Building and Understanding Adaptive Systems: A Statistical/Numerical Approach to Factory Automation and Brain Research

PAUL J. WERBOS

Abstract—**Successes with expert systems and other specialized systems have revived hopes for factory automation and productivity growth. A full realization of these potentials will require conscious effort to overcome obsolete rigidities, to develop unified and adaptive methods for integrating complex systems, and to increase our understanding of these systems (understanding which is vital to human productivity in developing software). How adaptive systems may be built and understood by extending control theory and statistics is discussed. Adaptive systems, like human infants, are less agile than young monkeys but have something important to contribute as they mature. It argues that the old dream of understanding intelligence in generalized terms, permitting a unified understanding of adaptive systems and of the human mind, was not incorrect; rather, the early attempts in that direction failed because they did not make full use of research possibilities in statistics, control theory, and numerical analysis (many of which are still unexploited) and were limited by hardware costs which are now coming down. A basic adaptive system derived from this approach fits the fundamental, qualitative empirical facts of human brain physiology in some detail (unlike the usual "general neuron models," which rarely even discriminate between basic components of the brain), and offers opportunities for further research; it can even translate certain fundamental ideas of Freud into something more mathematical and scientific. The mathematics of the basic system, and the fit to the brain, are described in detail.**

I. Alternative Routes to Automation

EARLY ATTEMPTS at factory automation were mainly based on mechanical engineering. Machines were designed to perform narrowly defined repetitive tasks. New products could be built only by giving humans instructions to use the machines differently or by scrapping or retooling equipment at great cost.

The spread of microchips has revived hopes for flexible manufacturing and untended assembly lines. Even though computer control now adds substantially to the cost of a machine and the software is at an early stage of development, 35–40 percent of the demand for machine tools is already going to computer-controlled varieties [1].

Computer control opens up a wide range of possibilities for increasing productivity. It will be a difficult challenge, however, to take full advantage of these possibilities. Conscious effort will be needed to meet three more specific challenges:

- to overcome rigid and mechanistic approaches to process control, inherited from the past;
- to integrate a variety of techniques on the factory floor, so as to get the full benefit of diverse, flexible technologies without causing chaos, bad fits, and unpredictable operations; and
- to maintain intellectual coherence to maximize flexibility and minimize confusion in the human research community that supports factory automation.

Because of the vital importance of raising productivity, many government programs have been proposed in the United States and elsewhere to accelerate automation. It is interesting to compare these proposals with the three specific challenges above.

NASA has proposed an advanced automation program [2], which would include a major effort on automated extraterrestrial materials processing. This effort would aim at a miniature but complete integrated manufacturing system, going all the way from dirt to usable products, operating in a new environment, and drawing on many subcontractors. It would certainly force a rethinking of manufacturing approaches and a high degree of integration. As with NASA's prior work developing microchips in the first place, one could expect major benefits to the private sector. NASA's emphasis on developing basic science is a major factor in this proposal; however, it is unclear how much funding and support there will be for it.

The Department of Defense (DOD) is said to be pursuing a large program of numerous independent seed grants to defense contractors to encourage automation; perhaps the phrase "seed grants" conjures up unfair analogies to urban and small business seed grants, whose effectiveness has been questioned, but it is still hard to imagine this approach helping very much with the three challenges listed above. DOD research has traditionally stressed quick and secret results, in a way which often leads to a profusion of *ad hoc* approaches; there have been a few important exceptions to this, however. Some scientists have suggested that the Strategic Defense Initiative, by stressing important long-term research and possibly encouraging

Manuscript received December 4, 1985; revised May 6, 1986 and July 12, 1986.

The author is with the Energy Information Administration, EI-621, Department of Energy, Washington DC 20585, USA.

IEEE Log Number 8610540.

0018-9472/87/0100-0007$01.00

international information sharing, may yield much larger contributions to civilian science than earlier DOD work has; however, many decisions have yet to be made about the level and types of open research in that program as it evolves.

The Japanese "Fifth Generation" program appears to be somewhere between the NASA and DOD approaches. The National Bureau of Standards [1] is conducting important research in-house to develop specific tools that should help integrate existing methods and contribute heavily to the near-term needs of industry.

From a technical point of view, the new kinds of software needed for factory automation can be divided crudely into three groups (or aligned along three dimensions):

- *fixed systems:* tools which direct machines to perform specific tasks, based on a straightforward rigid relation between sensor inputs and machine actions;
- *expert systems:* tools which facilitate the transfer of specialized information (mostly about how to perform a task) from humans and laboratories to factories or other users; (These tools often contain specialized routines for formal symbolic reasoning to help in extracting this information in a more useful form [3].)
- *adaptive systems:* tools which develop new information about how to do the task better by analyzing past experience and relating it to performance criteria set by humans.

All three types will play an essential role in the factory of the future. In general, however, as new tasks have grown more complex and difficult, there has been a movement up the scale, from purely fixed systems, to semi-expert systems, on up to systems with an adaptive component.

At present, adaptive systems are used mostly in laboratories, by human beings trying to acquire information (e.g., by statistical analysis) which they later use in designing fixed systems. However, adaptation within the factory itself is becoming increasingly important. Trial-and-error systems, pattern recognition systems, and even heuristic programming are now part of the mainstream of the machine tool industry [1]; statistical quality control is also becoming essential to industries' competitiveness. As the complexity of production systems grows, it will be more and more important to develop adaptive controllers capable of managing this complexity on a real-time basis.

II. Goals of a Statistical / Numerical Approach

The remainder of this paper will describe a unified statistical/numerical approach to building and understanding adaptive systems. A relatively small set of generalized, basic tools may permit us to meet the challenges of integration and coherence and to accelerate the growth in the knowledge base which underlies automation. This does not eliminate the need to take advantage of special-purpose tools or expert systems, where available, but it provides a general framework in which special-purpose tools may be inserted, enhanced, and coordinated. The goal here is to build or understand adaptive systems which can integrate information from many sources in order to *optimize overt action*; in this approach, symbolic reasoning may be treated as part of overt action but not as a universe unto itself.

To build such a general framework, this paper proposes that we keep in mind the *overall* problem of building a system to handle all of the calculations from crude inputs through to overt actions in an adaptive way, so as to maximize some measure of overall performance *over time*. Specific tools for automation can then be treated as subsystems of the more general system and evaluated on the basis of cost and performance just as we evaluate the choice of picture tubes within a television. (In some cases, we would use a nonadaptive component to save money, to condense the inputs or outputs from an adaptive component, or to provide initial conditions for an adaptive component.) General systems of this sort may be called *intelligent systems* (with apologies to those who have used this term in other ways).

This definition of intelligent systems is simply an extension of well-known approaches in operations research. For decades, system design problems have been broken into two main pieces: 1) modeling, which attempts to quantify the objective aspects of each problem; and 2) optimization, which calculates settings for the action (or control) variables so as to maximize the expected value of some utility function provided from the outside, assuming a model of the problem. Likewise, decision tree analysis [4] (which maximizes a utility function over time) has been widely used by corporate planners. Economists have suggested that factories should be built to maximize the flow of profits over time. In summary, the goal of maximization over time should be sufficient to represent the goals of factory automation.

A further goal of this approach is to help in the understanding of adaptive systems. This ties in with the challenge of maintaining intellectual coherence, which is important now that human productivity in developing software is a major economic issue. It also ties in with the more fundamental goal of understanding the human brain. In the last few decades, several attempts have been made to unify the complex world of brain studies by the use of "general neuron models," which were intended to explain the adaptiveness of the human brain. This paper will explore the possibility that the design requirements of intelligent systems may force a more restricted class of brain models and may therefore have more explanatory power; it will discuss preliminary results and possibilities for further research along these lines. The pursuit of such research may be useful to the field of automation, because the brain is the only working example we have to prove that general adaptive intelligence is possible. The problems we encounter in explaining the brain may turn into opportunities to build more powerful adaptive systems.

The next section of this paper will describe a generalized approach to building intelligent systems by linking three

basic components—a longer term optimization system (J), a modeling system ($\boldsymbol{f}$), and a short-term optimization system ($\boldsymbol{u}$). Sections IV–VI will discuss the mathematics of specific possibilities for filling in these three components; these possibilities will suggest additional research, above and beyond what is being done at present, based on the traditional fields of statistics, control theory, and numerical analysis. The final sections will compare this approach with traditional artificial intelligence, and discuss the possible applications to brain research.

III. A Statement of the Approach in General Terms

An intelligent system has been defined as a system which solves the problem of maximizing a measure of success *in the long-term*. This is a very difficult problem, and a special kind of design is needed to solve this problem in the general case. This section will discuss the kind of design which is required—a threefold design, made up of components symbolized by the letters J, $\boldsymbol{f}$, and $\boldsymbol{u}$. Mathematical tools already exist to fill in these three components; therefore, a primitive type of intelligent system could already be built today. However, this paper will suggest further research to enhance each of the three components and link them together more effectively.

Before discussing the problem of maximizing *over time*, it may help to review the problem of maximizing a function. Many software packages already let you maximize any function you can write down. To use such a system you have to do the following three things.

1) Type in the formula for whatever you want the system to maximize. (This is usually called U, which stands for "utility function." For example, U might represent profits.)
2) Tell the system what values to assume, at first, for all the variables in your formula.
3) Tell the system which variables it should try to adjust to maximize U. (This set of variables is called the "control" or "action" variables; it is usually symbolized as $\boldsymbol{u}$.)

The problem of maximizing something over time is far more difficult. There is usually no simple formula which spells out explicitly how the action variables ($\boldsymbol{u}$) relate to your ultimate long-term bottom line. Instead, you usually have a model $\boldsymbol{f}$, which predicts how your present actions ($\boldsymbol{u}$) will change the overall state of reality ($\boldsymbol{R}$) in the immediate future. If you do not have such a model, if you cannot predict the results of your actions at all, then you have no hope of picking out the right actions. (Actually, there once were a few computer scientists who hoped you could use the world itself as your only model; the failure of that approach may be explained by the numerical inefficiency which results from not being able to carry out the calculations to be described.) Even when you have such a model, there is a big difference between knowing the results you would get in the immediate future and knowing the long term results. As a practical matter, we know that human beings can figure out how to survive in the long term, beyond a few instants of time; therefore, we know that foresight is a basic part of human intelligence.

Many maximization methods exist for simple problems, where your model $\boldsymbol{f}$ has a simple, specialized form. However, there is only one family of methods which gives the right answer over time for any model f, even in cases where you cannot predict the future with certainty: dynamic programming.

In dynamic programming, we start out by assuming that the state of reality at any time t can be fully described by a set of numbers $\boldsymbol{R}(t)$. We usually think about reality as a series of snapshots, such that $\boldsymbol{R}(t+1)$ would represent reality in the "next" period of time. To use dynamic programming, you have to give the computer two major pieces of information:

- a utility function U, which represents your long-term basic objectives; (In chess, for example, U might be $+1$ for winning, -1 for losing, and 0 for anything else.)
- a dynamic model $\boldsymbol{f}$, which predicts reality at time $t+1$ as a function of reality at time t, actions at time t, and random factors ("noise"). (The random factors represent uncertainty.) There are many ways to specify such a model; for example, for each variable in $\boldsymbol{R}$, you could type in a formula which predicts that variable at time $t+1$ as a function of earlier information.

Dynamic programming then translates this long-term maximization problem into a short-term maximization problem, which is far more tractable. It does this by calculating a secondary or strategic utility function J, which has the following property: any operational system that picks its actions $\boldsymbol{u}(t)$ so as to maximize J in the immediate future will *automatically* do a perfect job of maximizing U in the long-term future. In chess, for example, the old rule of counting nine points for a queen, five for a castle, etc., is nothing but an attempt to approximate J for that game; by fighting for points in the short term, players hope to win the game in the long term. Because this rule gives an imperfect approximation to J, it sometimes makes sense to sacrifice points in the short-term to achieve other long-term strategic goals. However, a better approximation to J should account for these other strategic goals as well. Decision-tree analysis is one form of dynamic programming.

Conventional dynamic programming cannot be applied to problems with many variables, because the calculations are impossibly expensive. Therefore, intelligent systems must be based on generalized methods to approximate the results of dynamic programming.

Dynamic programming is expensive because it tries to estimate the value of J explicitly in every possible state of reality; we can approximate this by trying to find formulas, which approximate the values of J and $\boldsymbol{u}$ as a function of reality, just as we try to find models $\boldsymbol{f}$ which

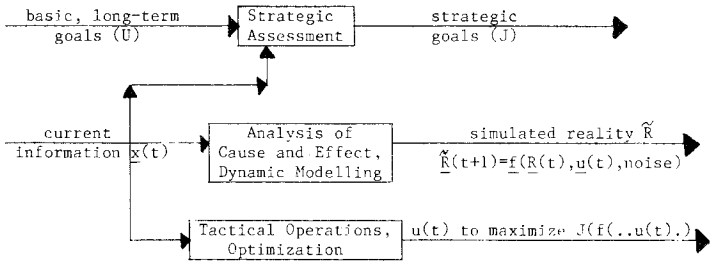


Fig. 1. Three core components of any intelligent system (J, $\boldsymbol{f}$, $\boldsymbol{u}$).

approximate the dynamics of the real world in a general way. We can try to develop methods to adjust the parameters in these formulas, and evaluate how good the formulas are on the whole, by methods just like the methods statisticians already use to adjust the parameters in dynamic models and evaluate their quality of fit. Section IV will describe one possible method called global dual heuristic programming (GDHP).

Following these ideas, a primitive intelligent system can be built simply by plugging together *generalized* routines to perform each of the three key functions shown in Fig. 1. For example, use GDHP for strategic assessment, use existing statistical routines (like nonlinear regression) for dynamic modelling, and use function maximization for the final decisions on actions to take. This leads to concrete possibilities for user-oriented software packages. These possibilities are described in the course of a more intuitive discussion [5] of the material to be given below in technical terms.

IV. GDHP: A Possible System for Estimating / Adapting J and $\boldsymbol{u}$

Howard has developed a version of dynamic programming which can be applied to decision problems which extend out into the indefinite future [6]. In Howard's version, J is derived by solving an equation which is slightly simplified as

$$J(\boldsymbol{R}) = U(\boldsymbol{R}) - \bar{U} + \max_{\boldsymbol{u}} [J(\boldsymbol{f}(\boldsymbol{R}, e, \boldsymbol{u}))] \quad (1)$$

where J, $\boldsymbol{R}$, $\boldsymbol{u}$, U, and $\boldsymbol{f}$ are all defined in Section III and $\bar{U}$ is a constant required to permit a global solution of the equation. Square brackets denote expectation values. Howard has also shown that J and $\boldsymbol{u}$ can be obtained from a process of successive approximations. This process involves the alternation of two steps: 1) upgrade the estimate of J on the left hand side of (1), using the present strategy $\boldsymbol{u}(\boldsymbol{R})$ and the present estimate of J on the right; 2) upgrade $\boldsymbol{u}(\boldsymbol{R})$ to maximize the new estimate of J. Some models $\boldsymbol{f}$ exist for which $\bar{U}$ is a function of one's starting state. However, these are not of great concern in automation or in brain research [7].

To solve (1) exactly in the general case, one needs to consider arbitrary functions $J(\boldsymbol{R})$, which leads to difficulties already discussed. To find the "best" approximation J^* of a specified functional form, one has to define the word "best"; in other words, one has to specify a measure of error (E) which represents the degree to which J^* does not fit (1). One possibility, called heuristic dynamic programming (HDP), is to minimize the square error of (1) itself over some set of simulated values of $\boldsymbol{R}$ and $\boldsymbol{e}$:

$$E_0 = \left[\left(J(\boldsymbol{R}) - U(\boldsymbol{R}) + \bar{U} - J(\boldsymbol{f}(\boldsymbol{R}, e, \boldsymbol{u}(\boldsymbol{R})))\right)^2\right]. \quad (2)$$

(In (2) and in what follows, the asterisk after J will be dropped; only the approximation to J is available to calculation.)

GDHP is based on differentiating (1) with respect to each component R_i of the reality vector $\boldsymbol{R}$ and trying to make the resulting equations fit as well as possible. This results in

$$E = \sum_i W_i \left[\left(\frac{\partial}{\partial R_i}(J(\boldsymbol{R}) - U(\boldsymbol{R}) - J(\boldsymbol{f}(\boldsymbol{R}, e, \boldsymbol{u}(\boldsymbol{R}))))\right)^2\right] \quad (3)$$

where $\boldsymbol{W}$ is a vector of weights yet to be specified. GDHP appears more promising than HDP for several reasons.

- The derivatives of J with respect to $\boldsymbol{R}$ are the "shadow prices" or Lagrange multipliers used in economics [8, p. 73] and [5].
- The purpose of this exercise is to make decisions ($\boldsymbol{u}$) as accurately as possible. This suggests an effort to minimize errors in the *derivatives* of J, rather than in J itself, because the decisions are generally based on a local comparison whose validity depends on the derivatives of J. (More precisely, the choice of $\boldsymbol{u}$ to maximize $[J(\boldsymbol{F}(\boldsymbol{R}, \boldsymbol{u}))]$ depends on comparisons between situations ($\boldsymbol{F}(\boldsymbol{R}, \boldsymbol{u})$) which are close to each other, because it is rare for large changes to occur in one instant of time; estimated differences in J between nearby points are determined by the derivatives.)
- Howard's $\bar{U}$ constant is no longer needed. This eliminates what could be a very troublesome problem for approximation methods.
- Intuitively, GDHP tends to weight the importance of a given variable (i.e., the derivative of J with respect to a variable R_i) based upon the *causal impact* of R_i on $\boldsymbol{R}(t+1)$. (See the discussion just after (7) below.) Given that HDP does not account for this information

about causality, one would expect GDHP to be more efficient numerically.

The use of GDHP involves certain theoretical difficulties, such as the need to choose a functional form J, the problem of choosing weights $\boldsymbol{W}$, dangers of autocorrelation leading to a poor approximation, and the issues of robustness and unobserved variables. All of these difficulties are exactly parallel to well-known difficulties with the conventional least-squares methods of statistical estimation; the same kinds of solutions and research can be carried over, and one may hope that reality vectors $\boldsymbol{R}$ developed by refined statistical analysis may reduce these problems directly. (See Sections V and VII.) For purposes of initial research, it may be adequate to set each W_i to the reciprocal of the standard deviation of R_i. A variety of numerical examples could be used to evaluate the relative strengths and weaknesses of these methods. Likewise, a variety of strategies may be considered for simulating $\boldsymbol{R}$ and $\boldsymbol{e}$, starting with the obvious strategy of choosing historical or current values for $\boldsymbol{R}$ and Monte Carlo simulation for $\boldsymbol{e}$.

Both GDHP *and* the conventional methods of statistics have serious numerical problems in adapting large-scale networks (systems of equations). For example, conventional packages to perform nonlinear regression tend to use a lot of computer time, and some of them will fail to converge at times when there are dozens of parameters to be estimated. Nevertheless, the Department of Energy has proven the feasibility of estimating many thousands of parameters simultaneously and has suggested specific possibilities for further numerical research [9]. (Whatever their other problems, real-time systems can usually afford to go through many iterations before finding an optimum; instead of doing a complete data search in each iteration of the estimation process, they can estimate or adapt their parameters as they go through the data.)

Crucial to the success of these adaptation/estimation methods is the availability of *derivatives*; if we know the derivative of error with respect to every parameter in a network, we can always adjust each parameter in the right direction. Previous papers have described the general principle of *dynamic* feedback, a numerical method which can calculate all the derivatives of a network for a cost comparable to that of exercising the network one time [10]. This principle can be applied to calculate the derivatives of E in (3) with respect to all the parameters in the J and $\boldsymbol{u}$ networks. This is crucial to the actual implementation of GDHP in the general nonlinear case.

The details of the derivative calculation will be given in the Appendix. For now, it is enough to spell out the basic concepts. Let $J(\boldsymbol{R})$ be expressed more explicitly as $J(\boldsymbol{R}, \boldsymbol{a})$, where $\boldsymbol{a}$ represents the parameters to be estimated in the system of equations that approximate J. Likewise, write $\boldsymbol{u}(\boldsymbol{R}, \boldsymbol{b})$, and define

$$\boldsymbol{F}(\boldsymbol{R}, \boldsymbol{e}, \boldsymbol{b}) = \boldsymbol{f}(\boldsymbol{R}, \boldsymbol{e}, \boldsymbol{u}(\boldsymbol{R}, \boldsymbol{b})). \tag{4}$$

Finally, define $U_i(\boldsymbol{R})$ as the derivative of $U(\boldsymbol{R})$ with respect to R_i. In this notation, (3) may be expressed as

$$E = \sum_i W_i \left[\left(-U_i + \frac{\partial}{\partial R_i} (J(\boldsymbol{R}, \boldsymbol{a}) - J(\boldsymbol{F}(\boldsymbol{R}, \boldsymbol{e}, \boldsymbol{b}), \boldsymbol{a})) \right)^2 \right]. \tag{5}$$

To minimize E as a function of the parameters $\boldsymbol{a}$, one would normally calculate the derivative of E with respect to each parameter a_k and adjust each parameter as suggested by its derivative [9]. However, this does not correspond exactly to Howard's original method of successive approximations. In Howard's method, one adjusts J on the left-hand side of (1) while holding fixed the prior estimate of J on the right-hand side. To mimic this procedure, we can adjust the parameters a_k based on the derivatives of E with respect to a_k calculated as if the $\boldsymbol{a}$ in $J(\boldsymbol{F}(\boldsymbol{R}, \boldsymbol{e}, \boldsymbol{b}), \boldsymbol{a})$ were constant:

$$E'_a(k) = \left[\sum_i 2 W_i d_i \frac{\partial^2}{\partial a_k \partial R_i} J(\boldsymbol{R}, \boldsymbol{a}) \right] \tag{6}$$

where

$$d_i = \frac{\partial}{\partial R_i} (J(\boldsymbol{R}, \boldsymbol{a}) - J(\boldsymbol{F}(\boldsymbol{R}, \boldsymbol{e}, \boldsymbol{b}), \boldsymbol{a})) - U_i(\boldsymbol{R}). \tag{7}$$

Note that there is no need to know the utility function U itself; it is enough that the U_i be available. Likewise, note that the calculation of d_i is the place where the "causal effect" of R_i on $J(\boldsymbol{R}(t+1))$ is accounted for explicitly.

The Appendix will spell out the calculations needed to get $E'_a(k)$ for all the parameters a_k, which is used as the basis for adapting each parameter. It will also spell out the calculations needed to get the derivatives of $J(\boldsymbol{F}(\boldsymbol{R}, \boldsymbol{e}, \boldsymbol{b}), \boldsymbol{a})$ with respect to all of the b_k parameters, to be used in adapting the b_k parameters concurrently (in imitation of Howard's method of successive approximations). This amounts to a complete recipe for how to adapt all the parameters in the system, *except* for the parameters of the dynamic model $\boldsymbol{f}$; these parameters will be discussed in the next section.

V. Beyond Regression: Research Requirements with Dynamic Models

Conventional textbooks already provide a broad set of techniques for estimating dynamic models ($\boldsymbol{f}$). One can fill in the $\boldsymbol{f}$ component of a basic intelligent system simply by programming the conventional techniques. To adapt the parameters of a model, one need only minimize the sum of squared error, weighting the variables as specified by maximum likelihood theory. To improve the functional form, one can use conventional t statistics to decide which terms to delete and a random or exhaustive search to locate new terms to try (as in stepwise regression). If there is a danger of autocorrelation or the like, one can simply expand the set of functional forms (terms to be tried) to account for this possibility. These conventional methods,

and the theory behind them, have been discussed at length in many publications [11]. They have proven their value in a wide range of applications, including artificial intelligence.

In practice, the conventional methods do have limitations, which are important in some applications. In order to build a more powerful intelligent system, one would want to enhance the $\boldsymbol{f}$ component by including capabilities to overcome these limitations automatically whenever they appear. This points to a need for further research in five major areas which are explained in detail in the following citations:

- convergence with very large nonlinear models (networks) in real time; (This will require the use of differentiation methods and solution methods discussed in Section IV.)
- the use of "screening" methods [12] to improve on the random insertion of new terms into a model; (Such methods were once too subtle for statisticians to grasp, but recent research in related areas [13], [14] has suggested possibilities such as the invocation of log-normal association-based prior probabilities (with approximations to hold down the cost).)
- methods to estimate a reality vector $\boldsymbol{R}$ which may differ from the vector of crude data inputs $\boldsymbol{x}$ and allow a better predictive model; (This is related to the assumption in dynamic programming that the model $\boldsymbol{f}$ does not use information about $\boldsymbol{R}$ from before time t; in effect, $\boldsymbol{R}$ needs to account for the lagged information of importance to good forecasts.) This might be done through a workable approximation to nonlinear filtering (as in "pattern analysis" [15]) or through a generalization of methods by Box and Jenkins which entail less of an identification problem in the linear case [11]. Valid simulations also require an effort to estimate $\boldsymbol{R}$ in such a way that the errors in $\boldsymbol{f}$ are uncorrelated across different components of $\boldsymbol{R}$.
- the problem of cumulative error, which has turned out to be very important in real-world forecasting; (Estimation methods have been developed, which minimize this problem through "dynamic robust estimation" [16], [17]. In the context of intelligent systems, where $\boldsymbol{R}$ may contain filtered variables and where the derivatives of J with respect to these variables are available (see the Appendix), these methods may be extended simply by weighting the error in forecasting all the variables as a function of their J-derivatives. More research is needed to pin down the possibilities and the underlying theory.)
- the integration of "precedent-based forecasting" or of "nearest neighbor methods" [18] into model-based forecasting systems (e.g., through "syncretism" [15] in some form).

These research problems or opportunities have mostly arisen in practical applications of statistical forecasting in economics and social science. In those fields, specialized solutions are usually much easier than general solutions [11]. To arrive at general solutions, it may be necessary for someone (the statistician? the computer scientist?) to carry out more research aimed at the large-network problem for its own sake, possibly in a theoretical context, or possibly drawing on simulation studies and examples from a variety of fields; in any event, a better understanding of these problems will continue to require a heavy reliance on basic statistical theory to unify our understanding of applications from a wide variety of fields.

VI. Beyond GDHP: Research Possibilites for J and $\boldsymbol{u}$

GDHP, like conventional statistics, may be sufficient by itself for certain applications. A basic intelligent system may be built using only GDHP to adapt the J and $\boldsymbol{u}$ networks. However, GDHP also has two likely limitations, related to the tricky operational problem of choosing a cycle time (the gap between t and $t+1$). Research is needed to assess these limitations, and to assess the value of methods to overcome them.

The first of these limitations concerns the $\boldsymbol{u}$ network. The cycle time of an intelligent system should be set large enough to permit a complex, thorough strategic assessment (i.e., a multilayer J network); in the human brain, for example, the alpha rhythm has a cycle time of about a tenth of a second, versus a neuron-to-neuron response time of only a few milliseconds. However, a cycle time as long as a tenth of a second may lead to jerky, uncoordinated actions which do not reflect the possibility of real-time control down to the level of milliseconds or less. In order to *combine* a complex strategic analysis (J) with *fine-tuning* of the ultimate actions, one may interpose another level of optimization between the high-level choice of actions ($\boldsymbol{u}(t)$) and the actual actions. More precisely, one might develop a way of interfacing the J network with a subordinate optimization system which runs on a much smaller cycle time but need not provide a profound degree of foresight. To do this, one might insert a subordinate optimization system based on GDHP and set it to use the derivatives of the primary J (as described in the Appendix) as its source of utility derivatives (U_i), and to perform only one or two layers of calculation. Alternatively, one might insert a system based on an older method, dual heuristic programming (DHP) [15], with the following characteristics:

- instead of J, the network forecasts the vector of derivatives of J with respect to $\boldsymbol{R}$. This vector may be denoted $\boldsymbol{L}$;
- the same error derivatives are used as in (6) and (7), except that references to the derivatives of J are replaced by references to $\boldsymbol{L}$.

DHP has less statistical/numerical efficiency than GDHP; for example, if J is a quadratic function of $\boldsymbol{R}$, DHP wastes data rediscovering the fact that the underlying matrix is

symmetric. Nevertheless, for this application speed may be more important than efficiency.

GDHP may or may not have a further limitation, related to long-term foresight. This possibility is related to a limitation of Howard's dynamic programming, which GDHP is based on.

In Howard's method, we must start the process of successive approximations with some initial guess for the function J. Typically, we would use U. Using this first iteration guess for J, one picks actions $\boldsymbol{u}(t)$ so as to maximize $U(t+1)$. Then, after one invocation of (1), the new version of J tends to represent $U(t) + U(t+1)$. After two iterations, it tends to represent $U(t) + U(t+1) + U(t+2)$. In general, after n iterations, the system accounts for the future n cycle times ahead in time. If the cycle time were a tenth of a second, and the time for a complete iteration (accounting for a wide range of possible situations) were 5 s, then the effective foresight horizon would always be two percent of the lifetime of the system. Yet humans seem to do much better than this, and it may be desirable to make sure that an intelligent system can also do better if this is possible.

In actuality, GDHP may do better than Howard's method here. Current methods for estimating parameters [9] do notice situations where a parameter keeps getting adjusted by a small amount in the same direction over many iterations. They generate much larger adjustments in these situations, so that many cycle times may be anticipated in a single iteration. Nevertheless, this suggests that there is a very difficult trade-off between effective foresight and stability in using these estimation methods with GDHP. Further research is needed, such as research on simulated or exact examples based on cases where the underlying J functions take on a vector–polynomial form or the like.

Perhaps there is no neat way to overcome this limitation. The brain itself may represent a working tradeoff between stability and effective foresight, rather than a sweeping solution to the dilemma. One might even argue that humans are the only animals that appear to show a high degree of effective foresight and that we arrive at this appearance *after learning* how to use symbolic reasoning to identify concepts (R_i variables), which GDHP finds it easy to digest. In building factories, one might even prefer to limit the effective foresight; with a high degree of foresight (plus the resulting instability), intelligent systems might sometimes become neurotic, manipulative, or even rebellious.

Nevertheless, there may be a way to improve on the effective foresight of GDHP. For example, one might build a system with two subsystems, one to approximate J and another to approximate $\boldsymbol{L}$, the vector of derivatives of J with respect to $\boldsymbol{R}$. The $\boldsymbol{L}$ network could be adapted to approximate the actual derivatives of the J network as closely as possible; at the same time, the J network might use $\boldsymbol{R}$ and $\boldsymbol{L}$ both as its inputs. This might allow a faster means/ends association, to permit de facto the kind of information flow called for in the literature on "plans" and "schemata" in the psychology of problem solving and education.

VII. Comparison with Traditional Artificial Intelligence

The approach here is intended to complement the work on expert systems and fixed systems, not compete with it, as discussed above; still, it does suggest the need to broaden the work in expert systems to convey probabilities and quantitative information (as required in production optimization) to a greater extent. Also, there is a possibility that the concepts discussed here may be useful in designing adaptive "evaluators" for use in expert systems (see [3, sec. 4.5]).

The proposal here is to represent the three basic subsystems as generalized "networks" of elementary calculations or processors, arranged in whatever combination works best; in other words, the structure should evolve through learning (still taking advantage of initial guesses) rather than forced to fit abstract Aristotelian concepts of relational knowledge structures. Human beings use abstract knowledge structures much as we use integral calculus; we treat them as important information, but they are not part of the "deep structure" hard-wired into the brain. Prof. Arbib of Amherst has recently co-edited a book [19] on neurolinguistics, which contains evidence that humans understand language by a process much closer to nonlinear filtering of time-series input than to the formal grammars described by Chomsky and others. Recent research on neurolinguistics has cast doubt on the idea that language is a "hard-wired" function [20]. Though human children are motivated by an early interest in sounds and people, there is now no reason to believe that linguistic symbols are learned by different rules at the cellular level after the brain has been motivated to learn them.

The approach in this paper is very similar in spirit to the early work on "perceptrons" [21], [22] and to Minsky's early attempt to develop artificial intelligence through function minimization by trial and error [23]. That early work reached a point of diminishing returns years ago, as it was found that good guessing was not quite good enough to figure out the equations for a powerful adaptive system. In reaction to this, the work of the 1970's emphasized the development of specialized programs to solve specialized problems, such as locating objects in a visual grid [24], language manipulation, and expert systems. With our approach, however, one does not have to rely so heavily on guessing to develop an adaptive system; each of the required components can be studied, in general form, by the use of well-developed statistical and numerical concepts, which can be extended by general and cumulative mathematical research.

It is also important that GDHP, unlike the earlier work on perceptrons and hill-climbing, directly addresses the requirement for foresight in decisionmaking. GDHP is similar in spirit to the Samuels checker-playing program, which used adaptive logic to estimate a "static position

evaluator" (an estimated J function). Samuels' program was one of the few strong successes of the early research, and its importance may have been underestimated. In artificial intelligence, it is common to perform a two-level position evaluation, with one level being global and the other based on a local search. However, when both are the result of adaptation, they can be combined, just as long-term trends and seasonal factors may both appear in a single regression equation to predict energy demand. (As a practical matter, this can work best with a "recursive" reality vector $\boldsymbol{R}$, as implied by our discussion of filtering in Section V, etc.)

There are also some technical reasons to expect better results with the new approach. Almost all of the old perceptron work was based on the McCulloch–Pitts model of the neuron, which was both discontinuous and nondifferentiable [21], [22]. It was felt, in the 1950's, that the output of a brain cell had to be 1 or 0, because the output consisted of sharp discrete "spikes." More recent work in neurology has shown that higher brain cells output "bursts" or "volleys" of spikes, and that the strength of a volley may vary continuously in intensity. (See Grossman [25, pp. 75, 105, 106].) This suggests the CLU model to be discussed in the Appendix. In any event, to exploit basic numerical methods, one must use differentiable processing units. Minsky once experimented with a "jitters" machine, which estimated one derivative per time cycle by making brute-force changes in selected variables; however, the methods to be discussed in the Appendix yield derivatives of all variables and parameters in one major time cycle and thereby multiply efficiency in proportion to the number of parameters (N), which may be huge.

Minsky has also proved that networks of perceptrons are inherently limited, if they are not "recursive" (self-referencing); the used of short-period lags or filtering in the cause and effect system ($\boldsymbol{f}$) provides this self-referencing. Furthermore, recent developments in hardware make it increasingly feasible to accept the extra costs required for a general adaptive system.

In recent years, approaches similar to ours have begun to prove very useful. Statistical "feature analysis" based on linear models is now central to the field of pattern recognition [18]. Basic control-theory methods are being used in robots. With more general statistical methods, however, one may automate or enhance the process of defining "features," especially when features may be related to the dynamics of a time series. Likewise, more general optimization methods may extend the versatility of robots in problem-solving, scheduling, etc.

VIII. Potential Applications to Brain Research

Overview

Despite the rapid progress of recent years, the study of the brain still has some similarity to the study of physics at the birth of Newton. The general unifying principles are still *qualitative* in nature; *quantitative* results generally apply to individual experiments, to particular phenomena, or to microscopic systems. Since the time of Hebb, a small group of brain theorists have tried to develop a general mathematical model of the neuron, which could explain the more general adaptive power of the brain. However, even simple "neuron models" have often bogged down in complex differential equations, whose implications for the brain as a whole system are not understood, and which fail in any case to explain the gross division of the brain into structures like the cerebral cortex versus the limbic system and so on. No one has been able to simulate practical problem-solving abilities in a computer by implementing this kind of general neuron model. Indeed, some of these models amount to the use of correlation (covariation) coefficients in place of regression coefficients in a simple linear model, something which could never cope with a wide range of environments.

Research on intelligent systems offers the hope of a more unified and quantitative explanation of brain dynamics, similar to the unification achieved in physics by the idea of minimizing a Lagrangian function. As in the case of physics, this will require a more extensive development of the mathematics as well as new empirical work. The previous discussion suggests that there is a continuum of possible designs for an intelligent system, ranging from a basic system built from GDHP and conventional statistics, through to complex versions with many additional capabilities. Mathematical research will be needed to characterize this continuum more precisely, and empirical research will be needed to locate the brain within that continuum. Nevertheless, the hypothesis that the brain is an intelligent system already leads to important testable predictions and insights. These predictions are not proposed as an alternative to the well-known unifying principles of MacLean, Lashley, Freud, and others; rather, they are proposed as an explanation and extension of those principles.

The following section will compare the major features of the brain, as they are now understood, with the hypothesis that the brain is a basic intelligent system incorporating the features discussed in Sections III–V. It will first discuss evidence which is consistent with this hypothesis and point towards further predictions that need to be tested. Then it will discuss the major evidence which contradicts this hypothesis; it will suggest that this further evidence might be explained as the result of additional generalized capabilities in the brain, such as those proposed in Section VI, which require further mathematical research. (In other words, the hypothesis that the brain is an intelligent system still appears tenable at this point.) A few parts of this discussion do require some understanding of the Appendix, but only for the fine points.

Evidence Which Fits the Basic Hypothesis

The human brain is not a homogenous organ. It is made up of very distinct suborgans. Its general structure is shown in Fig. 2.

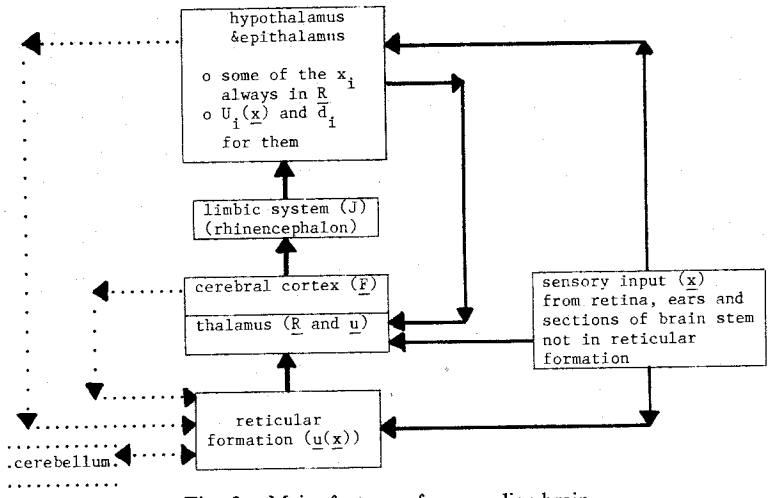


Fig. 2. Major features of mammalian brain.

The algebraic quantities in Fig. 2 identify brain structures which may perform the calculations discussed in Sections III–V. The solid lines represent known connections which fit this hypothesis; the dotted lines represent connections which do not. This chart is based on sources such as the Rockefeller University series on the Neurosciences [26], but the information may also be found in textbooks such as Grossman [25] and Gardner [27]. Three very small organs have been left out: 1) the olfactory bulb, which preprocesses smell data; 2) the subthalamus and corpus striatum, muscle control centers to be discussed below.

Every organ named in this chart is a very distinct mass of cells, but for analytical purposes they are often subdivided much further, sometimes to the cellular level or lower. Grossman sometimes refers to "connections," which are "anatomically indistinct but functionally important," and many of the textbook discussions gloss over such distinctions in some contexts; for example, Gardner (p. 255) describes "connections from the cortex to the cerebellum," which are then listed out as a set of indirect pathways via cells in the brain stem (apparently via the reticular formation).

Let us outline the major parallels between this chart and our framework, starting from the top and working down.

The hypothalamus is a tiny organ at the base of the brain, in the center of the head. It is tightly connected to the pituitary gland. Though small, it has been subdivided and analyzed in great detail, because its contents are highly stereotyped (preprogrammed) and very important in controlling behavior. For example [25, p. 195–201], it is known to be the main site registering basic drives, such as hunger, which do not result from learning. Thus an association with the utility function (U) seems plausible.

However, the calculations in Section IV only require certain derivatives, the $U_i(\boldsymbol{x})$ and the associated d_i, not a utility function as such. Since the hypothalamus registers many drives, and does not appear to "sum" them, it is more plausible to assume that the hypothalamus *measures* some components of the vector $\boldsymbol{x}$ and also outputs the associated $U_i(\boldsymbol{x})$. Note that the current state of the environment $\boldsymbol{x}$ is a vector made up of many components; it is well known [25, p. 233] that the hypothalamus works as a sensory center in measuring some of these variables directly. Some of the x_i output by the hypothalamus may actually be preprogrammed functions of other variables. However, the calculations in the Appendix are not invalidated when redundant variables are added to $\boldsymbol{x}$.

The epithalamus, connected to the pineal gland, has connections similar to those of the hypothalamus [25, p. 43]. Other factors besides hunger enter into the U_i here, but they are far beyond the scope of this paper.

The limbic system was singled out long ago by Papez, and more recently by MacLean, as the "emotional" system of the brain. Recent work [25, p. 287–295] has supported this view, but has emphasized that the limbic system is very complex, and subject to variation from individual to individual; its specific contents are subject to change through learning. "Emotions" may be interpreted as mainly being evaluations of the environment, either as favorable or as unfavorable to the organism; the estimated J function provides such an evaluation, for use in reinforcing action, and yet J itself is subject to change through learning. (The J-versus-U distinction is also parallel to "primary reinforcement" versus "secondary reinforcement" in experimental psychology.) In the Appendix, the derivatives of J are crucial to the adaptation of both parameter types $\boldsymbol{a}$ and $\boldsymbol{b}$; the limbic system appears crucial to the reinforcement of overt behavior [25, p. 309] and to adaptation generally [25, p. 292].

Given the electrical (electrochemical) connections in Fig. 2, it has been a disturbing paradox to some that the limbic system has such a powerful impact on behavior. In our discussion in the Appendix, there is no such paradox, because of the requirement that other information must be flowing backwards along these same connections.

Furthermore, in Section V we have noted that "dynamic robustness" appears to require that the parameters in the $\boldsymbol{f}$ network, as well, be adapted in proportion to the derivatives of J. This would imply that a cutoff of the derivative calculation (e.g., by cutting off the limbic system) would cut off adjustments to the $\boldsymbol{f}$ network, even though it would

not disturb the *existing contents* of that network. Such a situation may seem paradoxical; nevertheless, after reviewing extensive evidence and alternative hypotheses, Grossman [25, p. 407] does conclude: "These observations suggest that in man the hippocampus and perhaps the mammillary region and medial thalamus may form a pathway essential to the storage of new memories without itself being necessary for retrieval of well-established memories." (The hippocampus is the part of the limbic system adjoining the cerebral cortex, and, in Grossman's detailed discussion, is necessary by itself to memory formation.)

The cerebral cortex makes up most of the weight of the human brain, and it is commonly considered the seat of intelligence. It includes the outer gray matter, which covers the entire surface of the brain. Many popular accounts of the brain discuss the various lobes and halves of the cerebral cortex as if it were the entire brain. The electrical input to the cerebral cortex comes from the thalamus, a walnut-sized organ in the center of the brain. Given how much of human knowledge is stored in the cerebral cortex, both about objects and about the dynamics of the world, it seems plausible that it contains one's model of the world ($\boldsymbol{f}$). Electrical connections back from the cortex to the thalamus are consistent with the idea that the cortex produces forecasts that are compared with actual values in the thalamus. Certain parts of the thalamus are known to be very tightly linked with the reticular formation [25, p. 178]; therefore, it is not implausible that these $\boldsymbol{u}$ inputs receive different treatment from the $\boldsymbol{x}$ inputs, as our calculations would require. A small part of the thalamus, called the "nonspecific" part, is clearly not part of the $\boldsymbol{f}$ network itself; rather, it imposes synchronous operation on both the cerebral cortex and the limbic system, based on a cycle time which is relatively long (0.1–0.2 s) compared with the time necessary for a cell to respond to its inputs (less than 0.01 second). (See Grossman [25, p. 183] and Purpura *et al.* [26].) The calculations described in the Appendix clearly require synchronization and a relatively long cycle time.

The reticular formation has access to all "motor neurons" [25, p. 179] which in turn control all voluntary muscles. Here, we define the *reticular formation* to include all midbrain and hindbrain cells that are associated with it developmentally and functionally, including those cells with a direct control over action as discussed in [25, p. 159]. Among the crucial functions of an action network $\boldsymbol{u}(\boldsymbol{x})$ is the control of muscles such as eye muscles, which implies a control over "focusing" or "attention"; these functions are known to be an important aspect of the reticular formation, yet it is also known that they do not imply that the reticular formation is a generalized sensory input system or controller over the cortex [25, p. 346].

The most unavoidable implication of our mathematics is that there must be a backward flow of information along the major pathways of information flow to permit adaptation of those pathways. Adaptation of parameters is in response to "derivatives" resulting from the backward flow, a flow that obeys equations given in the Appendix. In actuality, there does exist a "retrograde transmission system" of chemical flows backwards across the electrochemical network shown in Fig. 2; it is widely used in mapping out connections between brain cells. This system is slow in laboratory preserved brains, but chemicals such as strychnine are known to be transmitted very quickly at times in living organisms. Freud's theory of ego formation and human development [28], [29] is rooted in the idea of such a backward flow of chemicals (representing "psychic energy" or "cathexis" associated with objects or variables) controlling the adaptation of the organism. New studies, based on microinjections of selected chemicals [25, p. 309–311], show that chemical flows do play a crucial role in reinforcing behavior. If the analysis above is correct, then such studies have only just begun to explore a very important territory; it should be possible to observe how reinforcing chemicals (such as norepinephrine) flow backwards from cell B to cell A through microtubules shortly after electrical impulses jump the synapse from A to B, if B was not "saturated" (see the description of CLU in the Appendix), following the equations of dynamic feedback.

The full functional implications of these parallels, and the specific types of information one would expect to see stored in the subsystems, are beyond the scope of this paper. Furthermore, the approach here suggests that the *contents* of the subsystems are not fixed or preordained in any case; it is the mechanics and mathematics of adaptation that are most invariant.

The approach here is consistent with the work of Lashley, who first showed a high degree of generality and interchangeability between parts of an individual brain component, the cerebral cortex [25, p. 380]. However, it is inconsistent with those "general neuron models" which would assume identical equations for all cells in all components of the brain.

Evidence Which Contradicts the Basic Hypothesis

There are a number of apparent discrepancies with the interpretation discussed above. This should be expected, because our discussion in Sections III–V described a basic minimal system to facilitate understanding of the underlying principles. The human brain evolved to be optimal, not minimal. These apparent discrepancies can be explained as additions to our basic design to increase the efficiency or survivability of the system. Some of these additions may also be very useful in robotics and automation.

The most important gap appears to be the strong connection from the cerebral cortex to the reticular formation and to the limbic system. This is partly related to an oversimplification in our discussion of the vector $\boldsymbol{R}$. In defining GDHP, $\boldsymbol{R}$ was originally defined as a description of all aspects of the environment, including those seen directly and those inferred by filtering (short-term memory) and the like. But we also pointed out that a complete cause-and-effect system should be able to estimate "hidden variables" (as in nonlinear filtering theory); these would be estimated as part of the work of the $\boldsymbol{f}$ network, as is common in control theory work even now. Since certain components of the vector $\boldsymbol{R}$ are calculated in the

cerebral cortex proper, one would expect this information to be available as part of the vector received as input by both the J and $\boldsymbol{u}$ networks.

Related to this discrepancy is the high mutual access of all parts of the brain to each other. In an efficient system, one would somehow expect mutual access to information, *qua* information, between networks. One might even expect some consideration of the value of a cell to other networks in the adaptation of that cell itself. As long as the basic final output cells of each network are kept free of any debilitating "conflicts of interest," this kind of mutual responsiveness could be allowed throughout the network and could have many implications; but problems could arise, for example, if the estimate of J were adapted so as to maximize the *assessment* of J (by taking feedback directly from the $\boldsymbol{u}$ network). This would bias J upwards, and lead to various distortions.

A second major discrepancy is related to the connections from the hypothalamus and thalamus to the reticular formation: the absence of sleep in our discussion so far. If GDHP were always run in real time, a weakness results: if the only $\boldsymbol{R}$ considered are actual situations, then the optimization process will be biased, and unable to anticipate or take advantage of future situations very different from the past. Decoupled simulations can solve this problem. However, they require a specific network control mechanism, to turn them on and off. The hypothalamic and thalamic pathways, which control sleep and dreams may serve this and related purposes in the brain. The reticular formation, like the cerebral cortex and limbic system, is involved in the activities during sleep [25, p. 346]; otherwise, one might have considered the possibility that the cortex represents a closed model $\boldsymbol{f}$ independent of the actual $\boldsymbol{u}(\boldsymbol{x})$ network. Howard's equation would allow such a procedure, though it would be less efficient than what we have described.

The pathways controlling sleep are known to be able to inhibit all muscular activity in sleep; otherwise, the organism would act out its dream actions in reality. Since these pathways converge on the reticular formation, we would question the research study which claims a few direct links from the cortex to motor neurons [25, p. 155], in baboons as opposed to other mammals; that claim also conflicts with the observation that the action-oriented parts of the cerebral cortex relate to generic movements like walking rather than contracting specific muscle fibers [25, p. 157].

The cerebellum, the subthalamus, and the basal ganglia are all outside the framework of Section IV. The cerebellum is a muscle-oriented center; it is crucial to the fine-tuning of muscle control, but not to any specific function [25, p. 51]. The system described above permits control only at intervals of 0.1–0.2 s, rather than the 0.01-s or less of which the muscular system is capable. To add such fine control requires the addition of a subordinate optimization system, as discussed in Section VI. The basal ganglia are also devoted to the fine control of muscles [25, p. 158]. It is unclear whether they or the subthalamus [25, p. 159] should be regarded as parts of a subordinate optimization system, or as evolutionary relics like the human appendix or tonsils. In this view, the lower brain stem and spine would be controlled by the subordinate system, and would not be part of the "higher" $\boldsymbol{u}(\boldsymbol{x})$ of the upper reticular formation.

One significant discrepancy remains: the dense connections from the limbic system to the hypothalamus and epithalamus. Since the function J is parallel to U, these connections do not seem surprising at first. Yet in GDHP, one would not really expect a single ultimate $J_1(T)$ to reside in the brain, as described in the Appendix; rather, one would expect a set of $J_i(T-1)$ to be represented explicitly, as a *set* of basic cells, and one would expect $J_1(T)$ to be defined in theory as their sum but not calculated (since there is no need to). For physical convenience, one might even locate these "top" cells in the hypothalamus and epithalamus, next to the corresponding U_i cells; thus the mammillary bodies of the hypothalamus may play a role in the limbic system analogous in a way to the role of the thalamus in the cerebral cortex.

This discussion does not provide a definite explanation of the more microscopic organization which exists within the limbic system. Some of that division, as suggested by MacLean, may be based on a coexistence of early preprogrammed subsystems with more advanced, adaptive systems. On the other hand, Section VI suggests the possibility of a more fundamental explanation. Or both possibilities may be correct, if the limbic system combines preprogrammed subsystems, subsystems based on GDHP, and subsystems based on the possibilities in Section VI.

Appendix
Computational Aspects of GDHP

This Appendix will spell out procedures for the low cost calculation of the derivatives required with GDHP, including $E_a'(k)$ as defined in (6) and the derivatives of $J(\boldsymbol{F}(\boldsymbol{R}, \boldsymbol{e}, \boldsymbol{b}), \boldsymbol{a})$ with respect to the b_k parameters. For the sake of generality, it will be assumed that J and $\boldsymbol{F}$ are arbitrary functions, represented as a hierarchy of elementary operations (functions) set up to be calculated efficiently in a network of distributed or parallel processors [30]. This will make the calculations appear more complex, as they are written out here, but it actually allows them to be broken down into elementary operations which require a simple pattern of connections between processors.

Before proceeding, we must briefly state some general results on the efficient calculation of derivatives, originally derived from the "dynamic feedback" principle [10]. These results permit the calculation of *all the derivatives* required in adapting the entire network at a cost comparable to that of exercising the network one time. To maintain generality, we will use different notation in (8)–(11) than is standard in the rest of this paper.

For a general, nonlinear dynamic system with parameters $\boldsymbol{c}$, running from time 0 to time T:

$$\boldsymbol{z}(t+1) = \boldsymbol{s}(\boldsymbol{z}(t), \boldsymbol{u}(t), \boldsymbol{c}), \tag{8}$$

one may linearize about a solution trajectory using the

Jacobian of $\boldsymbol{s}$, S:

$$\boldsymbol{x}(t+1) = S(t)\boldsymbol{x}(t) + \boldsymbol{k}(t). \tag{9}$$

The derivatives of $\boldsymbol{z}(T)$ with respect to $\boldsymbol{z}(0)$ are implicit in:

$$\boldsymbol{x}(T) = S(T-1)S(T-2)\cdots S(0)\boldsymbol{x}(0) + \boldsymbol{k}'(t) \tag{10}$$

and a similar formula may be derived (summing over t) for derivatives with respect to $\boldsymbol{c}$. From this, one can easily verify the validity of the following recursion formulas to determine the derivatives of a target variable $z_i(T)$ with respect to *all* of the $z_j(0)$ (to appear in $x_j'(0)$) and with respect to *all* of the components of $\boldsymbol{c}$ (to appear in $\boldsymbol{w}(0)$):

$$\boldsymbol{x}'(T) = \boldsymbol{e}^{i^T} \tag{11a}$$

$$\boldsymbol{x}'(t) = \boldsymbol{x}'(t+1)S(t) \tag{11b}$$

$$\boldsymbol{w}(T) = \boldsymbol{0} \tag{11c}$$

$$\boldsymbol{w}(t) = \boldsymbol{w}(t+1) + \boldsymbol{x}'(t)s_c'(t), \tag{11d}$$

where s_c' refers to the matrix of derivatives of s_i with respect to c_k.

Likewise, to compute all the second derivatives of $z_i(T)$ with respect to a particular $z_j(0)$ and all of the components of $\boldsymbol{c}$, one may differentiate (10) with respect to the $\boldsymbol{a}$ and verify the following additional recursion relations yielding such derivatives in $\boldsymbol{v}(0)$ (strictly speaking, the derivatives being calculated are "ordered derivatives" or system derivatives rather than conventional partial derivatives; to be sure of the rigor of the obvious steps in verifying these relations, consult [10]):

$$\boldsymbol{x}(0) = \boldsymbol{e}^j \tag{11e}$$

$$\boldsymbol{x}(t+1) = S(t)\boldsymbol{x}(t) \tag{11f}$$

$$\boldsymbol{v}(T) = \boldsymbol{0} \tag{11g}$$

$$\boldsymbol{y}'(T) = \boldsymbol{0}^T \tag{11h}$$

$$\boldsymbol{y}'(t) = \boldsymbol{y}'(t+1)S(t) + \boldsymbol{x}'(t+1)S'(t)\boldsymbol{x}(t) \tag{11i}$$

$$\boldsymbol{v}(t) = \boldsymbol{v}(t+1) + \boldsymbol{x}'(t+1)S_c'(t)\boldsymbol{x}(t) + \boldsymbol{y}'(t+1)s_c'(t) \tag{11j}$$

where S_c' is defined such that each component of $\boldsymbol{v}$ is updated based on the derivative of $S(t)$ with respect to the corresponding component of $\boldsymbol{c}$ (accounting only for direct effects as in twice differentiating the function $\boldsymbol{s}$).

To apply these methods to differentiating E with respect to the parameters $\boldsymbol{a}$ of $J(\boldsymbol{R}, \boldsymbol{a})$, a network representation of J must be assumed:

$$J = z_1(T) \tag{12a}$$

$$z_i(t+1) = g_{i,t}(\boldsymbol{z}(t)) \tag{12b}$$

$$\boldsymbol{z}(0) = \boldsymbol{R} \tag{12c}$$

where t represents an elementary time cycle in the computational process, where each i may refer to an independent processor (or a simple calculation). (In the brain, the elementary cycles appear to be about 3 m long, while a major time cycle, including a whole *set* of calculations from cycle 1 to cycle T, is about a tenth or a fifth of a second, depending on where measured.) For readability, it is being assumed that each cycle $t+1$ only uses information from cycle t; however, the generalization to assume arbitrary lags is trivial.

For simplicity, a single network representation can be used for $\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{e}, \boldsymbol{b})$, which includes both the stochastic model $\boldsymbol{f}$ and the action network $\boldsymbol{u}$:

$$\boldsymbol{F} = \boldsymbol{y}(T) \tag{13a}$$

$$y_i(t+1) = h_{i,t}(\boldsymbol{y}(t)) \tag{13b}$$

$$\boldsymbol{y}(0) = \boldsymbol{R}//\boldsymbol{e} \tag{13c}$$

where // denotes the concatenation of two vectors. We will assume that each parameter a_i enters into precisely one of the functions $g_{j,t}$, and that each of the b_i enters into one $h_{j,t}$. This does not reduce generality, because we could define $g_{j,t} = a_i$, and then use this quantity throughout the network of functions $g_{k,t+1}$.

The following six steps will show how to calculate the derivatives in (6) and the corresponding derivatives of J with respect to the b_k, for a given choice of $\boldsymbol{R}$ and $\boldsymbol{e}$. To get the expectation values across many $\boldsymbol{R}$ and $\boldsymbol{e}$, one merely need repeat this process. Alternatively, in a real-time system, one can adjust parameters immediately (though slightly) to respond to the current derivatives. These six steps can be regarded as flows of information up or down the greater network from $\boldsymbol{x}$ to $\boldsymbol{u}$ to $\boldsymbol{f}$ to $J(\boldsymbol{f})$; each derivative calculation can be made on the basis of information already "at" the relevant processor or "at" one of the processors it is already connected to.

Step 1, Calculate J(F(R, e, b)a): To do this, set $\boldsymbol{y}(0) = \boldsymbol{R}//\boldsymbol{e}$; use (13b) to calculate $\boldsymbol{y}(t)$ for all cycle times t; use (13a) to calculate $\boldsymbol{F}$. Then set $\boldsymbol{z}(0) = \boldsymbol{F}$; use (12b) to calculate $\boldsymbol{z}(t)$ for all cycle times; use equation 12a to get J. The $\boldsymbol{z}$ and $\boldsymbol{y}$ values need to be stored for use in step 2, but can then be forgotten.

Step 2, Calculate Derivatives of J(F(R, e, b), a) with respect to R and b: To do this, use a backwards sweep (11a)–(11d), which requires the following series of calculations:

$$z_1'(T) = 1$$

$$z_i'(t) = \sum_j z_j'(t+1) * \frac{\partial g_{j,t}}{\partial z_i(t)}$$

$$\boldsymbol{y}'(T) = \boldsymbol{z}'(0)$$

$$y_i'(t) = \sum_j y_j'(t+1) * \frac{\partial h_{j,t}}{\partial y_i(t)}.$$

The beginning section of $\boldsymbol{y}(0)$ will then contain the derivatives with respect to the components of $\boldsymbol{R}$; let us add each of them to the corresponding $U_i(\boldsymbol{R})$, and store the vector of sums into a vector $\boldsymbol{V}$. The derivatives with respect to $\boldsymbol{b}$ come from (11d), which normally requires us to add up its right-hand term over all times t to get $\boldsymbol{v}(0)$; however, in this case we have assumed that each b_i occurs in only one function $h_{j,t}$ so that we only need to read off one term from that sum to get the derivatives required to adjust the

action network ($\boldsymbol{u}$):

$$\frac{\partial J(\boldsymbol{F}(\boldsymbol{R}, \boldsymbol{e}, \boldsymbol{b}), \boldsymbol{a})}{\partial b_i} = y_j'(t+1) * \frac{\partial h_{j,t}}{\partial b_i}.$$

Step 3, Calculate $J(\boldsymbol{R}, \boldsymbol{a})$: Set $z(0) = \boldsymbol{R}$, and then use (12b) to calculate z for all cycle times t. Be sure to store these values.

Step 4, Calculate Derivatives of $J(\boldsymbol{R}, \boldsymbol{a})$ with respect to $\boldsymbol{R}$: Use a backwards sweep, as in (11a) and (11b). This implies

$$z_1'(T) = 1$$

$$z_i'(t) = \sum_j z_j'(t+1) * \frac{\partial g_{j,t}}{\partial z_i(t)}.$$

Note that the results of this step differ from those of step 2, because the values of the vectors $z(t)$ were recalculated in step 3. As in step 3, be sure to store the new values for the z'.

Step 5, Calculate the Weights in (6), Using the Definition of d_j in 7: set $x_i(0) = 2W_i(z_i'(0) - V_i)$.

Step 6, Compute the Derivatives in (6): Use a triple sweep, as per (11a)–(11j). In step 4, we already computed z', which corresponds to $\boldsymbol{x}'$ in (11a) and (11b). Next, we must compute $\boldsymbol{x}(t)$ as defined in (11f) for all cycle times t. We will start from $\boldsymbol{x}(0)$ as calculated in step 5; this is a linear combination of the possibilities implied by (11e), and it will result in the appropriate linear combination of second derivatives as required in (5) because the (11) are linear in the $\boldsymbol{x}(t)$. To do all this, first perform, for t from 0 to $T-1$ and for all i:

$$x_i(t+1) = \sum_j \frac{\partial g_{i,t}}{\partial z_j(t)} * x_j(t).$$

Next set $\boldsymbol{y}'(T) = 0$, and perform, for t from $T-1$ to 0, the calculations implied in (11h):

$$y_i'(t) = \sum_j y_j'(t+1) * \frac{\partial g_{j,t}}{\partial z_i(t)} + \sum_{j,k} z_j'(t+1) * \frac{\partial^2 g_{j,t}}{\partial z_i(t)\partial z_k(t)} * x_k(t).$$

Finally, the derivatives we need to adapt the J network can be calculated by (11j). That equation appears to require a sum of its two right-hand terms over all times t. However, we have assumed that each a_k occurs in only one function $g_{j,t}$, so that we can simply read off the derivative of E with respect to each parameter a_k as follows:

$$E_a'(k) = \sum_j z_j'(t+1) * \frac{\partial^2 g_{j,t}}{\partial z_i(t)\partial a_k} * x_i(t) + y_j'(t+1) * \frac{\partial g_{j,t}}{\partial a_k}.$$

For certain applications (e.g., artificial intelligence or neuron modeling), one may restrict the functions $g_{i,t}$ and $h_{i,t}$ so as to simplify these calculations. For example, with continuous logic units (CLU), each processor i would output a linear combination of its inputs weighted by its parameters, but restricting its output to the interval (0, 1). In this case, one may assume that $y^*(t) = 0$ above (for complex reasons), and the calculations are simplified. In this case, the backward sweeps involve the following at each processor: 1) sum the flows of feedback into the processor; 2) reset the sum to zero if the original output of the processor was cut off at 1 or 0; and 3) send back as a "flow" to each of its input processors the sum (as reset) multiplied by the parameter applied to that input processor. Note that these flows of feedback followed the same paths (backwards) as the original forwards calculation followed.

REFERENCES

[1] E. Raia, "Helping machine tools help themselves," *High Technology*, June 1985.

[2] R. A. Freitas and W. P. Gilbreath, Eds., *Advanced Automation for Space Missions*, NASA Conference Publication 2255. Washington, DC: GPO, 1982.

[3] F. Hayes-Roth *et al.*, Eds., *Building Expert Systems*. Reading, MA: Addison-Wesley, 1983.

[4] H. Raiffa, *Decision Analysis: Introductory Lectures on Making Choices Under Uncertainty*. Reading, MA: Addison-Wesley, 1968.

[5] P. Werbos, "Generalized information requirements of intelligent decision-making systems," in *SUGI 11 Proc.*, Cary, NC: SAS Institute, 1986. (A revised version, available from the author, is easier to read and contains more discussion of psychology.)

[6] R. Howard, *Dynamic Programming and Markhov Processes*. Cambridge, MA: MIT Press, 1960.

[7] P. Werbos, "Changes in global policy analysis procedures suggested by new methods of optimization," *Policy Anal. Inform. Syst.*, vol. 3, no. 1, June 1979.

[8] G. Hadley, *Nonlinear and Dynamic Programming*. Reading, MA: Addison-Wesley, 1964.

[9] P. Werbos, "Solving and optimizing complex systems: Lessons from the EIA long-term model," in *Energy Models and Studies*, B. Lev, Ed. New York: North-Holland, 1983.

[10] ——, "Applications of advances in nonlinear sensitivity analysis," in *Systems Modeling and Optimization*, R. Drenick and F. Kozin, Eds. New York: Springer-Verlag, 1982.

[11] ——, "Econometric methods," in *Handbook of Energy Modeling*, J. Weyant and T. Kuczmowski, Eds., to be published.

[12] K. H. Booth and D. R. Cox, "Some systematic supersaturated designs," *Technometrics*, vol. 4, no. 4, Nov. 1962.

[13] B. Efron and C. Morris, "Comment," *American Statistical Association*, vol. 72, no. 357, p. 91–93, Mar. 1977.

[14] R. J. Solomonoff, "Mathematical foundations of induction," *Synthese*, Sept. 1964. (See also R. J. Solomonoff, "A formal theory of inductive inference," *Inform. Contr.*, Mar./June 1964.)

[15] P. Werbos, "Advanced forecasting methods for global crisis warning and models of intelligence," in *General Systems Yearbook*, 1977.

[16] ——, *A Statistical Analysis of What Drives Industrial Energy Demand: Volume III of the PURHAPS Model Documentation*, DOE/EIA-0420/3. Washington DC: National Energy Information Center (NEIC), 1983, ch. 4. (Available at no charge on request to NEIC at (202)-252-8800.)

[17] P. Werbos and J. Titus, "An empirical test of new forecasting methods derived from a theory of intelligence: The prediction of conflict in Latin America," *IEEE Trans. Syst., Man, Cybern.*, Sept. 1978.

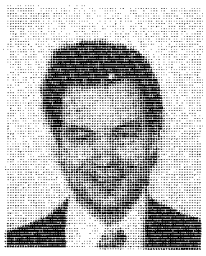
[18] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley-Interscience, 1975.

[19] M. A. Arbib *et al.*, Eds., *Neural Models of Language Processes*. New York: Academic, 1982.

[20] S. J. Segalowitz, Ed., *Language Functions and Brain Organization*. New York: Academic, 1983.

[21] B. Widrow, "Generalization and information storage in networks of adaline 'Neurons'," in *Self-Organizing Systems 1962*, M. C.

Yovits *et al., Eds.* Washington, DC: Spartan, 1962.
[22] F. Rosenblatt, "A comparison of several perceptron models," in *Self-Organizing Systems 1962*, M. C. Yovits *et al.*, Eds. Washington, DC: Spartan, 1962.
[23] M. Minsky, "Steps towards artificial intelligence," in *Computers and Thought*, E. A. Feigenbaum and J. Feldman, Eds. New York: McGraw-Hill, 1963.
[24] A. Guzman, "Computer recognition of three-dimensional objects in a visual scene," Ph.D. dissertation, Mass. Inst. Technol., Cambridge, 1968, and MIT Project MAC Report. (Available through Clearinghouse for Federal Scientific and Technical Information, Springfield, VA, 1970.)
[25] S. P. Grossman, *Essentials of Physiological Psychology*. New York: Wiley, 1973.
[26] F. O. Schmitt, Ed., *Neurosciences* (first and second study programs). New York: Rockefeller University, 1970 and 1971.
[27] E. Gardner, *Fundamentals of Neurology*. Philadelphia, PA: Saunders, 1963.
[28] S. Freud, *The Ego and the Id* (standard ed.). London: Hogarth, 1961.
[29] D. Yankelovitch and W. Barrett, *Ego and Instinct: The Psychoanalytic View of Human Nature-Revised*. New York: Vintage, 1971.
[30] E. Lerner, "Parallel processing gets down to business," *High Technology*, July 1985.



Paul J. Werbos holds four degrees from Harvard and the London School of Economics, in economics, in international political economy, mathematical physics, and applied mathematics.

He is the lead analyst in the Energy Information Administration (EIA) for industrial and transportation energy demand, for which he has built detailed econometric models. In the past, when EIA projected to the year 2000 and beyond, he had lead responsibility for evaluating these forecasts and comparing them with other long-term projections in depth, as well as responsibility for evaluating mid-term models of oil and gas supply. He has served on three interagency committees, involving a reassessment of *Global 2000* data and models, emissions assumptions for acid rain, and data revision policy. He is on the organizing committee of the Global Futures Roundtable, and has published papers and government reports on subjects ranging from future electricity demand and a comparison of global models through to basic physics, mathematical methods, and philosophy. Before working for the government, he taught at the University of Maryland, and worked on research projects to develop mathematical methods and computer-based applications at Maryland, Harvard, the Massachusetts Institute of Technology, the RAND Corporation, the Center for Research in Conflict Resolution, and others.