) 15927–15932. `doi:10.1073/pnas.1010356107`.

[3] B. J. MacLennan, Technology-independent design of neurocomputers: The universal field computer, in: M. Caudill, C. Butler (Eds.), Proceedings of the IEEE First International Conference on Neural Networks, Vol. 3, IEEE Press, 1987, pp. 39–49.

[4] B. J. MacLennan, Field computation in natural and artificial intelligence, in: R. Meyers et al. (Ed.), Encyclopedia of Complexity and System Science, Springer, 2009, Ch. 6, entry 199, pp. 3334–3360. `doi:10.1007/978-0-387-30440-3\_199`.

[5] B. J. MacLennan, The promise of analog computation, International Journal of General Systems 43 (7) (2014) 682–696. `doi:10.1080/03081079.2014.920997`.

[6] B. J. MacLennan, The nature of computing — computing in nature, Tech. Rep. UT-CS-05-565, Department of Computer Science, University of Tennessee, Knoxville, also available from `web.eecs.utk.edu/~mclennan` (November 25 2005).

[7] B. J. MacLennan, Field computation: A theoretical framework for massively parallel analog computation, parts I–IV, Tech. Rep. CS-90-100, Department of Computer Science, University of Tennessee, Knoxville, also available from `web.eecs.utk.edu/~mclennan` (1990).

[8] M. A. Nielsen, I. L. Chuang, Quantum Computation and Quantum Information, 10th Edition, Cambridge, 2010.

[9] G. Brachman, L. Narici, Functional Analysis, Academic Press, New York, 1966.

[10] Mathematical Society of Japan, Encyclopedic Dictionary of Mathematics, MIT Press, Cambridge, 1980, s. Iyanaga and Y. Kawada (eds.).

[11] W. Light, Ridge functions, sigmoidal functions and neural networks, in: E. Cheney, C. Chui, L. Schumaker (Eds.), Approximation Theory VII, Academic Press, Boston, 1992, pp. 163–206.

[12] M. Powell, Radial basis functions for multivariable interpolation: A review, in: IMA Conference on Algorithms for the Approximation of Functions and Data, RMCS, Shrivenham, UK, 1985, pp. 143–67.

[13] T. Sanger, Probability density estimation for the interpretation of neural population codes, Journal of Neurophysiology 76 (1996) 2790–3.

[14] D. E. Rumelhart, J. L. McClelland, the PDP Research Group, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, MIT Press, Cambridge, MA, 1986.

[15] M. Miller, B. Roysam, K. Smith, J. O'Sullivan, Representing and computing regular languages on massively parallel networks, IEEE Transactions on Neural Networks 2 (1991) 56–72.

[16] P. Ting, R. Iltis, Diffusion network architectures for implementation of Gibbs samplers with applications to assignment problems, IEEE Transactions on Neural Networks 5 (1994) 622–38.

[17] O. Steinbeck, A. Tóth, K. Showalter, Navigating complex labyrinths: Optimal paths from chemical waves, Science 267 (1995) 868–71.

[18] A. Adamatzky, Computing in Nonlinear Media and Automata Collectives, Institute of Physics Publishing, Bristol, 2001.

[19] A. Adamatzky, B. De Lacy Costello, T. Asai, Reaction-Diffusion Computers, Elsevier, Amsterdam, 2005.

[20] B. J. MacLennan, Field computation in motor control, in: P. G. Morasso, V. Sanguineti (Eds.), Self-Organization, Computational Maps and Motor Control, Elsevier, 1997, pp. 37–73, also available from `web.eecs.utk.edu/~mclennan`.

[21] J. P. Spencer, G. Schöner (Eds.), Dynamic Thinking: A Primer on Dynamic Field Theory, Oxford University Press, New York, NY, 2015, in press.

[22] B. J. MacLennan, Field computation in natural and artificial intelligence, Information Sciences 119 (1999) 73–89, also available from `web.eecs.utk.edu/~mclennan`.

[23] J. Hopfield, Pattern recognition computation using action potential timing for stimulus response, Nature 376 (1995) 33–6.

[24] D. Bohm, B. Hiley, The Undivided Universe: An Ontological Interpretation of Quantum Theory, Routledge, 1993.

[25] E. M. Pothos, J. R. Busemeyer, Can quantum probability provide a new direction for cognitive modeling?, Behavioral and Brain Sciences 36 (2013) 255–327. `doi:10.1017/S0140525X12001525`.

[26] B. J. MacLennan, Cognition in Hilbert space, Behavioral and Brain Sciences 36 (3) (2013) 296–7. `doi:10.1017/S0140525X1200283X`.

[27] D. Aerts, L. Gabora, S. Sozzo, Concepts and their dynamics: A quantum–theoretic modeling of human thought, Topics in Cognitive Science 5 (2013) 737–772.

[28] D. Gabor, Theory of communication, Journal of the Institution of Electrical Engineers 93, Part III (1946) 429–57.

[29] B. J. MacLennan, Gabor representations of spatiotemporal visual images, Tech. Rep. CS-91-144, Department of Computer Science, University of Tennessee, Knoxville, also available from `web.eecs.utk.edu/~mclennan` (1991).

[30] C. Heil, D. Walnut, Continuous and discrete wavelet transforms, SIAM Review 31 (4) (1989) 628–66.

[31] I. Daubechies, A. Grossman, Y. Meyer, Painless non-orthogonal expansions, Journal of Mathematical Physics 27 (1986) 1271–83.

[32] B. J. MacLennan, Field computation in the brain, in: K. Pribram (Ed.), Rethinking Neural Networks: Quantum Fields and Biological Data, Lawrence Erlbaum, Hillsdale, NJ, 1993, pp. 199–232, also available from `web.eecs.utk.edu/~mclennan`.

[33] S. Haykin, Neural Networks and Learning Machines, 3rd Edition, Pearson Education, New York, 2008.

[34] S. Leon, Linear Algebra with Applications, 2nd Edition, Macmillan, New York, 1986.