Biol. Cybernetics 37, 167–186 (1980)

Biological Cybernetics

On the Representation of Multi-Input Systems: Computational Properties of Polynomial Algorithms

T. Poggio and W. Reichardt
Max-Planck-Institut für biologische Kybernetik, Tübingen, FRG

Abstract. This paper introduces a theoretical framework for characterizing and classifying simple parallel algorithms and systems with many inputs, for example an array of photoreceptors. The polynomial representation (Taylor series development) of a large class of operators is introduced and its range of validity discussed. The problems involved in the polynomial approximation of systems are also briefly reviewed. Symmetry properties of the input-output map and their implications for the system structure (i.e. its kernels) are studied. Finally, the computational properties of polynomial mappings are characterized.

1. Introduction

Visual information processing begins with a large array of "photoreceptors" which transduce local light intensities into time-dependent signals. An algorithm (system) operating on an image thus has many inputs; and it must be nonlinear to execute nontrivial information processing. As will become clear in this paper, every nontrivial computation must be essentially nonlinear, i.e. not even approximately representable by linear operations. The property "nonlinearity" is of course too general; the class of systems and algorithms which are considered must be suitably restricted. In this paper we restrict ourselves to that class of nonlinearities which can be described by polynomial operators.

Via a many input polynomial representation we introduce and develop in some detail a theoretical framework for characterizing and classifying "simple", parallel algorithms and systems. Polynomial algorithms are perhaps the simplest class of parallel algorithms. In their framework we attempt to capture the distinction between "local" and "global" computational problems. Such a theoretical language has been already applied to the analysis of several algorithms used by the visual system of the fly (Poggio and Reichardt, 1976; Reichardt and Poggio, 1979; Bulthoff et al., 1980).

The paper is organized as follows: Section 1 introduces the polynomial representation of an algorithm; it deals with its range of validity and briefly reviews the theory of polynomial approximations of a system. Section 2 discusses symmetry properties and their implications for the system's conceptual structure. Section 3 develops a theory of the computational properties of polynomial systems and compare them with more standard computational machines.

1.1. Polynomial Representation of Algorithms and Networks

An algorithms can be considered as an operation on some kind of input to yield a corresponding output. In formal terms, an algorithm may be thought of as a mapping between an "input" and an "output" space. Perhaps the simplest of all nonlinear operators on a linear space are the so called polynomial operators. They encompass a broad spectrum of mappings, including all linear ones and they approximate all sufficiently smooth nonlinear operators. In this paper, we consider only such "simple" algorithms as can be represented exactly or approximately (see later) in a suitable topology, by polynomial operators.

We consider a polynomial operator to represent an algorithm or its implementation (in terms of a system or network). Thus, we will speak interchangeably of a system (network) and its algorithm. This point of view is somewhat new and may seem strange. Algorithms are commonly associated with procedures of a rather discrete or "stepwise" nature, typically in the form of a computer program. Systems and networks have, on the other hand, a more analog-like character; they are usually thought as transformating, in real time, some more or less continuous input function into an output function. The traditional system point of view, which

0340-1200/80/0037/0167/$04.00

by polynomial mappings. Our previous discussion shows that "smooth" systems are contained in this subclass but does not say how much larger the class is.

If the input space is restricted to a compact subset X of a Banach space E, all non-continuous functionals of X can be approximated by polynomial operators as well as by polynomial operators with separable kernels. The input space is in this case very small, in fact "nearly finite dimensional" (Palm, 1978). In this context it is useful to mention that when the set of input functions is finite dimensional (as it is practically the case for discrete-time systems), all topologies on it are equivalent (see Palm, 1979). Consequently polynomial systems (and also separable kernel systems) can approximate every continuous system (in particular, they represent exactly every system, if the input set is finite). Incidentally, given a function f mapping a Banach space into a Banach space, there is always a Lagrange (and an Hermite) polynomial of degree $n-1$ that interpolates f at n points $x_1 \dots x_n$ (Prenter, 1971). Thus, if the input space is finite, in the sense that only a finite number of input functions can occur, a polynomial, albeit of possibly very high degree, can approximate every system.

Uniform convergence of a Volterra representation of a system takes place in general (apart from the case of entire functionals) only in the neighborhood of a zero function. This means that, as in the function case, the coefficients of the expansion, i.e. the kernels, depend on the "zero" input function around which the system has been expanded. It is often important to determine the radius of convergence of such an expansion. The methods available for this purpose are basically the same used for series expansions. In some cases the radius of convergence and stability criteria are directly provided by local inverse theorems (Halme et al., 1971; Halme, 1972; De Santis and Porter, 1975; Barrett, 1963, 1974). Often the Volterra series can be easily bound in some appropriate norm, in a Banach space (see Poggio and Torre, 1977; Barrett, 1963). The Volterra series does converge for all bounded inputs in the case of bilinear systems (d'Alessandro et al., 1974) and in the case of nonlinear Volterra integral equations describing "synaptic systems" (Poggio and Torre, 1978). These are, perhaps, the first large classes of nonlinear systems for which global convergence of the Volterra series has been established.

1.6. General Formulae

We consider the case of a system processing visual information. The system thus operates on time-dependent signals provided by a set of photoreceptors. One can define the optic array as the 2-dimensional manifold E of visual directions parametrized by the coordinates $\underline{x}=(\psi, \vartheta)$. The intensity field on E, $i(x,t)$ is the input intensity of light (in suitable units) at the point $\underline{x}$, at time t.

The polynomial representation of an operation performed on the input $i(\underline{x},t)$ is then

$$y(t)=k_0+\sum_{J=1}^{M}\int\dots\int\Phi_j(\underline{x}_1,\dots,\underline{x}_j\,;\,t-\tau_1,\dots,t-\tau_j)\cdot\prod_{r=1}^{J}i(x_r,\tau_r)dm(\underline{x}_1)\dots dm(\underline{x}_j)d\tau_1\dots d\tau_j\,, \tag{1.7}$$

where the kernel Φ_j contains information about the interactive structure of the visual system operating on the intensity field and $m(x)$ represents the spatial density of the inputs. If the network has a finite number N of discrete inputs (for instance photoreceptors) located at $\underline{x}_j, j=1,\dots,N$ with the same acceptance function $\varrho(\underline{x})$, one can define the integral operator

$$\int\dots\int\Phi_j(\underline{x}_1,\dots,\underline{x}_j\,;\,t-\tau_1,\dots,t-\tau_j)dm(\underline{x}_1)\dots dm(\underline{x}_J)=\sum_{i_1\dots i_J}\int\dots\int k_{i_1\dots i_J}(t-\tau_1,\dots,t-\tau_j)\cdot\varrho(\underline{x}_1-\underline{x}_{i_1})\dots\varrho(\underline{x}_j-\underline{x}_{i_J})d\underline{x}_1\dots d\underline{x}_j \tag{1.8}$$

and the field representation Eq. (1.4) becomes equivalent to the N input representation

$$y(t)=k_0+\sum_{i=1}^{N}\int f_i(\tau)k_i(t-\tau)d\tau+\sum_{i=1,j=1}^{N}\cdot\iint f_i(\tau_1)f_j(\tau_2)k_{i_j}(t-\tau_1,t-\tau_2)d\tau_1d\tau_2+\dots+\sum_{i_1=1,\dots,i_M=1}^{N}\int\dots\int f_{i_1}(\tau_1)\dots f_{i_M}(\tau_M)\cdot k_{i_1\dots i_M}(\tau-\tau_1,\dots,\tau-\tau_M)d\tau_1\dots d\tau_M \tag{1.9}$$

with

$$f_i(\tau)=\int_E\varrho(\underline{s}-\underline{x}_i^*)i(\underline{s},\tau)d\underline{s}\,. \tag{1.10}$$

The N inputs (photoreceptors) are labelled with $i=1,\dots N$. Unless otherwise stated we will consider in this paper the case of a 1-dimensional slice of the 2-dimensional array of inputs. In cases, however, in which it is important to consider the 2-dimensional geometry, one may label the inputs (photoreceptors) with 2 indices (instead of a collective one) corresponding to their ψ, ϑ coordinates.

Equation (1.10) can be rewritten as

$$y(t)=h_0+\sum_{i=1}^{N}f_i*k_i+\dots+\sum_{i_1=1,\dots,i_M=1}^{N}\cdot f_{i_1}\dots f_{i_M}*^M k_{i_1,\dots,i_M}\,, \tag{1.11}$$

where $*^M$ represents a straightforward generalization of the convolution integral [compare Eqs. (1.10) and (1.8)], i.e.

$$r_1 \dots r_M * k_{i_1,\dots,i_M} = \int \dots \int k_{i_1,\dots,i_M}(t-\tau_1, \dots, \tau_M) \cdot r_1(\tau_1) \dots r_M(\tau_M) d\tau_1 \dots d\tau_M .$$

The input f_i are real valued functions of time, each indexed by the associated input (receptor). The kernels are Volterra kernels, i.e. they are identically zero if one of the arguments is negative. Kernels which have all indexes identical are called selfkernels, otherwise cross-kernels. Thus, selfkernels "look" only at one input (photoreceptor). Each kernel

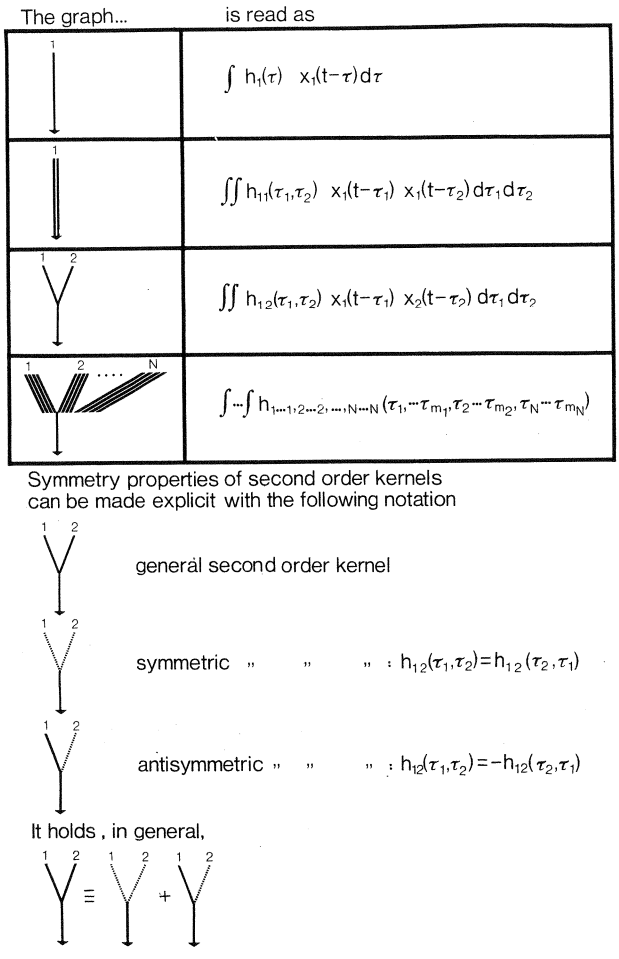
$$k_{\underbrace{i,\dots,i}_{m},\underbrace{j,\dots,j}_{p}}(\tau_1, \dots, \tau_m, \tau_{m+1}, \dots, \tau_{m+p}) \qquad (1.12)$$

is assumed to be symmetric in $\tau_1, \dots, \tau_m$ and in $\tau_{m+1}, \dots, \tau_{m+p}$ separately. The summation in Eq. (1.10) are always carried out through all indices: Thus both k_{i_J} and k_{j_i} appear and may in general differ (see Poggio and Torre, 1978). In some case we will group together such kernels $\bar{k}_{i_J} = k_{i_J} + k_{j_i}$; this will be usually obvious from the context [this was actually done implicitely in previous papers; see Poggio and Reichardt (1976)].

The polynomial mapping, Eq. (1.8), has degree M. Its last term is a M-linear mapping and its kernels is said to be of Mth degree.

The representation, Eq. (1.10), is equivalent to a conceptual decomposition of the network into a sum of interactions of different orders: the kernels k_i are associated with linear mappings, the kernels k_{i_J} with quadratic ones and so on.

Table 1



1.7. Graphical Notation

Table 1 introduces a simple graphical notation which facilitates the interpretation of the various terms of Eq. (1.10) [or the abstract Eq. (1)]. A N-input system can be decomposed into a sequence of graphs (see Fig. 1). Because a specific functional representation may be read from the graphs, they are actually another notation for the power series itself. In this way an algorithm or its network implementation can be decomposed into an additive sequence of simple, canonical terms. Properties of symmetry or structure can be symbolized in the graphical representation. For instance, symmetry properties of the system can be represented in graphs (see Table 1). If some knowledge is available about the system structure, the interactions and the associated graphs may take some more specific form. For instance, the graphs associated with synaptic interactions (see Poggio and Torre, 1978) have a particularly simple structure. Similarly, it is easy to

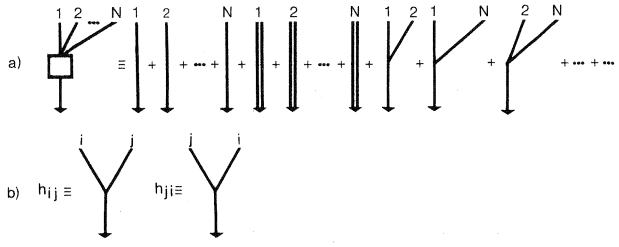


Fig. 1. a Under the conditions discussed in the first sections of this paper a N-input system can be decomposed into a series of standard terms, according to Eq. (1). The various terms can be represented by the graphs defined in Table 1. **a** is actually another way of writing Eq. (1). **b** In the general representation all terms can exist. For instance, h_{i_J} and h_{j_i} are, in general, present and can be indicated as shown in **b**. It is often convenient to lump them together thus defining a new kernel:

$$\bar{h}_{i_J}(\tau_1, \tau_2) = h_{i_J}(\tau_1, \tau_2) + h_{j_i}(\tau_1, \tau_2)$$

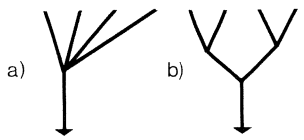


Fig. 2a and b. The general graph **a** may take the specific form **b**, when the fourth order interaction arises from two second order interactions cascaded with another second order graph. Graph **b** implies that the fourth order kernel has the special form

$$h_4(\tau_1,\ldots,\tau_4)=\iint h_{ab}(t_1,t_2)\cdot h_{12}(t_1-\tau_1,t_1-\tau_2)h_{34}(t_2-\tau_3,t_2-\tau_4)\,dt_1\,dt_2$$

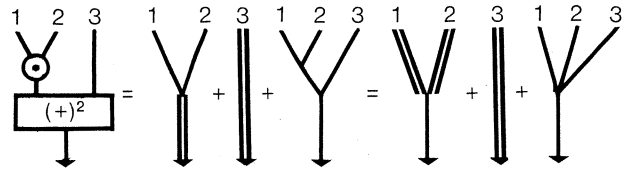


Fig. 3. The Volterra development of the system on the left (linear filters are not indicated). The structure of the kernels corresponding to the "general" graphs on the extreme right has a specific form, as implied by the mixed notation used in the graphs in the middle

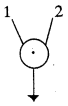
represent the structure of a graph arising from a cascade of lower order nonlinearities (see Fig. 2).

The Volterra development of a system can be computed directly in the graphical notation. Figure 3 gives a very simple example of such a graphical development. A more complex case is represented by feedback systems. Figure 4 shows a case in which the Volterra structure of the open loop systems is known. Of course, such a development may converge only for restricted input regions, if it converges at all.

Some care is needed with the notation proposed in this paper. We use  ... to indicate Volterra terms of a general type, i.e. of the form of Eq. (1.12). A slightly different notation is used for nomemory operations cascaded with linear operations. For instance, a linear system followed by a squaring box is written as

(1.13)

and this is a specific case $[k(\tau_1,\tau_2)=k(\tau_1)k(\tau_2)]$ of  in a similar way



(1.14)

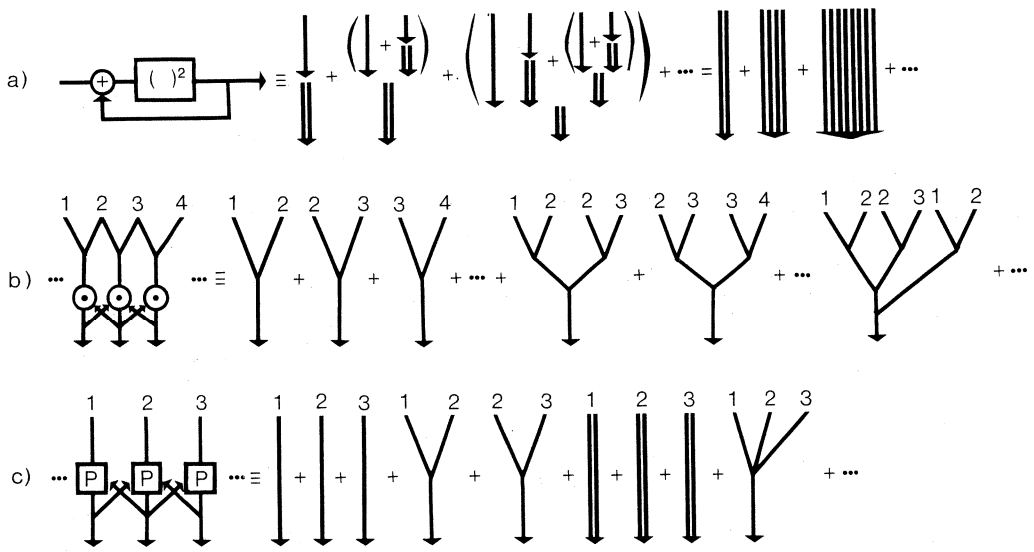


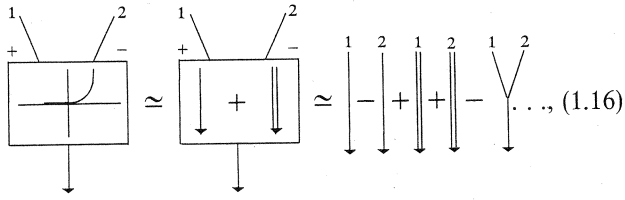
Fig. 4. The Volterra development of a feedback system can be visualized as in **a**. The various graphs can be imagined (if there is a delay in the loop) to arise from a sequence of iterations in the loop. At first the feedback loop is not active and the first graph represents the input-output transduction. One interation later a new term  appears and so on. The development in the middle corresponds to the following "feedback series" $x+(x+x^2)^2+[x+(x+x^2)^2]^2+\cdots$. The feedback system can thus be built by successive iterations which give terms of increasing order (if the open-loop system is nonlinear), modifying at the same time the terms of lower order. The general Volterra series is sketched on the extreme right: notice that the first second order kernel results from contributions from every term in the iteration series indicated in the middle. **b** shows a similar example for a network with many inputs and lateral multiplicative interactions. **c** shows a network of processors (P) interacting locally with other processors. Notice that although the interaction is local, the p-order of the Volterra representation is infinite because of the recursive structure

corresponds to the Volterra graph  with $k_{12}(\tau_1,\tau_2)=k_1(\tau_1)k_2(\tau_2)$. Note that

(1.15)

corresponds to  with $k(\tau_1,\tau_2)=k(\tau_1)\delta(\tau_2-\tau_1)$, i.e. the input is first squared and then filtered by a linear system.

This is of course a mixed notation in the sense that it contains the general graphs of Table 1 but also special information about the structure of a specific system. It may often be interesting to use the graphic notation in this mixed, flexible way. For instance, we write for the system at the left



(1.16)

where it is clear that the second diagram is in mixed notation; the general Volterra notation of the right hand side does not make explicit the structure of the kernels, which is, in this case, strongly constrained.

The graphical notation can also be used to make explicit properties of symmetry or invariances under some operation of the various graphs.

1.8. Combination of Systems

In forming new systems by interconnecting a set of given systems, four basic operations are usually used. They correspond to sum, composition, inversion and synthesis of a feedback loop. Formulae giving the kernels of the total system when the kernels of the subsystems are known, can be found in the literature (see especially Halme, 1972; Brilliant, 1959; Marmarelis and Marmarelis, 1978). Table 2 outlines graphically some of these formulae for the various cases.

1.9. Dynamical Systems, Feedback, and Volterra Representations

Systems and corresponding operators are very often implicitely defined by functional equations. Physical systems are in fact usually described in this way. Typically a functional equation has the form

$$\mathbb{K}(\underline{x},\underline{y})=0 \tag{1.17}$$

where $\underline{x}$ is the (vector) input and $\underline{y}$ is the output. When $\mathbb{K}$ is an integral operator Eq. (1.17) is an integral equation; if it is a differential operator Eq. (1.17) is a

Table 2a

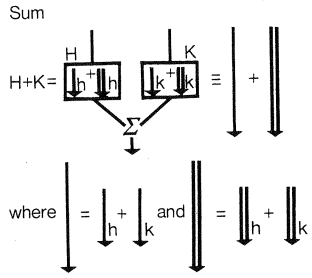


Table 2b

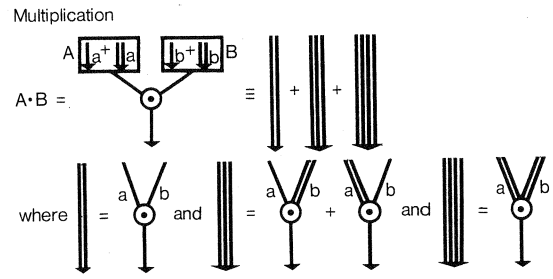


Table 2c

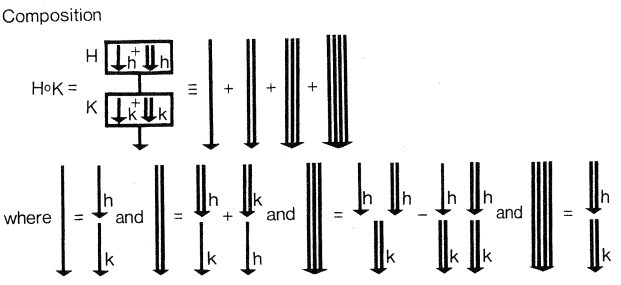


Table 2d

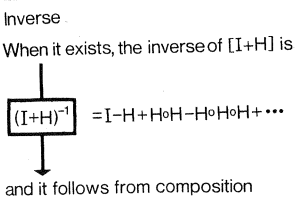
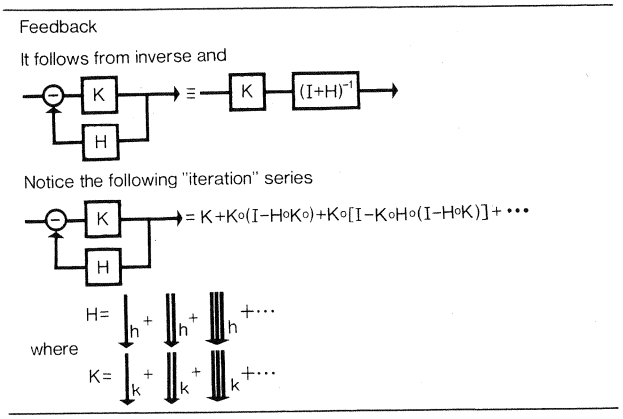


Table 2e



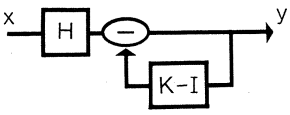


Fig. 5. The figure shows the feedback system equivalent to the functional equation $Ky = Hx$

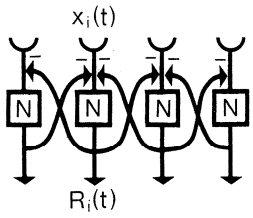


Fig. 6. A nonlinear recurrent, lateral inhibitory network. Its Volterra representation, if it exists and converges, can be obtained with standard methods (Poggio and Torre, 1977)

differential equation, which represents the most common implicit representation of a physical system. The problem of finding the solution of Eq. (1.17) is strictly connected to the implicit function theorem in Banach spaces. If Eq. (1.17) can be rewritten as

$$Hy = Qx$$

the problem is connected with the problem of finding an inverse of the operator H. Although a global inverse does not in general exist, local inverses usually exist and can then be expressed through polynomial operators of the Volterra type. A functional equation of the type of Eq. (1.18) can be always represented as a feedback system (Fig. 5). Thus the solution of a functional equation, the representation of a feedback system and the inversion of an operator are essentially one and the same problem. Implicit function theorems in Banach spaces can be usefully used in this context (see Berger, 1977; Holtzmann, 1970; Halme et al., 1971). Several techniques are thus available, that provide the Volterra series representation for either a feedback system or the inverse of a system or the solution of a differential equation. Notice that even when the equation itself is time invariant, in the sense that its parameters are time independent, the associated operator may be time variant, depending on the initial conditions. This is the reason for the appearance of subharmonics in some nonautonomous differential equations. It is always possible, however, to obtain a time-invariant operator, representable through a Volterra series from a nonlinear differential equation (with time independent parameters). As an example one can consider the forced Van der Pool equation with a limit cycle behaviour (for zero input). We only mention that two practical problems arise in connection with dynamical systems. The first problem is an analysis problem: how to determine the kernels associated with given differential equations. Various simple methods are available for this purpose: the use of local inverse theorems (Halme et al., 1971), the harmonic input method (Bedrosian and Rice, 1971) and others. Secondly, there is a "synthesis" problem: given a Volterra series it is possible to find a dynamical system corresponding to this representation? The problem must be of course suitably specified in order to be tractable. No satisfactory answer is known yet. Only some particular results are known. For instance, a particular class of dynamical systems (the so called bilinear systems) have a specific Volterra representation and it is easy to compute the parameters of the differential equation from knowledge of the kernels (see for instance D'Alessandro et al., 1974). Another case is represented by Volterra systems with separable kernels $[k_n(\tau_1, \tau_2, \ldots, \tau_n) = k_1(\tau_1) \ldots k_n(\tau_n)]$ which can be transformed in a standard way into a dynamical system (Harper and Rugh, 1976).

The treatment outlined here also applies to many input, many output systems. In particular, nonlinear networks with recurrent interactions can be represented through Taylor (Volterra) expansion around some stable, "zero" input. As an example, consider the 1-dimensional Limulus-like network with recurrent interactions shown in Fig. 6. The kernels, despite their complex form, can be found iteratively using the substitution method.

The recent development of fast algorithms for manipulating formal power series (see for instance Brent and Kung, 1978) can be very helpful for actually computing the kernels of rather complex interconnected systems.

2. Symmetry Properties

Invariance properties of the operations performed on the visual input correspond to properties of the underlying interactions (and to invariant "properties" of the patterns). In this chapter we outline how invariance of a polynomial functional under some transformation of the input pattern restricts the structure of the corresponding kernels. We will discuss a specific application to the "minimal" algorithm computing directional movement in a forthcoming paper (Bülthoff et al., 1980).

Let G be a group and V a linear vector space. We denote by $\mathscr{L}(V)$ the class of all linear transformations mapping V into itself one-to-one. A representation of G on V is a homomorphism from G onto the set $\mathscr{L}(V)$. Thus, $g \to T_g$ in such a way that

$$T_{g_1} \circ T_{g_2} = T_{g_1 g_2}. \tag{2.1}$$

We consider V to be the space of continuous functions on $\mathbb{R}(X \times Y \times T)$ and $G(3)$ some group consisting of 3×3 matrices. Then, the family of operations defined by

$$T_g(f)\underline{r} = f(O^{-1}\underline{r}) = f(\underline{r}') \qquad \underline{r} = (\underline{x}, t) = (\psi, \vartheta, t) \tag{2.2}$$

constitutes a group representation of $G(3)$ on V.

If S is a functional on the linear vector space V, we say that S is invariant under the group G if

$$S\{i(\underline{r})\} = S^* = S\{T_g i(\underline{r})\} \qquad \forall g \in G. \tag{2.3}$$

We consider here invariant polynomial functionals. In particular, we study the effect of the invariance property Eq. (2.3) on a general term of Eq. (1.8) of order j

$$S = : \int \dots \int \Phi(\underline{x}_1, \dots, \underline{x}_j; \tau_1, \dots, \tau_j) \prod_{l=1}^{j} i(\underline{x}_l, \tau_l) dm(\underline{r}_l). \tag{2.4}$$

Substituting Eq. (2.2) into Eq. (2.4) we obtain

$$S^* = S\{Tf(\underline{r})\} = \int \dots \int \Phi(\underline{x}_1, \dots, \underline{x}_j; \tau_1, \dots, \tau_j) \cdot J\left(\frac{O^{-1}\underline{r}}{\underline{r}}\right) \prod_{l=1}^{J} i(\underline{x}'_l, \tau'_l) d\underline{r}'_l, \tag{2.5}$$

where J is the Jacobian of $\underline{r}' = O^{-1}\underline{r}$. We consider, in the following, measure preserving transformations for which the Jacobian

$$J\left(\frac{O^{-1}\underline{r}}{\underline{r}}\right) = 1. \tag{2.6}$$

Then Eq. (2.5) becomes

$$S^* = \int \dots \int \Phi(O\underline{r}'_1, \dots, O\underline{r}'_j) \prod_{l=1}^{j} i(\underline{r}) d\underline{r}_l. \tag{2.7}$$

Condition (2.3) then implies

$$\Phi(O\underline{r}_1, \dots, O\underline{r}_l) = \Phi(\underline{r}_1, \dots, \underline{r}_l). \tag{2.8}$$

Thus, invariance of the algorithm S under the operation T implies that the kernel structure is restricted by Eq. (2.8). Of course, the conditions can easily be translated in terms of the N input representation Eq. (1.3) and the corresponding kernels $k_{i_1, \dots, i_M}$. In the case of N receptor the kernels, associated with a system invariant to the transformation g, will be invariant under some element of the permutation group S_r. For instance, the equation

$$\Phi(\underline{r}) = \Phi(O\underline{r}) \tag{2.9}$$

implies that

$$k_{i_1, \dots, i_J}(t - \tau_1, \dots, t - \tau_j) = k_{P\{i_1, \dots, i_J\}}(t - \tau_1, \dots, t - \tau_j), \tag{2.10}$$

where P is the element of the permutation group S_J corresponding to O, i.e. P is defined by

$$\int \dots \int \Phi(x_1, \dots, x_j; t - \tau_1, \dots, t - \tau_j) \cdot \prod_{l=1}^{J} i(O\underline{x}_l, \tau_l) dm(x_1) \dots dm(x_j) d\tau_1 \dots d\tau_j = \sum_{i_1 \dots i_j}^{N} \int \dots \int f_{P_{i_1}}(\tau_1) \dots f_{P_{i_j}}(\tau_j) \cdot k_{i_1, \dots, i_M}(t - \tau_1, \dots, t - \tau_j) d\tau_1 \dots d\tau_j, \tag{2.11}$$

where P_{i_1} is the "permuted" i_1. This case will be explicitly discussed in the next chapter in the context of the group invariance theorem.

Consider, for example, the case of the translation group. Notice that we must use as the underlying "retinal space" a torus, in order to deal with an infinite Euclidean transformation group. Of course, for "small" translation and "small" figures there is no important difference between the torus and the plane.

In the translation case

$$O^{-1}\underline{r} = r' = (\underline{x} - \underline{a}, t). \tag{2.12}$$

Thus, translation invariance of a j-order monomial system implies

$$\Phi_J(\underline{x}_1 + \underline{a}, \dots, \underline{x}_J + \underline{a}; \tau_1, \dots, \tau_J) = \phi_J(\underline{x}_1, \dots, \underline{x}_J; \tau_1, \dots, \tau_J) \tag{2.13}$$

In particular, Eq. (2.11) says that first order, translation invariant kernels are constant on $\underline{x}$

$$\Phi_1(\underline{x}_1, \tau_1) = \Phi_1(\underline{0}, \tau_1) \tag{2.14a}$$

and translation invariant second-order kernels depend only on the "distance" of interacting inputs

$$\Phi_2(\underline{x}_1, \underline{x}_2; \tau_1, \tau_2) = \Phi_2(0, \underline{x}_2 - \underline{x}_1; \tau_1, \tau_2). \tag{2.14b}$$

In the discrete representation, conditions Eqs. (2.14) become

$$k_i(\tau) = k(\tau), \tag{2.15a}$$

$$k_{ij}(\tau_1, \tau_2) = k_{i-j}(\tau_1, \tau_2). \tag{2.15b}$$

Note that in the discrete representation, spatial and temporal variable have a different nature, the first being discrete and the second ones continuous. In the context of group arguments it may therefore be convenient to discretize the time variable as well. There is essentially no loss in generality and the transformation O is then discrete for both spatial and temporal variables.

Later, in Chap. 3, we will discuss a case in which the time variable is more directly involved. The discussion in this paragraph is equivalent to the group theorem, originally stated for perceptrons and to be introduced in the next section, where it will become

clear how conditions like Eqs. (2.14) and (2.15) can often characterize which computations can be performed by the associated interactions.

3. Computational Properties

The subject of this chapter will be treated in more detail in a forthcoming book on discrete system theory (Palm and Poggio, 1980).

3.1. Canonical Decomposition of an Algorithm

A power series representation of a functional like Eq. (1.10) is essentially a canonical decomposition of the system into the sum of simpler, standard components. Computational properties of the mapping are then "additively" determined by the computational properties of the standard components, i.e. k-linear forms (represented as graphs in Table 1). Typically one would like to know what is the "simplest" set of graphs that can perform a given computation. And, conversely, it is clearly interesting to know what can a given mapping compute, if some rather general information about its polynomial decomposition is available.

We will give an outline of a computational theory of polynomial mappings, much in the spirit of the theory of Perceptrons (Minksy and Papert, 1969).

In the following we consider "discrete" representations of mappings in the form of Eq. (1.10), i.e.

$$S\{x_1(t), \ldots, x_N(t)\} = S_0 + \sum_{i=1}^{N} \sum_{j} L_{j_1 \ldots j_i}\{\underline{x}(t)\}, \qquad (3.1)$$

where $L_{j_1 \ldots j_i}$ is an i-linear form in the i components labelled $j_1 \ldots j_i$ (not necessarily distinct) of the input vector $(x_1(t), \ldots, x_N(t))$. L has, of course, an integral representation in terms of a kernel. By a retina R we mean here a (2-dimensional) collection of $|\mathrm{R}| = N$ "photoreceptors" (inputs) arranged on a hexagonal lattice on a torus and by a pattern on the retina a set of input functions $x_1(t), \ldots, x_N(t)$, seen by the photoreceptors. The inputs are functions of time $x_1(t) \ldots x_N(t)$, $x_i(t)$: $\mathbb{R} \to \mathbb{R}$ and can be assumed to be continuous. It may often be convenient to discretize time. In this case $x_i(t_j)$: $\mathbb{N} \to \mathbb{R}$ and the integrals over time appearing in the integral representation of $L_i\{x_1, \ldots, x_i\}$ become finite or infinite sums over the (discrete) time set. Thus, for instance,

$$\int x_i(t-\tau) k_i(\tau) d\tau \to \sum_j x_i(\tau_j) k_i(t-\tau_j). \qquad (3.2)$$

When $x_i(t) = x_i \in \mathbb{R}$, and S is a polynomial function of degree N, Eq. (1) is an analogue perceptron of the polynomial type (Uesaka, 1975). In this case, the input pattern is a grey level "figure" on the retina. If the x take only 0 and 1 values the pattern is essentially a "geometric figure" as considered in "Perceptrons" by Minsky and Papert (1969), see also Abelson (1977).

The canonical decomposition Eq. (1) can be also given as in Fig. 1a. We shall often use this notation. The canonical decomposition of Fig. 1a represents a N input system as the sum of $1 + N + N^2 + \ldots + N^M$ systems (1 constant, N linear, N^2 quadratic, $\ldots N^M$ m-linear), i.e. $\dfrac{1 - N^{M+1}}{1 - N}$ "elementary" systems (for example if $N = 100$ and $M = 4$ than there are 10101 such systems, where M is the maximum degree of nonlinearity). Many of the graphs are often identically zero, e.g., only the first $N + 1$ graphs may be non zero if S is linear.

From such a computational point of view, one is naturally interested in representing "complex" systems in terms of the sum of "simpler" graphs. The idea is then to synthetize the computational properties of the system as the "sum" of the properties of these standard graphs. We have already explained what we mean by canonical graphs. It is important now to define the notion of "simplicity". Intuitively, in the canonical representation Eq. (1) the graphs become increasingly complex going from left to right. The concept of simplicity of a graph can be formalized in terms of the notions of "degree" and "p-order".

3.2. Degree, p-Order and Rank

Definition. A canonical graph is said to be of degree k, if it corresponds to a k-linear form.

The degree of a graph is the total number of incoming lines (compare Tables 1 and 2); it is essentially the degree of the nonlinearity. Linear graphs have degree 1, quadratic ones (bilinear forms) have degree 2.

More important than the degree of a graph, as a measure of its complexity, is its p-order.

Definition. A canonical graph has p-order h, if it has inputs from h distinct photoreceptors.

Because of the structure of the canonical representation, the p-order of a graph is always less than or equal to its degree ($h \leq k$). p-order corresponds essentially to the concept of order in the perceptron framework. Therefore, the name p-order (see Poggio and Reichardt, 1976).

It is clearly possible to speak of the degree and p-order of a polynomial system, in the usual sense (compare the normal use of the term degree of an algebraic polynomial):

Definition. The degree of a polynomial system is the maximum degree of any of the graphs in its canonical

decomposition; its p-order is the maximum p-order of any of the graphs in its canonical decomposition.

Finally we introduce the definition of rank *Definition.* The rank of a polynomial system is the number of non-zero, distinct canonical graphs.

Thus, the rank of a polynomial system of degree M, with N inputs is at most $\frac{1-N^{M+1}}{1-N}$; the rank of an actual polynomial algorithm is usually much lower.

Although rank, degree and p-order, all help to characterize a polynomial algorithm, the p-order is the single most important measure of the simplicity of a graph or set of graphs. Linear graphs are the simplest components (p-order 1 and degree 1); the p-order of the graphs increases monotonically from left to right in the canonical representation Eq. (1). Furthermore, the notion of p-order is an appealing characterization of "local versus global".

The most interesting issues about parallel computation, revolve around the notion of "local and global" or "parts and wholes". Loosely speaking, a computational problem is inherently local if it can be divided into small, non-interacting modules. It is inherently global if any way of dividing it into subcomponents must entail substantial interaction among the modules. The concept of p-order formalizes the issue of local and global in the framework of the polynomial representation of a system or an algorithm. A low p-order system is local (and non recurrent!): the canonical subsystem make independent computations based on small patches of the retina. A high p-order computation is global: individual graphs receive inputs from many photoreceptors in the retina. More precisely, a computation is said to be global (in a polynomial representation) when the p-order of the simplest algorithm which can perform it, depends on the size of the retina and becomes arbitrarily large as the size of the retina (i.e. the number of photoreceptors) becomes large. One can ask if a certain computation is of finite p-order, i.e. if it can be computed by a polynomial mapping of some fixed p-order, regardless of the size of the retina (compare Minsky and Papert, 1969, Sect. 1.6).

It is clear that the definition of complexity of a computation, depends very critically on the representation that we have used. Different representations could be used as well and in their framework a computation which can be performed only with high p-order in a polynomial framework, may be very simple (and conversely). For example, an alternative "representation", beside the polynomial one, is the cascade representation: a mapping is represented in terms of a sequence of linear systems and 1-input nomemory nonlinearities. Other "representations" are restricted to specific sets of inputs: for example, in the perceptron framework, time is "frozen" and the input patterns are "static" figures on the retina.

It is thus important to compare our polynomial systems with some of the more common representations, from a computational point of view. For example, we would like to know whether a polynomial mapping is computationally less "powerful" than a perceptron; and what is its relation to universal computers. In the next section we will outline some answers to this questions. The previous chapter on approximation is fundamental for our treatment.

3.3. The Computational Power of Polynomial Systems

Several abstract computational machines have been conceived and their properties characterized (at least in part). We would like to compare some of them with our polynomial systems. The main question is whether a certain machine is computationally equivalent (or at least as powerful) as another one. The comparison strongly depends on the input set. We consider here a discrete input set, since we want to compare the computational power of polynomial mappings with standard classes of machines – which are only defined for this type of input. In particular we consider discrete time inputs to the photoreceptors; furthermore, the values taken by the input are restricted to a finite (in the extreme case to a binary set, i.e. figures on the retina with values 0 and 1). We also assume that the input vectors have a finite number of components i.e. there are a finite number of time "points".

Typical machines that may be considered are:

1) finite state machines,

2) cascades (with and without loops) (see Palm, 1979).

3) McCulloch-Pitts networks (with or without loops) (McCulloch and Pitts, 1943),

4) difference equations and feedback systems,

5) perceptrons (Minsky and Papert, 1969),

6) analog perceptron (Uesaka, 1971),

7) linear threshold machines (Abelson, 1977).

We distinguish now two cases. First, let us assume that the input vectors have a finite number of components, i.e. there are a finite number of "time points". In this case the set of input functions is finite, implying (see Chap. 1), that polynomial functionals can represent exactly all mappings. In particular, it is always possible to determine a polynomial mapping that simulates exactly the behaviour of one of the machines listed above for such inputs.

As a second case, we relax the assumption of a finite time set. The input vectors can have now in-

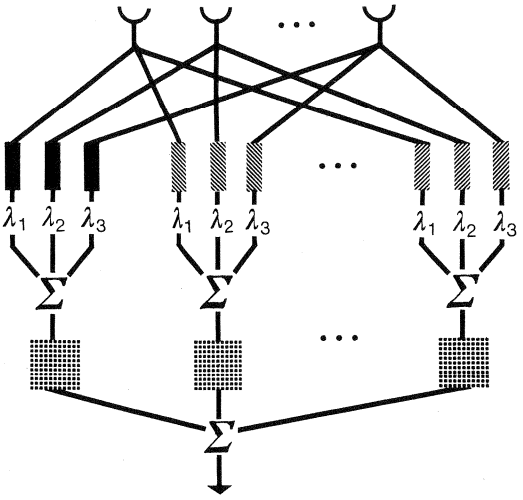


Fig. 7. Kolmogorov's and Arnold's solution to Hilbert's 13th problem. A continuous function of n variables can always be represented as the superposition of functions of 1 variable. Here $n=3$ and

$$f(x_1, x_2, x_3)=\sum_q^7 g\left(\sum_p^3 \lambda_p \Phi_q(x_p)\right).$$

There is no such decomposition of a differentiable function (in differentiable functions)

finitely many components. Systems with finite memory can still be represented exactly by polynomial mappings (since the effective input set remains in fact finite). This is clearly the case for cascades without loops (with a finite number of stages) and all other forms of loop free cascades (i.e. McCulloch-Pitts nets without loops, perceptrons, analog perceptrons, linear threshold machines). Finite state machines cascade with loops and difference equations, however, can only be approximated by polynomial mappings (depending on the specific machine). These problems will be discussed in more detail elsewhere (Palm and Poggio, 1980). Thus, finite state machines, cascades with loops, and difference equation systems are, for discrete-time inputs, more powerful than polynomial mappings. Although they are not universal Turing machines, they are practically universal computers, since every machine we can build can be approximated as closely as we like by defining sufficiently many stages. In practice, the time is always finite and, therefore, polynomial mappings are equivalent to finite state machines and cascades with loops. This situation, however, indicates that polynomial representations may often become too cumbersome to be useful. Theoretically one can always represent a feedback system with a polynomial representation if time is finite; but the order of the representation usually becomes larger if the time interval becomes larger, often leading to unmanageable representations. In a somewhat analogous fashion, the polynomial representation of a McCulloch-Pitts neural network will usually be much more "complex" than the original machine. The choice of the appropriate representation of a given computation and thus the associated complexity clearly depends on the type of elementary components and connections that are available. If the visual system of the fly were to consist of McCulloch-Pitts threshold elements, we would use (see Bülthoff et al., 1980) a much too cumbersome representation (the Volterra series). If, however, the computational elements are smooth and perhaps analytic [like in the case of postsynaptic interactions, Poggio and Torre (1978)], then a polynomial representation is certainly better than a McCulloch-Pitts one. It is also possible that mixed representations prove to be, at the end, a good compromise: some subsystems may be represented in one way, others in another.

Interestingly, this perspective on the complexity of mappings is related to Hilbert's 13th problem [brought to our attention by D. Marr, which concerns the possibility of representing functions of several variables as superposition of functions of a smaller number of variables. In fact, Kolmogorov (1957) and Arnold (1963)] have shown that any continuous function of n variables can be always represented as in Fig. 7. This result essentially shows that all continuous functions can be reduced to one stage systems (modulus a change of scale in the various inputs given by the ▮) and thereby hints at the universality of "Taylor" representations.

If we now consider discrete time mappings, we can easily interpret the Kolmogorov result within our framework. In discrete time a polynomial functional with separable kernels can approximate arbitrary well (all topologies are equivalent) every continuous mapping (see Sect. 1.4). Exactly this type of Volterra polynomial is canonically provided by the Kolmogorov representations of Fig. 7, if the inputs are filtered through suitable linear filters, ▮ are all log transformations and ▩ are exponential transformations. Note that in a certain sense Kolmogorov result shows that only continuous functions of 1 variable exist. On the other hand, it is easy to show that this result does not hold for continuous time functionals (Palm, 1978). Furthermore, the Kolmogorov result shows that in a "cascade" representation, the number of stages cannot be taken as a measure of complexity and again hints at the Taylor representation as more natural for defining the "globality" of a mapping (see also Palm, 1979).

In any case it is interesting and encouraging to know that in principle polynomial systems are, for appropriate inputs, universal and equivalent to other, classical classes of machines. We will now give with some more details the connection between polynomial mapping on one side and perceptrons and analog perceptrons on the other.

3.4. Connections with Perceptrons

Let us first recall a few definitions.

In the perceptron framework, the retina R is equivalent to our retina; a "figure" is a pattern on the retina (see Fig. 8) at one given time t and with values (seen by the photoreceptors) 0 or 1. A predicate on R is a function ψ from a figure of R to $\{0,1\}$. A perceptron is a predicate of the form

$$\psi(R)=\lceil\sum a_i\varphi_i(R)>\vartheta\rceil, \quad a_i\in\mathbb{R}e, \tag{3.4}$$

where $\lceil$some condition$\rceil$ is 1 if the condition is true and 0 if the condition is false (see Fig. 9). The support of φ is the set of all photoreceptors for "points") of R which affect the value of φ, and the order of φ is the size of support (φ). Thus, a predicate φ is said to be of order k if φ makes its decision by examining at most k points of R. The order of the perceptron ψ is the maximum order of any of the φ. A perceptron of order 1 is precisely what is usually called a linear threshold function on R. Observe that of the 16 Boolean functions of 2 variables, all have order 1 except for exclusive-or ($x\oplus y$) and its complement identity ($x\equiv y$), which are of order 2

$$x\oplus y=\lceil x\bar{y}+\bar{x}y>0\rceil$$

$$x\equiv y=\lceil xy+\bar{x}\bar{y}>0\rceil.$$

The connection between polynomial mappings and perceptrons is relatively straightforward: of course we restrict the comparison to inputs for which Perceptrons are defined, i.e. "time-frozen" figures on the retina which are only black and white (only 0, 1 values). The following simple result holds:

Lemma. *Any (perceptron) φ function of support n can be represented exactly for all figures by a set of polynomial graphs of p-order n and degree $\leqq[N-1]$, $N=2^n$.*

Proof. The input set is $\{0,1\}^{|R|}$, and is therefore finite; time is frozen. For each function φ of order n defined on the input set $\{0,1\}^n$, a Lagrange polynomial can be constructed that on this finite set of points $N=2^n$ attains the same values as φ. The degree of the polynomial is at most $N-1$. □

Thus, support 2 predicates can be implemented through polynomial graphs of maximal p-order 2 and degree 3; support 4 predicates can be implemented through graphs of at most p-order 4 and degree 15. Although the degree of the polynomial equivalent to a predicate of order n, grows very quickly with n, its p-order, which is the most important measure of complexity, remains n.

One may think that the p-order of the polynomial mapping representing a perceptron of order n – and not simply a φ function – may be larger than the maximal p-order needed for the φ functions, because of the threshold operation. The next theorem shows, however, that this is not so. Thus, polynomial mappings can exactly represent, on this input space, any Perceptron. Observe that interpolation with Lagrange polynomials is always possible also when the input space is a Banach space and the function to be interpolated is a mapping between two Banach spaces (Prenter, 1971 and Sect. 1.4).

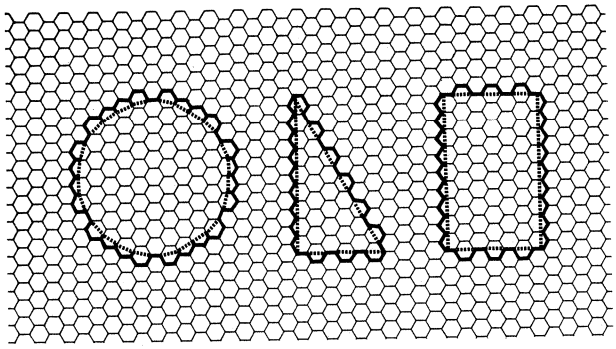


Fig. 8. Geometric figures on a hexagonal grid

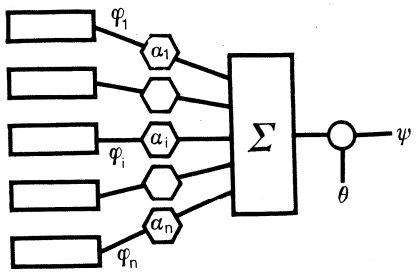


Fig. 9. A perceptron

Theorem. *For input figures on the retina $\{0,1\}^{|R|}$, there always exists a polynomial mapping of p-order n and degree 2^n-1 which exactly represents a given perceptron of order $\leqq n$. Conversely any given polynomials taking only values 0, 1 on this input space can be represented by a perceptron of order not greater than its p-order. Thus, perceptrons of order $\leqq n$ and polynomials of p-order n and degree 2^n-1 are equivalent on this input and output space.*

Proof. The proof makes use of the concept of mask and the "positive normal form theorem" (Minsky and Paper, 1969, p. 34). A "mask" is a predicate of the type $\varphi_A=\lceil A\subset X\rceil$. For instance, if

$$A=\{y_1,\ldots,y_n\}\varphi_A(x)=y_1\cdot\ldots\cdot y_n.$$

All masks are perceptrons of order 1. In addition every perceptron of order k can be represented as another perceptron that uses only masks with support $\leqq k$. This

perceptron, called the positive normal form,

$$\psi(X)=\sum_i a_i\varphi_i(X) \tag{3.5}$$

is unique. The values of Eq. (3.5) are 0 and 1 as ordinary arithmetic sums, i.e. without requiring the $\lceil\ \rceil$ device of interpreting the validity of an inequality as a predicate. In other words, $\sum_i a_i\varphi_i(X)$ is the same as $\left\lceil \sum_i a_i\varphi_i(X)>0 \right\rceil$. Given a perceptron of order n, write it as its positive normal form. Then use the previous lemma for each of its φ functions (the masks). Their maximum support is n. The first part of the theorem follows. For the second part of the theorem it is enough to observe that a polynomial of p-order n assuming only the values 0, 1 on this input space is a predicate. Such a predicate can be represented as a linear threshold function of order $\leqq n$. □

It follows, then, that all available results on the order of perceptrons which compute various geometrical predicates also apply to the p-order of the corresponding polynomial mappings, of course for Perceptron-type inputs (dark "frozen" figures on white background as in Fig. 8) for which geometrical (perceptron) computations are defined.

Thus,

$\lceil X$ is locally convex$\rceil$

can be computed with a perceptron of order 3 and therefore, with a Volterra-like polynomial of p-order 3 (and degree 7). Table 3 lists some of the other computations considered by Minsky and Papert. Especially interesting is the question of whether a computation is of finite p-order, i.e. if it can be computed by a polynomial mapping of some fixed p-order, regardless of the size of retina. Computations like $\lceil X$ is connected$\rceil$ are not of finite p-order. In other words, for such computations the polynomial decomposition is not very succesfull (or not at all) in splitting an algorithm with n inputs into "simpler" algorithms with fewer inputs. Since we will be mainly dealing with a fixed retina, the infinite p-order problem may not concern us too much. For a fixed size retina the p-order is always finite. Infinite p-order, however, usually means that the p-order or a fixed size retina is of the order of the number of photoreceptors. A polynomial representation can hardly be said to be "simple" under these conditions. For these computations, other descriptions should be used. A polynomial representation is an useful way of describing such algorithms that are "simple" enough, i.e. local enough to be representable in terms of low p-order polynomials. Theorem 1 shows that the severe limitations of Perceptrons as global computational schemes carry on to polynomial algorithms (restricted to the same class of inputs). It is very likely that similar limitations also hold for polynomial mappings defined on more general time dependent input pattern.

3.5. *Connection with Analog Perceptrons*

Analog Perceptrons are, essentially, Perceptrons with real valued inputs and outputs (Uesaka, 1971, 1975). An analog perceptron is a real valued function $f:R\to\mathbb{R}$, where R is the retina of N photoreceptors, looking at real valued figures ($R=\mathbb{R}^N$) such that

$$f(X)=\sum_{A\in R} u_A(X), \tag{3.6}$$

where $u_A(X)$ are real valued functions. The order of f is the maximum of the supports of the u_A. When in

Table 3. p-Order of some computations on geometrical figures

Computation	p-order	Computation	p-order
$\lceil X$ is a circle$\rceil$	4	Translation invariant recognition of figures	3
$\lceil \lvert X\rvert \geqq M\rceil$	1	$\lceil X$ is odd$\rceil$	$\infty(\lvert R\rvert)$
$\lceil \lvert X\rvert \leqq M\rceil$	1	$\exists\,\psi_1,\psi_2$ p-order $(\psi_1)=1$	∞
$\lceil \lvert X\rvert = M\rceil$	2	p-order $\psi_2=2$: $\psi_1\wedge\psi_2$	
$\lceil$The n-th central moment of X about the origin is $>\vartheta\rceil$	1	$\lceil X$ is connected$\rceil$	$\infty(>c\lvert R\rvert^{1/2})$
All Boolean functions of 2 variables, except $x\oplus y$ and $x\equiv y$	1	Translation invariant geometric spectra ψ in context	2 ∞
$x\oplus y$	2	$\lceil X$ is a hollow square$\rceil$	3
$x\equiv y$	2	$\lceil$One component of X is a hollow square$\rceil$	∞
$\lceil X$ is convex$\rceil$	3	$\lceil X$ has symmetry under reflection$\rceil$	$\leqq 4$
		$\lceil$Pattern A is a translate of pattern $B\rceil$	$\leqq 5$

The p-order of the polynomial algorithm that performs some computations on "geometrical" figures. The input space is $\{0,1\}^{\lvert R\rvert}$, where $\lvert R\rvert$ is the total number of photoreceptors; the values seen are either 0 or 1. The results are translated from the Perceptron framework via the Theorem of Sect. 3.4 (see Minsky and Papert, 1969). p-order $=\infty$ means that the p-order depends on the number of photoreceptors and grows with it

Eq. (3.1) $x_i(t)=x_i$ and S is a polynomial function, Eq. (3.1) represents an analog perceptron of the polynomial type. One can again consider various computations and the associated order. The following (trivial) propositions are of interest:

Proposition 1. *A polynomial mapping [Eq. (1)] is an analog perceptron of the polynomial type, when time is "frozen", i.e. when* $h_n(\tau_1, \ldots, \tau_n)=h_n\delta(\tau_1)\ldots\delta(\tau_n)$. *The p-order equals the order.*

Proposition 2. *For every analog perceptron of order n on real valued figures* $[0,1]^N$, *there exists a polynomial of p-order n and degree high enough to approximate it arbitrarily well. One can thus obtain Table 4, depicting the p-order of some computations performed by a real valued polynomial algorithm on real valued figures.*

We conclude here this comparison of polynomial mappings with standard computational machines. In order to carry out this comparison we had to restrict the input space of the polynomial mappings to the input set for which the other machines are usually defined. In the next section we will give some example of theorems which can be used to characterize the computational properties of polynomial algorithms with the (unrestricted) input space defined in Eq. (1). In particular, input patterns need not be "frozen" in time.

3.6. Some Results on Polynomial Algorithms: Group Invariance and Collapsing Theorem

We consider here the invariance of a computation under a group of transformations. When such an invariance exists, it usually implies that the corresponding polynomial algorithm has a simple and characteristic structure. We are actually taking here a point of view which is complementary to Chap. 2.

Let G be a finite group of transformation and V the space of the time functions, seen by the photoreceptors $x_1(t), \ldots, x_N(t)$. For our purposes, we consider such groups G that are equivalent to a permutation group on the finite space of the photoreceptors, i.e. the retina. In other words, the groups we consider here do not act on the time variable. The retina can be considered to be a torus.

Each term of Eq. (1) is a multilinear form, which we call a graph. Each multilinear form has an integral representation

$$\begin{aligned} L_{1,\ldots,J}\{x_1,\ldots,x_J\} &= \int\ldots\int k_{1,\ldots,J}(\tau_1,\ldots,\tau_J) \\ &\quad \cdot x_1^t(\tau_1)\ldots x_J^t(\tau_J)d\tau_1\ldots d\tau_J \\ &= k_{1,\ldots,J} * {}^J(x_1\ldots x_J), \end{aligned} \tag{3.7}$$

where k is the kernel of the graph, in other words its "coefficient", and the product $(x_1\cdot\ldots\cdot x_J)$ is the "window" of the graph, i.e. the ordered set of inputs on which the kernel operates. Instead of Eq. (3.7) we will use here the notation $k_w * \ldots * W$, where W indicates some monomial $(x_1\cdot\ldots\cdot x_J)$ and w labels the associated kernels.

A term like $L_{13}\{x_1(t), x_3(t)\}$ is a graph. We consider (in principle) as different graphs all graphs occurring in the canonical representation.

The notation $\bigvee$ denotes a p-order 2, degree 2 graph but leaves arbitrary its kernel and its window.

Table 4. p-Order of some computations on grey-level figures

Computation	p-order	Degree
mult $(X)=x_1\,x_2\ldots x_n$	n	n
max $(X)=\max(x_1,\ldots,x_n)$	n	
min $(X)=\min(x_1,\ldots,x_n)$	n	
uniform $(X)=\begin{cases}1 \text{ if } x_1=\ldots=x_n\\0 \text{ otherwise}\end{cases}$	n	
equal $(X;P)=\begin{cases}1 \text{ if } x_i=P_i\ i=1,n\\0 \text{ otherwise}\end{cases}$	∞	
tolerance $(X;P;\varepsilon)=\begin{cases}1 \text{ if } \lvert x_1-P_i\rvert\leqq\varepsilon \text{ for } i=1,n\\0 \text{ otherwise}\end{cases}$	∞	
$f(X)=(x_1+x_2+\ldots+x_n)^m$	$\min\{n,m\}$	m
$f(X)=\frac{1}{n}\sum_{i=1}^{n}x_i^m$	1	m
$f(X)=\frac{1}{n}\sum_{i=1}^{n}(x_1-\bar{x})^m$	$\min\{n,m\}$	m

The p-order of the polynomial algorithm that performs the above computations on "grey-level" figures. The input space is $\mathbb{R}^{|R|}$ where $|R|$ is the total number of photoreceptors; the values seen are real numbers. The results are translated from the Analog Perceptron framework (see Uesaka, 1971). When the degree is not given, its value is determined by the accuracy with which one wants to approximate the Analog Perceptron. In the other cases, the Analog Perceptron is of the polynomial type

The symbol $\overset{1}{\underset{}{\vee}}\overset{2}{}$ specifies a p-order 2, degree 2 graph together with its window $(x_1^I(\tau_1)x_2^I(\tau_2))$; the kernel is not given explicitly. Since each term in the canonical representation has a different window (each window is an ordered monomial) the canonical representation of S can be written as

$$S\{x\}=\sum_w k_w * * \ldots * W, \tag{3.8}$$

where the summation is over all windows; $* \ldots *$ indicates the required (multi-)convolution (see Chapter 1).

Given a group G, we will say that two patterns $X=\{x_1(t), \ldots, x_n(t)\}$ and $Y=\{y_1(t), \ldots\}$ are G-equivalent if there is $g\in G$ for which $X=gY$. In a similar way, we will say that two windows W and W' are equivalent with respect to G if there is $g\in G$ such that $W\{gX\}=W'\{X\}$ for every pattern X. Given a window W and a group element g, we define Wg to be that window which for each pattern X has the value $W\{gX\}$. We call $\mathcal{J}$ a set of windows on R. We will say that $\mathcal{J}$ is closed under G, if for every W in $\mathcal{J}$ and g in G the window Wg is also in $\mathcal{J}$.

Notice that on our discrete grid of photoreceptors, the translation group does not require any tolerance theory. The rotation group does; but all interesting points can be made in the context of 60° rotations (remember that we deal with a hexagonal lattice of photoreceptors).

We can now state, (following Minsky and Papert), the basic result of this section.

Group-Invariance Theorem. *If*

1) *G is a finite group of permutations on V;*

2) *$\mathcal{J}$ is a family of windows W and it is closed under G;*

3) *the polynomial algorithm S is invariant under G; its elementary graphs (in the canonical representation) have windows in $\mathcal{J}$ then there exists a representation of S*

$$S=\sum_{w\in\mathcal{J}} k_w * \ldots * W$$

for which the kernels k_w depend only on the G-equivalence class of $\mathcal{J}$, i.e. if $W=W'$ then $k_w=k_{w'}$.

Proof. Each element $g\in G$ defines a permutation of the W's. Thus,

$$S\{X\}=\sum_{w\in\mathcal{J}} k_w * \ldots * W(X)=\sum_{w\in\mathcal{J}} k_{wg} * \ldots * Wg(X) \tag{3.9}$$

for all X, simply because the same numbers (at every t) are added in both sums. Since S is G-invariant, i.e. $S\{X\}=S\{g^{-1}X\}$

$$S\{X\}=\sum_{w\in\mathcal{J}} k_{wg} * \ldots * Wg(g^{-1}X). \tag{3.10}$$

Summing sidewise for all $g\in G$ we see that

$$|G|S\{X\}=\sum_{g\in G}\sum_{w\in\mathcal{J}} k_{wg} * \ldots * W(X)$$

$$=\sum_{w\in\mathcal{J}}\left(\sum_{g\in G} k_{wg}\right) * \ldots * W(X). \tag{3.11}$$

We sum for each w over g and obtain a new set of kernels

$$\sum_{g\in G} k_{wg}=k_w^*. \tag{3.12}$$

Thus

$$S\{X\}=\sum_{w\in\mathcal{J}} \bar{k}_w * \ldots * W(X), \tag{3.13}$$

where

$$\bar{k}_w=k_w^*\frac{1}{|G|}.$$

If $W=W'$ then $W=W'h$ for some $h\in G$ and thus

$$k_w=\frac{1}{|G|}\sum_{g\in G} k_{wg}=\frac{1}{|G|}\sum_{g\in G} k_{w'hg}$$

$$=\frac{1}{|G|}\sum_{g\in G} k_{w'g}=\bar{k}_{w'}. \ \square \tag{3.14}$$

Although we have essentially discussed a simple application of the group invariance theorem in Chap. 2 (for the translation group), we list here some of the simple consequences of the group invariance theorem in terms of the new concepts introduced here (like the concept of window). Notice that the invariance of the functional under the group G for all patterns $X=\{x_1(t), \ldots, x_N(t)\}$ is a stronger requirement than the invariance for all figures $\{x_1(0), \ldots, x_N(0)\}$ (as in the perceptron case).

Following Minsky and Papert with "geometric property" we mean something invariant under translation, usually rotation and often dilation or contraction. Observe that a visual system which does not have built-in preferred directions corresponds to an isotropic functional, i.e. a functional invariant under the orthogonal group.

A p-order 1 polynomial (on a torus) is

$$S=\overset{1}{\downarrow}+\ldots+\overset{n}{\downarrow}+\overset{1}{\Vert}+\ldots+\overset{n}{\Vert}+\overset{1}{\Vert}+\ldots$$

$$=\sum_w k_w * \ldots * W(X), \tag{3.15}$$

where the graphic representation makes clear what are the W windows. If S is translation invariant, that is

$$S=\sum_i\left\{\overset{i}{\downarrow}+\overset{i}{\Vert}+\overset{i}{\Vert}+\ldots\right\} \tag{3.16}$$

the polynomial is the same for each photoreceptor i and

$$S=\sum_w k*\ldots*W(X)$$
$$=k_1*\sum_i x_i^t+k_2**\sum_i x_i^t(\tau_1)x_i^t(\tau_2)+\ldots$$
$$=N[k_1*\langle x(\tau)\rangle+k_2**\langle x(\tau_1)x(\tau_2)\rangle+\ldots], \qquad (3.17)$$

where k_1 is a linear kernel, k_2 a bilinear one etc.; $\langle x(\tau)\rangle$ is the spatial average of the pattern values at time τ and $\langle x(\tau_1)x(\tau_2)\rangle$ is essentially a two-(time) point ensemble autocorrelation function of the pattern on the retina, and so on. Thus, a translation invariant p-order 1 polynomial system, being translation invariant, can only compute a linear combination of functionals of the time "autocorrelations" of the pattern (of various orders: order 1 is the ensemble average of the pattern, order 2 is the usual time autocorrelation of the pattern etc.). Observe that if translation invariance is not required, then polynomials of p-order 1 can, of course, compute other properties, for example, the time dependent position of a stimulated photoreceptor in the retina. Thus, an efficient eye-centering servomechanism can be based on a p-order 1 polynomial (see Sect. 6 and Minsky and Papert's footnote at p. 99).

If the system processes only one temporal frame of the input pattern, then it can only compute moments of the grey level figure about zero (see Table 5). If the degree is 1, the only translation invariant function is the mean intensity of the pattern. If the degree is arbitrarily high, a translation invariant, p-order 1 polynomial can compute arbitrarily high moments of the pattern and thus characterize its statistics.

Using similar arguments, a translation invariant, p-order 2, polynomial system has the form

$$S=\sum_w k_D*\ldots*W(X), \qquad (3.18)$$

where the windows W are all 1 and 2 photoreceptors windows and the kernel k depend only on the "difference vector" D between the two elements of the window. In the case of the system being a function (only a frame of the real valued pattern is considered), p-order 2 polynomials can compute the difference-vector moments of the pattern up to the degree corresponding to the polynomial degree. If the degree of the polynomial is arbitrary high, the full second-order statistics of the pattern may be computed with arbitrary accuracy. The second-order statistics of a texture is defined as the bivariate probability distribution of two pattern values $p(x_1, x_2)$; because of translation invariance it becomes the probability distribution of the difference *vector* values, i.e. $p(\varrho, \vartheta)$. If rotation invariance is added, it becomes the distribution of the *distance* values, i.e. $p(\varrho)$. Under rotation invariance the kernels k of Eq. (3.18) depend only on the distance of the two elements of the window; as a consequence only windows with different distances need to be considered in the polynomial representation. We note, incidentally, that this framework provides an interesting interpretation of Julesz' conjecture about discriminability of random textures (Julesz, 1975; Poggio and Reichardt, 1980). In particular, if the pixels (i.e. picture elements by the photoreceptors) of the texture have values 0 and 1, two figures may or may not be distinguished (because of the lemma) by p-order 2 polynomials according to their "difference vector spectra" (as defined by Minsky and Papert, 1969, p. 99).

Table 5. p-Order of some computations on time-dependent patterns

Computation	p-order	Degree
Directional movement (translation invariant)	2	2
Directional movement (translation- and rotation invariant)	1 or 2	2
Relative movement	2 or better 4	4
Absolute position of a stimulus on the retina (not invariant to any geometric group)	1	
Expansion or contraction of a texture (translation invariant)	$\geqq 4$	
First order statistics of a (timeless) texture	1	High
Second order statistics of a (timeless) texture	2	High
Expansion-concentration of a texture (around the coordinate origin)	2	2

p-order of the polynomial algorithm that performs some computations on real-valued time dependent signals from the receptors. For more details see Bülthoff et al. (1980)

Distance spectra, although still applicable through our concept of windows, cannot fully characterize the general time dependent situation. For instance, it can be shown (Bulthoff et al., 1980) that a p-order 2, degree 2, translation invariant (but not rotation invariant!) system can compute directional movement of a figure. This conclusion does not follow only from the group invariance theorem.

We discuss now the change of p-order induced by transformation of the retina. Let $R_1=\{x_1, \ldots, x_n\}$ and $R_2=\{y_1, \ldots, y_m\}$. For every pattern

$X=\{x_1(t), \ldots, x_n(t)\}$

defined on R, one obtains another pattern

$Y=\{y_1(t), \ldots, y_m(t)\}$

through the mapping f as follows

$$Y=\{y_1(t), \ldots, y_m(t)\}=f(X)=\{f_1(X), \ldots, f_m(X)\}.$$

Given a polynomial mapping S_2 on R_2, let S_1 be defined on R_1 through

$$S_1\{X\}:=S_2\{Y\}:=S_2\{f(X)\} \quad \text{for all } X.$$

We say that S_1 is a system on R_1 induced by f from a system S_2 on R_2.

Theorem. *Let S_1 be a polynomial functional on R_1, induced by $f: X^{|R_1|} \to X^{|R_2|}$ from a functional S_2 on R_2. If for $i=1, \ldots, m$ the support of f_i is at most equal to 1, then p-order $(S_1) \leqq$ p-order (S_2). If order $(f_i)>2$, then degree $(S_1)>$ degree (S_2).*

Proof. S_1 can be simply regarded as the composition of f and S_2. Clearly a sufficient condition for f not to increase the p-order of S is to have f_i looking for $\forall i$ at most one photoreceptor in the retina.

This simple theorem restates in a formal way that nonlinear coding of each photoreceptor input does not increase the p-order of a polynomial mapping acting on the photoreceptor output. Clearly, such a coding will in general affect the degree of the polynomial mapping.

Example. A useful f is the following one: $x_i(t)$ is coded into $x_i(t)=1$ if $x_i(t)$ satisfies some condition (not containing $x_j(t) j \neq i$); $x_i(t)$ is coded into $x(t)=0$ otherwise.

This function, together with the collapsing theorem, may often help to find bounds on the p-order of a computation.

Finally, we remark that some computations involving time may be translated in a "time frozen" framework with a construction which has the flavour of the collapsing theorem. Imagine a 1-dimensional retina R looking at some time dependent pattern; map this retina R on a 2-dimensional "static" retina, each successive row being the retina R at successive instants of (discrete) time. Now, the computation of a time-dependent property (for instance movement) is mapped into the computation of geometric properties of a 2-dimensional static figure.

3.7. *Multioutput Systems*

A multioutput polynomial system can be regarded as a sequence of systems like Eq. (1) and can be thus analyzed (componentwise) in the same way. Especially in the case of discrete systems it may be possible to have a more compact understanding of the input-output transduction (Palm and Poggio, 1980). It is clear, of course, that nonlinear, polynomial mappings can be considered as linear from a suitably "enlarged" input space (Poggio, 1975b; Kohonen, 1977). In this way, then, all linear techniques again be used. However, because of the high number of resulting input dimensions, such an approach is likely to be somewhat opaque.

3.8. *Germ Invariance*

We have considered in this section a polynomial system as given and studied its computational properties, in particular the p-order and the degree needed for a given computation. It is clear that in practice the degree and the p-order of a biological system performing a certain computation may be larger than the minimum value, depending for example on the input modulation and range.

At the input stage the range of the signals may vary over several orders of magnitude and no physical sensor is linear in such a range; saturation and range compression phenomena may occur at the output stage. In both cases the degree of the computation will become higher; furthermore in the second case the p-order will be larger. Usually such a large degree and p-order does not mean that a different or more complex computation is performed. Essentially, the same transduction takes place and only range compression occurs. In using the theoretical language discussed here we will often assume that a certain computation – detection of movement, say – will take place independently from inputs amplitude and mean values, over a large range. Under this assumption then, the key terms of the polynomial, responsible for the computation, must be present also under input and output conditions that minimize "trivial" input and output nonlinearities (like saturation effects). Small input modulation, for instance, tends to linearize all "trivial" nonlinearities and leaves "essential" nonlinearities, responsible for the computation. We call these "essential" terms "germ" of the polynomial algorithm. In other words, the germ of a computation is the characteristic part of its polynomial representation to which essential p-order and degree refer. Usually, nothing qualitative changes in a computation in so far as the "germ" does not disappear. Trivial nonlinearities resulting from high input modulation add terms of higher order without changing the germ – and without changing characteristic properties of the computation. Thus, we assume that for the computations to which our approach applies the germ is invariant over the whole input space on which the computation is performed.

When the mapping is an entire function, i.e. analytic in the whole input space, the previous assumption holds trivially. When the mapping is analytic only in the neighborhood of "zero" inputs, the previous assumption implies that the basic properties of its low order terms – the germ – do not change from one neighborhood to another one, although the numerical value of the associated kernels will in general change. In some sense, a "computational bifurcation" may be expected only when existent low order graphs – part of the germ – disappear and qualitatively new graphs appear, as a consequence of a parametric change in the mapping (for example because of changes of the "zero" input). In the same spirit one may argue that, given a

mapping, its computational properties may be studied through its low order Frechet derivatives around the "working point", i.e. the "zero" input $x(t)$. Similarly to the bifurcation theory of nonlinear functional equations, one would expect that qualitatively new outputs can only result from a bifurcation, characterized by the disappearance of the previous lowest order Frechet derivatives. Furthermore, one may at first conjecture that in biological systems that process information, the "germ" of the computation (to be found experimentally using, for instance, small inputs etc.) has the minimal p-order theoretically possible for that computation. Higher p-orders would usually be the effect of range compression and other "trivial" input or output nonlinearities.

3.9. Synthesis of a System: Associative Memory Algorithms

One can ask also a somewhat complementary question: how to synthesize a polynomial system from a given set of inputs and desired outputs? In other words, one can ask whether there exist learning algorithms that allow the synthesis of the various kernels. In the discrete case there exist general algorithms providing either directly or iteratively the required polynomial (Poggio, 1975a, b; Kohonen, 1977). Notice that, in the discrete case, it is thus possible to synthesize an arbitrary, nonlinear, finite memory mapping (compare Chap. 1). In particular, a nonlinear operation between boolean variables can be substituted by a linear associative mapping, implementing the corresponding "truth table". In the continuous case it seems possible to use, for instance, Coleman's and Mizel's results. This topic will be treated in detail in a forthcoming book (Palm and Poggio, 1980).

4. Conclusion

The analysis carried out in this last section shows that polynomial algorithms are in principle computationally powerful. It is clear, however, that the polynomial framework is often too cumbersome to be useful. Only "simple", parallel computations may be usefully described in terms of polynomial mappings. More complex, symbolic computations or even systems which include recurrent structures and reafferent controls may be better described with a different language. On the whole we believe that the computational limitations of perceptrons, as stressed by Minsky and Papert, also hold for polynomial descriptions of algorithms. In addition, it must be very clear that the decomposition of a nonlinear system in the set of standard graphs used in this paper is a conceptual device: in a physical (or biological) system individual graphs do not in general correspond to separate pieces of hardware.

Polynomial representations, however, may be useful for characterizing preprocessing operations in the nervous system that have an analogue, not yet symbol-like (or threshold logic-like) character. Algorithms for computing movement and relative movement for example, can be classified and analyzed rather exhaustively in terms of a polynomial representation (Bulthoff et al., 1980). Such studies may in turn, help to guide and interprete experiments aiming to understand information processing in the nervous system in addition to its own interest, understanding at the level of the algorithm may help significantly to unravel the underlying hardware. According to a recent conjecture (Torre and Poggio, 1978; Poggio and Torre, 1978) information processing in the nervous system is performed in part by local circuits, that is by local synaptic interactions between graded potentials. Since these interactions can directly compute polynomial functionals of the inputs, the link between polynomial algorithms and nervous hardware may well turn out in some instances to be rather direct.

Acknowledgements. We thank J. Cowan and G. Palm for several discussions, helpful criticism and for reading various versions of the manuscript. We are grateful to I. Geiss for typing it and to L. Heimburger for drawing the figures.

References

Abelson, H.: Computational geometry of linear threshold functions. Inf. Control **34**, 66–92 (1977)

Abelson, H.: Towards a theory of local and global in computation. Theor. Comput. Sci. **6**, 41–67 (1978)

d'Alessandro, P., Isidori, A., Ruberti, A.: Realization and structure theory of bilinear dynamical systems. SIAM J. Control **12**, 980–996 (1974)

Arnold, V.I.: Representation of continuous functions of three variables by the superposition of continuous functions of two variables. Am. Math. Soc. Trans., Ser. **2**, 28 (1963)

Barrett, J.F.: The use of functionals in the analysis of nonlinear systems. J. Electron. Control **15**, 567–615 (1963)

Barrett, J.F.: The series reversion method for solving forced nonlinear differential equations. Mathematica Balkanica **4.5**, 43–60 (1974)

Bedrosian, E., Rice, S.O.: The output properties of Volterra systems. Proc. IEEE **59**, 1688–1707 (1971)

Berger, M.S.: Nonlinearity and functional analysis. Lectures on nonlinear problems in mathematical analysis. New York, San Francisco, London: Academic Press 1977

Brent, R.P., Kung, H.T.: Fast algorithms for manipulating formal power series. J. ACM **25**, 581–595 (1978)

Brilliant, M.B.: Theory of the analysis of nonlinear systems. Tech. Rep. 345 Res. Lab. El. M.I.T. (1959)

Bülthoff, H., Poggio, T., Reichardt, W.: Figure-ground discrimination by relative movement in the visual system of the fly. Part II. Movement and relative movement algorithms. To be submitted to Biol. Cybernetics (1980)

Coleman, B.D.: Finitely convergent learning programs for the separation of sets of shaded figures. Biol. Cybernetics **25**, 57–60 (1976)

Coleman, B.D., Mizel, V.J.: Generalization of the perceptron convergence theorem. Brain Theory Newsletter **1**, 67–68 (1976)
DeSantis, R.M., Porter, W.A.: On the analysis of feedback systems with a polynomic plant. Int. J. Control **21**, 159–175 (1975)
Halme, A.: Polynomial operators for nonlinear systems analysis. Dissertation, Helsinki 1972. Acta Polytechnica Scand. M24 (1972)
Halme, A., Orava, J., Blomberg, H.: Polynomial operators in nonlinear systems theory. Int. J. Syst. Sci. **2**, 25–47 (1971)
Harper, T.R., Rugh, W.J.: Structural features of factorable Volterra systems. IEEE Trans. Autom. Control **17**, 226–228 (1976)
Holtzman, J.M.: Nonlinear system theory. A functional analysis approach. Englewood Cliffs, N.J.: Prentice-Hall 1970
Julesz, B.: Experiments in the visual perception of texture. Sci. Am. **232**, 34–44 (1975)
Kohonen, T.: Associative memory. A system-theoretical approach. Berlin, Heidelberg, New York: Springer 1977
Kolmogorov, A.N.: On the representation of continuous functions of several variables by superpositions of continuous functions of one variable and addition (in russian). Dokl. Akad. Nauk SSSR **114**, 679–681 (1957); Am. Math. Soc. Transl. **28**, 55–59 (1963)
Marmarelis, P.Z., Marmarelis, V.Z.: Analysis of physiological systems. New York: Plenum Press 1978
McCulloch, W.S., Pitts, W.H.: A logical calculus of ideas immanent in nervous activity. Bull. Math. Biophys. **5**, 115–133 (1943); New York, London: Plenum Press 1978
Minsky, M., Papert, S.: Perceptrons. An introduction to computational geometry. Cambridge, Mass., London: MIT Press 1969
Palm, G.: On representation and approximation of nonlinear systems. Biol. Cybernetics **31**, 119–124 (1978)
Palm, G.: On representation and approximation of nonlinear systems. Part II. Biol. Cybernetics **34**, 49–52 (1979)
Palm, G., Poggio, T.: The Volterra representation and the Wiener expansion: validity and pitfalls. SIAM J. Appl. Math. **33**, 195–216 (1977a)
Palm, G., Poggio, T.: Wiener-like system identification in physiology. J. Math. Biol. **4**, 375–381 (1977b)
Palm, G., Poggio, T.: Stochastic identification methods for nonlinear systems: an extension of the Wiener theory. SIAM J. Appl. Math. **34**, 524–534 (1978)
Palm, G., Poggio, T.: Discrete system theory (in preparation 1980)
Poggio, T.: On optimal nonlinear associative recall. Biol. Cybernetics **19**, 201–209 (1975a)
Poggio, T.: On optimal discrete estimation. In: Proceedings of the 1st Symp. on Testing and Identification of Nonlinear Systems, March 17–20, 1975. McCann, G.D., Marmarelis, P.Z. (eds.), pp. 30–37. California Inst. of Technology 1975b
Poggio, T., Reichardt, W.: Visual control of orientation behaviour in the fly. Q. Rev. Biophys. **9**, 377–438 (1976)
Poggio, T., Reichardt, W.: Characterization of nonlinear interactions in the fly's visual system. In: Theoretical developments in neurobiology. Neurosciences research program bulletin. Cambridge, Mass., London: MIT Press 1980 (in press)
Poggio, T., Torre, V.: A Volterra representation for some neuron models. Biol. Cybernetics **27**, 113–124 (1977)
Poggio, T., Torre, V.: A new approach to synaptic interactions. In: Theoretical approaches to complex systems. Proceedings, Tübingen, June 11–12, 1977. Heim, R., Palm, G. (eds.), pp. 89–115 (1978)
Prenter, P.H.: A Weierstrass theorem for real separable Hilbert spaces. J. Approx. Theory **3**, 341–351 (1970)
Prenter, P.M.: On polynomial operators and equations. In: Nonlinear functional analysis and applications, pp. 361–398. New York, London: Academic Press 1971
Reichardt, W., Poggio, T.: Figure-ground discrimination by relative movement in the visual system of the fly. Part I. Experimental results. Biol. Cybernetics **35**, 81–100 (1979)
Torre, V., Poggio, T.: A synaptic mechanism possibly underlying directional selectivity to motion. Proc. R. Soc. London, Ser. B **202**, 409–416 (1978)
Uesaka, Y.: Analog perceptrons: on additive representation of functions. Inf. Control **19**, 41–65 (1971)
Uesaka, Y.: Analog perceptron: its decomposition and order. Inf. Control **27**, 199–217 (1975)

Received: March 11, 1980

Dr. T. Poggio
Max-Planck-Institut
für biologische Kybernetik
Spemannstrasse 38
D-7400 Tübingen
Federal Republic of Germany

Verantwortlich für den Textteil: Prof. Dr. W. Reichardt, Max-Planck-Institut für biologische Kybernetik, Spemannstr. 38, D-7400 Tübingen. Verantwortlich für den Anzeigenteil: L. Siegel, G. Sternberg, Springer-Verlag, Kurfürstendamm 237, D-1000 Berlin 15, Fernsprecher: (030)8821031, Telex: 01-85411. Springer-Verlag, Berlin·Heidelberg·New York.
Druck der Brühlschen Universitätsdruckerei, Gießen. Printed in Germany.