RC24742 (W0902-048) February 9, 2009
Mathematics

IBM Research Report

Neural Learning of Kalman Filtering, Kalman Control, and System Identification

Ralph Linsker
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

Neural Learning of Kalman Filtering, Kalman Control, and System Identification

Ralph Linsker

Abstract— **Kalman filtering and control methods have been important in engineering since they were introduced in 1960. Recent theoretical work on principles that may describe core computations of cerebral cortex has focussed on the possible role of Kalman filtering and its nonlinear extensions, and of Bayesian inference more generally. However, no neural network (NN) method for learning either the optimal Kalman filter or the optimal Kalman control matrix (for nonstationary control problems) has to my knowledge been described.**

Here we show that Kalman estimation (including filtering and prediction) and control, and system identification, can be fully implemented within an NN whose only input is a stream of noisy measurement data. The operation of the fully-integrated algorithm is illustrated by a numerical example. The resulting network is a multilayer recurrent NN that may be useful for engineering applications, and that also bears certain resemblances to the putative 'local circuit' of mammalian cerebral cortex.

I. Introduction

This work is motivated primarily by a fundamental question in neuroscience: Do the regularities observed in the architecture of mammalian cerebral cortex[1] point to the existence of certain core functions that underlie the variety of sensory, motor, and other processing performed by cortex, and if so, what might these core functions be?

The main mathematical result presented here is a novel recurrent neural net (NN) architecture and algorithm that both learns and executes Kalman estimation (including prediction and filtering) and control[2], as well as performing system identification. This result is of interest for several reasons, both engineering and neurobiological:

1) It has been proposed that the core functions of cerebral cortex may relate to the operation of a Kalman filter (KF) or more generally to Bayesian inference (see Section VII for discussion). For example, Rao and Ballard[3,4] have shown that a simplified KF-like algorithm can be used in a hierarchical generative model for visual recognition. However, ref. [3] [see, e.g., Eqs. (5.5) and (6.5)] does not use a NN to learn a KF; and ref. [4] (Sec. 4.3) uses a 'constant fixed and possibly scalar' value for the KF matrix, in lieu of any KF learning. Other workers have also described the use of a Kalman filter by a NN where, however, the KF is learned by non-NN means (see [5] for references). The difficulties that have limited prior efforts to perform KF learning in a NN include the fact that Kalman's algorithm requires updating the KF matrix by using multiple matrix inversions and matrix-times-matrix (as opposed to matrix-times-vector) computations, which are not 'natural' NN operations.

   Such proposals relating KF operation to cortical functions motivate the question of how (and whether) the *learning* of an optimal KF can be performed by a NN. Despite these proposals, no NN implementation of KF learning has to my knowledge been previously described.[1]
2) NN learning of Kalman control (KC) has also not been previously described, apart from the special case of *stationary* control[7].
3) The present algorithm is unsupervised and, unlike the classical Kalman methods, requires no side information describing the measurement process or the dynamics of the external system (the 'plant'). This is appealing because biological systems must generally learn without such side information. The algorithm learns the required properties of the combined plant+measurement process using only a stream of noisy measurement data.
4) To the extent that the algorithm is based on local computations, it may be of value for hardware implementation.
5) The present algorithm appears to impose substantial constraints on the network architecture required to support it. Interestingly, the resulting NN architecture bears several similarities to that found in cerebral cortex at the level of the 'local circuit' (LCC) (see Sections VI and VII).

In this paper we show how optimal Kalman prediction and control (KPC), as well as system identification, can be learned and executed by a recurrent neural network composed of linear-response nodes, using as input only a stream of noisy measurement data. We then identify the constraints on the resulting NN architecture, and compare this architecture with that of the LCC. Further details and derivations are given in [5]. New simulation results illustrate the operation of the fully-integrated NN algorithm.

II. Transforming the Kalman algorithms for NN implementation

We start with the classical KPC algorithms of Kalman[2]. The main equations are summarized in Table I (left column). In essence, Kalman filtering (resp., one-step-ahead prediction) optimally blends a noisy measurement y_t at time t

Ralph Linsker is with the IBM T. J. Watson Research Center, PO Box 218, Yorktown Heights, NY 10598 (email: linsker@us.ibm.com).

[1] A recent paper[6] is motivated by the goal of a KF-learning NN; however, its algorithm does not learn a KF even approximately (see [5], App. C).

with a prediction based on an internal model (learned from prior data), to generate an improved estimate of the 'plant' (external system) state x at time t (resp., $t+1$). Kalman's algorithm for linear quadratic control learns a sequence of matrices $\{S_\tau, \tau = N-1, N-2, \ldots, t_0\}$, which is used to generate control signals $\{u_t, t = t_0, t_0+1, \ldots, N-1\}$ that minimize a 'cost-to-go' function J. J contains quadratic terms that penalize both large u_t and large deviations x_t from the intended target $x_{\rm targ} = 0$ that is to be reached or approached at time $N > t_0$.

We now make the problem more difficult, but also more biologically relevant, by insisting that *only* a noisy stream of input data is available – and *not* the matrices that define the plant dynamics and measurement processes (side information that is assumed known when using Kalman's classical methods). Thus the NN must learn system identification as well as estimation and control.

Since the plant-to-measurement transformation matrix cannot, in principle, be discovered by the NN in the absence of side information,[2] we transform the classical Kalman equations into a mathematically equivalent form that depends only on values obtainable from the measurement stream. See Table I, right column, for the definitions of the new variables, and the resulting transformed equations. Thus all variables referring to the plant state x are replaced by variables referring to the measurement space.

III. Derivation of the NN activation dynamics and Hebbian learning rules for Kalman prediction and control (Table II)

To embody the transformed equations in an NN, we approximate the various expectation values $E[\cdots]$ by finite-sample[3] averages $\langle\cdots\rangle$. The validity of this approximation depends on the sample items being sufficiently independent of one another (which we assume here) and on the size of the sample being sufficiently large (see [5] for discussion).

The KF matrix to be learned is K_t in the classical formulation, and Z_t in our transformed version; they are related via $I - HK_t = RZ_t^{-1}$ (Table I).[4]

A crucial fact we use in deriving the NN algorithm for KF is this: If $Z_0 = E[\eta_0\eta_0']$ and η_{t+1} evolves as in the first equation of Table II, then $Z_t = E[\eta_t\eta_t']$ (see [5] for proof). By evolving (over time) a representative sample $\{\eta_t^p\}$ of η_t values such that $\langle\eta_t^p(\eta_t^p)'\rangle_p \approx E[\eta_t\eta_t']$ at each t, the NN evolves Z_t as required by the transformed equation for Z (within the limits of the finite-sample approximation).

The corresponding key to our KC learning algorithm is the 'evolution' (in *decreasing* time-index) of a vector w_τ in such a way that if $T_\tau = E[w_\tau w_\tau']$, then (up to the finite-sample approximation) $T_{\tau-1} = E[w_{\tau-1}w_{\tau-1}']$ (derived in [5]). Note that in KF, η_t has a physical meaning (i.e., the predicted minus actual value of y_t) and the same η is used for learning the KF matrix Z and for executing KF (i.e., computing the next prediction $\hat{y}_{t+1}^-$). By contrast, in KC, w_τ is computed using internal noise generators (having covariances as in Table II), and w_τ is used only for learning the KC matrix T, and not for KC execution (i.e., the computation of u_t using T_t).

Our NN algorithm for KF remains approximately valid even if the matrices describing the plant and measurement matrices change slowly with time, or change abruptly (in which case the algorithm yields a valid KF apart from a transient period following the change). The algorithm in effect performs 'extended Kalman filtering' (EKF) by linearizing the dynamics in the vicinity of the current operating point.

A. Technical issues

(a) The sample average: To obtain a set of measurement vectors at each time step of the Kalman algorithm, rather than a single vector, we assume either (i) that multiple input vectors – all obeying the same dynamics but with independent noise – are being measured at each time step,[5] and/or (ii) that each 'Kalman time step' (t in Table I, left column) is being performed by the NN using many successive (in time) observations of a plant variable (in which case the sample items may or may not be sufficiently independent to ensure $E[\cdots] \approx \langle\cdots\rangle$). Both (i) and (ii), and their combination, are treated in [5].

(b) Learning or computing the inverse of a covariance matrix: The inverses of two covariance matrices, Z_t^{-1} for Kalman estimation and T_t^{-1} for Kalman control (see Table I), are required by the NN learning algorithms. We have the option of learning either Z or Z^{-1} as a weight matrix (similarly for T). But neither the computation of $M^{-1}v$ (where M is a connection weight matrix and v is an activity vector), nor the learning of a weight matrix M^{-1} (where M is a covariance matrix), is a 'standard' or familiar NN operation. However, methods for performing these computations have been given in [8] and [9], respectively, and their use in the present context is detailed in [5].

To save space here, we simply write the Z and T update rules as '$\approx E[\cdots]$' in Table II (along with an explicit example of a Hebbian update rule for Z), and refer to Z^{-1} and T^{-1} in the NN activity calculations.

(c) Kalman's classical control algorithm requires learning a sequence of matrices 'backwards in time'; that is, a matrix S_N is defined at the future endpoint $\tau = N$ in time of the control operation to be generated, and $S_{\tau-1}$ is computed from S_τ for each $\tau = N-1, \ldots, t_0$ where t_0 is the current time. The NN algorithm and circuit must operate in real (forward-moving) time. This is accomplished by running the control matrix learning algorithm (through a set of decreasing

[2] We therefore use the term 'system identification' to mean the determination of the plant dynamics as these dynamics are reflected in the measured quantities y – e.g., the determination of the matrix $\tilde{F}$ rather than F.

[3] We use 'sample' in its statistical sense, to mean a subset of a population (ensemble) that is defined by a distribution.

[4] Notation: Prime denotes transpose, I is the identity matrix, and superscript '+' denotes a matrix pseudoinverse. Also, a hatted variable is an estimate, a superscript '-' denotes a predicted value, and a tilde denotes a transformed variable.

[5] Each such measurement vector y_t^p could describe, e.g., the coordinates of an identified and tracked feature p of an input 'scene' (which would require preprocessing to perform this tracking), or a patch p of pixel values, sound spectrogram intensities, etc. (which would require no such preprocessing).

TABLE I

SUMMARY OF CLASSICAL KALMAN SOLUTION, AND THE MATHEMATICALLY EQUIVALENT FORM USED TO GENERATE THE NN ALGORITHM

CLASSICAL KALMAN	TRANSFORMED
Plant & measurement models:	
$x_{t+1} = Fx_t + Bu_t + m_t$, $Q \equiv E[m_t m_t']$	$Y_{t+1} = \tilde{F}Y_t + \tilde{u}_t + \tilde{m}_t$, where $Y_t \equiv Hx_t$,
(Vectors are column vectors throughout.)	$\tilde{F} \equiv HFH^+$, $\tilde{u}_t \equiv HBu_t$, $\tilde{m}_t \equiv Hm_t$
$y_t = Hx_t + n_t$, $R \equiv E[n_t n_t']$	$y_t = Y_t + n_t$
Optimal filtering: Minimize	
$E[(x_t - \hat{x}_t)'C(x_t - \hat{x}_t)]$ w.r.t. $\{\hat{x}_t\}$	$E[(Y_t - \hat{y}_t)'\tilde{C}(Y_t - \hat{y}_t)]$, where
	$\hat{y}_t \equiv H\hat{x}_t$, $\hat{y}_t^- \equiv H\hat{x}_t^-$, $\tilde{C} \equiv H'^+ C H^+$
Optimal (one-step) prediction: Minimize	
$E[(x_{t+1} - \hat{x}_{t+1}^-)'C(x_{t+1} - \hat{x}_{t+1}^-)]$ w.r.t. $\{\hat{x}_{t+1}^-\}$	$E[(Y_{t+1} - \hat{y}_{t+1}^-)'\tilde{C}(Y_{t+1} - \hat{y}_{t+1}^-)]$
Optimal linear quadratic control: Minimize	
$J \equiv E[\Sigma_{t=t_0}^N (u_t' g u_t + x_t' r x_t)]$ w.r.t. $\{u_t\}$	$J \equiv E[\Sigma_{t=t_0}^N (\tilde{u}_t' \tilde{g} \tilde{u}_t + Y_t' \tilde{r} Y_t)]$, where
	$\tilde{g} \equiv H'^+ B'^+ g B^+ H^+$ and $\tilde{r} \equiv H'^+ r H^+$
Kalman estimation solution:	
Execution step:	
$\hat{x}_t = \hat{x}_t^- + K_t(y_t - H\hat{x}_t^-)$	$\hat{y}_t = y_t + RZ_t^{-1}(y_t - \hat{y}_t^-)$,
	where $Z_t \equiv HP_t^- H' + R$
$\hat{x}_{t+1}^- = F\hat{x}_t + Bu_t$	$\hat{y}_{t+1}^- = \tilde{F}\hat{y}_t + \tilde{u}_t$
Learning step:	
$P_{t+1}^- = F(I - K_t H)P_t^- F' + Q$	$Z_{t+1} = \tilde{F}(I - RZ_t^{-1})R\tilde{F}' + HQH' + R$
$K_t = P_t^- H'(HP_t^- H' + R)^{-1}$	[K_t eqn. & Z_t dfn. $\Rightarrow I - HK_t = RZ_t^{-1}$]
Kalman control solution:	
Execution step:	
$u_t = -L_t\hat{x}_t$	$\tilde{u}_t = (-I + T_t^{-1}\tilde{g})\tilde{F}\hat{y}_t$,
	where $T_t \equiv H'^+ S_t H^+ + \tilde{g}$
Learning step:	
$S_{\tau-1} = (F' - L_\tau' B')S_\tau F + r$	$T_{\tau-1} = \tilde{F}'\tilde{g}(I - T_\tau^{-1}\tilde{g})\tilde{F} + \tilde{r} + \tilde{g}$
$L_\tau = (B'S_\tau B + g)^{-1} B'S_\tau F$	

values of the time-index τ) at each of a subset of 'real' times t [5].

(d) When the measurement (y) space is of lower dimension than the state (x) space, a first order system in x may correspond to a higher-order system in y. In this case, the algorithm remains unchanged, but the measurement vector is augmented as needed by including computed difference terms $y(t) - y(t-1)$, etc. (or derivatives dy/dt etc. in a continuous-time implementation) as additional components, in the standard way that a general nth-order differential equation is transformed into a system of n first-order equations. With this augmentation, the equations for the y dynamics and for predicting $\hat{y}^-$ remain of first order. (In the absence of knowledge of the x-to-y transformation, such difference term(s) can be included provisionally to improve y-prediction, if prediction is found to fail in the absence of these terms.)

(e) Generalization: Our derivation of the NN learning algorithms for KF and KC has used the fact that we can define a vector v (which $= \eta$ for KF, $\tilde{w}$ for KC) and its evolution equation in such a way that the required Kalman matrix M (which $= Z$ or T) satisfies $M_t = E[v_t v_t']$ for all t. This method can be generalized and used to implement, in a NN, certain classes of matrix computations that are difficult or impossible for a NN to perform directly. Suppose we want to implement $M_{t+1} = h(M_t)$ where M is a time-varying matrix and h is such a matrix computation. Our approach consists of constructing – when possible – NN-implementable functions f and g, and a set of vectors $\{v_t(k)|k = 1, \ldots, K\}$ at each time step t, such that $M_{t+1} \approx f(\{v_t(k)\}, M_t)$ and (for each k) $v_{t+1}(k) \approx g(v_t(k), M_t)$, where the approximations become exact in the limit $K \to \infty$. Thus, for example, matrix-times-matrix computations in h may be replaced by matrix-times-vector computations in f

TABLE II

SUMMARY OF THE NN ALGORITHM'S ACTIVITY DYNAMICS AND LEARNING RULES

Function	Equations
Estimation:	
Execution: (activity dynamics; η used also for learning)	$\eta_{t+1} = -y_{t+1} + \tilde{F}(y_t + RZ_t^{-1}\eta_t) + \tilde{u}_t$ where $\eta_t \equiv \hat{y}_t^- - y_t$;
Learning (weight update):	$Z_t \approx E[\eta_t \eta_t']$; e.g., $Z_t = (1-\gamma_Z)Z_{t-1} + \gamma_Z \langle \eta_t^p (\eta_t^p)' \rangle_p$.
System identification: To minimize	$E[(Y_t^{\rm pred} - Y_t)'(Y_t^{\rm pred} - Y_t)]$ with respect to $\tilde{F}$, where $Y_t^{\rm pred} = \tilde{F} Y_{t-1} + \tilde{u}_{t-1}$:
perform gradient descent using for Y either raw data y:	$\tilde{F}_t = \tilde{F}_{t-1} - \gamma_F E[\epsilon_t y_{t-1}']$, $\epsilon_t \equiv \tilde{F}_{t-1} y_{t-1} + \tilde{u}_{t-1} - y_t$;
or Kalman-estimated values $\hat{y}$:	$\tilde{F}_t = \tilde{F}_{t-1} - \gamma_F E[\eta_t \hat{y}_{t-1}']$.
Control:	
Execution (activity dynamics):	$\tilde{u}_t = (-I + T_t^{-1}\tilde{g})\tilde{F}\hat{y}_t$ (as in Table I);
Learning (activity dynamics):	$w_{\tau-1} = -\nu_{\tau-1}^g + \nu_\tau^r + \tilde{F}'(\nu_\tau^g + \tilde{g} T_\tau^{-1} w_\tau)$, where ν_τ^r, ν_τ^g are r.v.'s having zero mean and covariances $\tilde{r}$ and $\tilde{g}$ resp.;
Learning (weight update):	$T_\tau \approx E[w_\tau w_\tau']$.

and/or g.

We have shown[10, cols. 17-18] that this construction can be performed for the algebraic Riccati equation, $X_{t+1} = A'X_tA - A'X_tB(B'X_tB + R)^{-1}B'X_tA$, with $M_{t+1} \equiv B'X_tB + R$. Both the KF and KC problems are described by equations of this form, leading to the NN algorithm described here. How applicable this generalized approach is to other classes of computations is an open question.

IV. THE DERIVED MULTILAYER RECURRENT NN, ITS SIGNAL FLOWS, AND ITS CIRCUIT ARCHITECTURE

The NN algorithms for Kalman estimation and control, and system identification, are realized by a physical NN having four layers of nodes ($D_{\rm y}$ nodes per layer where $D_{\rm y}$ is the dimensionality of the measurement vector y), and wherein a set of operations is performed in a prescribed sequence during each time step t (or τ) of the algorithm. For each 'macro' time step t, the sequence of 'micro time steps' or 'time ticks' is denoted by the letters (a, b, ..., n) in Figure 1. For clarity the set of four layers is depicted twice.

The upper set of four layers shows the signal flows as solid lines for the learning and execution of Kalman estimation (the learning of Z and R, and the computation of $\hat{y}_t$ and $\hat{y}_{t+1}^-$) and for the learning of the system identification matrix $\tilde{F}$, and shows the signal flows as dashed lines for the execution (but not learning) of Kalman control (i.e., the computation of the control signals $\tilde{u}$ given matrices T and $\tilde{g}$). During each 'macro' time step t, the flow is from left to right; the signal output from time tick n of time step t is 'wrapped around' to become the input at tick a of the next time step (incremented by one). (Thus, e.g., $\hat{y}_t$ at time step $t = 3$ is the same activity vector as $\hat{y}_{t-1}$ at the incremented time step $t = 4$.)

The lower set of four layers (all dashed lines) shows the signal flows for the learning of the Kalman control matrix T and the auxiliary matrix $\tilde{g}$. Again the flow is from left to right, for a given value of the time-index τ; now the signal at tick n of time-index τ is wrapped to become the input at tick a of the next time-index value (*decremented* by one).

The four layers are required in order to store five weight matrices and the activity values required for updating them. Four of these matrices are: Z (or Z^{-1}); T (or T^{-1}); R, the covariance of the measurement noise vector; and $\tilde{g}$, a matrix in the J function of KC. (Although $\tilde{g}$ is prescribed by the KC problem, we think of it as being represented indirectly in the NN by an internal random-noise generator of vectors ν_t^g such that $\tilde{g} = E[\nu_t^g (\nu_t^g)']$ and $E[\nu_t^g] = 0$.) Each of these four matrices is learned by a Hebbian rule of the form $M_t = (1-\gamma_M)M_{t-1} + \gamma_M \langle v_t z_t' \rangle$ (or a Hebbian variant of this when it is the matrix M^{-1} that is being learned[7,3]), where $v \equiv z$. Because the activity vector $v = z$ is the same at both ends of the connection matrix whose weights are being updated, it is 'natural' to implement these matrices as sets of *lateral* connections, each within its own layer.

The fifth matrix to be learned (also by a Hebbian rule) is $\tilde{F}$. This matrix and its transpose (needed for KC) are stored as feedback and feedforward weights between a pair of layers.

If Kalman control is not needed, two layers (to store the matrices Z and R) suffice instead of four. The signal flow

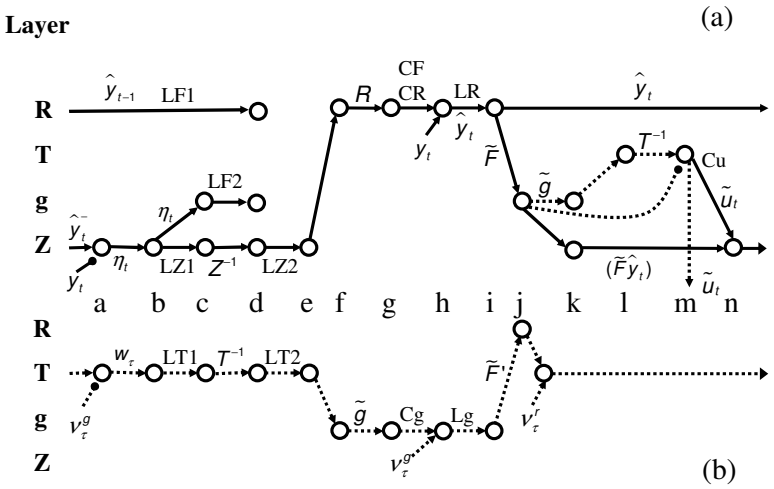

Fig. 1. Layered organization of signal flow required by full NN algorithm for Kalman estimation, system identification, and Kalman control (figure originally published in [5]). (a) Signal flow for system identification and for learning and execution of Kalman estimation (solid links), and for *execution* of Kalman control (dashed links), shown in a four-layer recurrent NN, all at time t. Notation: All unfilled circles within a single layer denote the same set of D_y nodes in that layer, at each time tick a, b, . . ., n. Where two inputs enter an unfilled circle, the input activity vectors are combined; arrowhead inputs are added and filled-circuit inputs are subtracted. (b) Signal flow for *learning* of Kalman control (dashed links), shown separately in the same four-layer NN, all at time index τ of the KC learning process.

then consists only of the bold solid lines in Fig. 1, with minor changes: The layer-Z link between time ticks a and b is labeled 'LF2' (see meaning below), the layer-Z to layer-g links at ticks b-d are removed, $\tilde{F}$ runs from layer R to Z, and $\tilde{u}$ (if any) is provided as external input at tick n, rather than coming from layer T.

We find empirically that, apart from a few minor variations, the NN algorithm (summarized in Table II) appears to require the specific multi-layer signal flow shown in Fig. 1, once we have defined 'layer' as above (i.e., such that when a Hebbian rule has $v \equiv z$, we choose to embody the connection matrix being learned as a set of intra-layer or lateral connections). (See also section VI.)

A. Technical issues

In Figure 1, many links of the signal paths are labeled either by the activity vector they carry at a given time 'tick,' by the matrix that multiplies the activity vector at a given link, or by a label starting with 'L' (for 'learning') or 'C' (for 'cut-point'). Hebbian learning of $\tilde{F}$ (and its transpose) uses the product of the activities at links LF1 and LF2; R learning uses the activity at LR (similarly for $\tilde{g}$ and Lg); Z learning uses the activity at LZ1; and Z^{-1} learning (for the option in which the weight matrix is Z^{-1} rather than Z[9]) uses the activity at LZ2. LT1 and LT2 have the corresponding meanings for T and T^{-1} learning.

The labeled cut-points indicate that, for different phases of operation of the algorithm, the signal flow may start, or be cut off, at different time ticks. Thus, $R = E[n_t n_t']$ is learned by cutting off the signal flow at link CR *and* shutting off external input, so that $y_t \equiv n_t$ (sensor noise) while in this learning mode. Before KF learning starts, $\tilde{F}$ must be approximately learned using the raw measurements y; this is done by cutting signal flow at CF (the same link as CR), so that the 'estimated' value $\hat{y}_t$ between ticks is equal to the raw y_t in this learning mode. After $\tilde{F}$ has been learned sufficiently well, the full circuit is active (no signal cutoff points). During this phase Z is learned, and the learning of $\tilde{F}$ is continued using the estimate $\hat{y}_t$ rather than the raw measurement y_t.

If Kalman control (KC) is to be performed, the KC vector T is learned by performing, at a given time step $t = t_0$, a sequence of steps in which τ is decremented from N to t_0, and in which the signal flow proceeds as in the lower panel of Fig. 1 for each decremented τ in turn. Before any T learning is done, however, the matrix $\tilde{g}$ is learned by cutting off the signal flow at link Cg, so that $\tilde{g}$ learns an approximation of $E[\nu_t^g (\nu_t^g)']$ by Hebbian learning at link Lg.

B. The derived neural circuit

For each link in the signal flow Fig. 1, we provide a directed link joining the corresponding layers in the circuit diagram of Fig. 2. Intra-layer processing (multiplication by a matrix of lateral connections) is indicated by the undirected horizontal link within each layer. The result is a four-layer recurrent NN having certain distinctive features, discussed in Section VI.

V. Numerical simulation of Kalman filtering

We illustrate the NN algorithm's results for KF and KC in Figure 3. Here we assume that R has already been learned, and that $N_{\rm p} = 300$ independently evolving input vectors are being measured at each time step. The plant and measurement spaces are two-dimensional, and the plant, measurement, and control processes are defined as follows (refer to Table I, left column): F, H, and B are 2-d counterclockwise rotations about the coordinate origin by 15^o, 50^o, and 20^o respectively; the plant and measurement

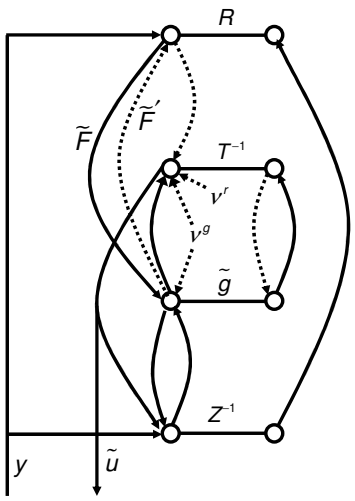

Fig. 2. NN circuitry required to enable the signal flow of Fig. 1 (figure originally published in [5]). Within each layer, the pair of circles denotes the set of $D_{\rm y}$ nodes, and the labeled line joining them denotes the weight matrix (or its inverse) of its lateral connections. Dashed lines denote connections that are only involved in KC learning.

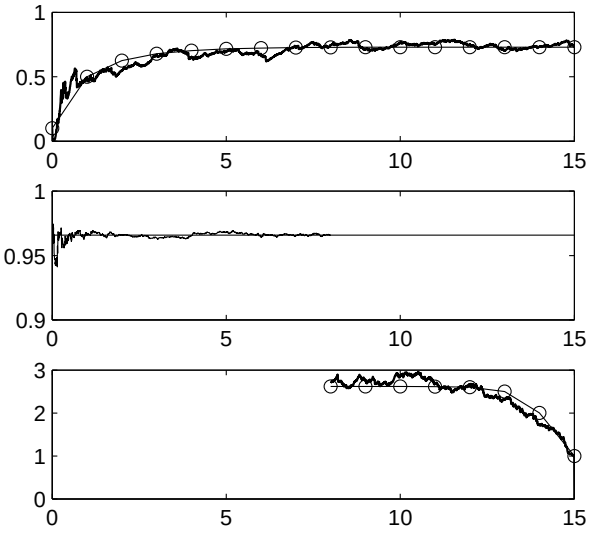

Fig. 3. Example of classical and neural Kalman filter and control learning, and neural system identification. See text.

noise covariance matrices are $Q = 10^{-5}I$ and $R = 10^{-4}I$; and the control matrices are $\tilde{g} = \tilde{r} = I$.

At each time step, Z_t is (for more efficient learning) incrementally updated using each of the $N_{\rm p}$ vectors in turn, instead of being batch-updated as the example in Table II would imply. That is, for each $t = 1, \cdots$, given Z_{t-1}, set $Z_t^0 \equiv Z_{t-1}$; then, for $p = 1, \cdots, N_{\rm p}$, update[6] $Z_t^p = (1 - \gamma_Z)Z_t^{p-1} + \gamma_Z \eta_t^p (\eta_t^p)'$; then set $Z_t \equiv Z_t^{N_{\rm p}}$. The (1,1) component of $(I - HK_t) \equiv R(Z_t^p)^{-1}$ is plotted vs. $(t - 1 + p/N_{\rm p})$, as the jagged line in Fig. 3, top panel. The same component of $(I - HK_t)$ for the classical KF solution are plotted for comparison as circles (connected by line segments) at integer values of t.

The (1,1) component of the learned $\tilde{F}$ matrix is similarly incrementally updated and plotted in Fig. 3, middle panel, for $t \leq 8$. The horizontal line denotes the true value of $\tilde{F}_{11}$. In this example, $\tilde{F}$ is updated from the outset using the $\hat{y}^-$ value predicted using the current Z, and Z is updated using the current $\tilde{F}$, even though both matrices are arbitrarily initialized and therefore do not at first approximate the true $\tilde{F}$ and optimal Z. Thus the two learning processes 'bootstrap' each other. Note that $I - HK$ decreases from 0.1 at $(t = 1, p = 1)$ to ≈ 0.003 at $(t = 1, p = 10)$ (not readily seen in the figure), automatically responding to the fact that the initially inaccurate model should be ignored (corresponding to $I - HK = 0$), and the y measurement used instead, to estimate $\hat{y}$. As the learned $\tilde{F}$ converges to its true value, the optimal model contribution to the estimate increases, and the learned $I - HK$ thereby starts to converge to its asymptotic value.

In this example, the control process is turned on at $t = 8$, after the $\tilde{F}$ and Z values have reached their asymptotic values (apart from random fluctuations). At that point (refer to Fig. 3, bottom panel), the KC time variable τ is decremented for each of $N_{\rm p}$ vectors w_τ^p (see Table II), implementing the NN learning of the matrix $\tilde{T}_\tau$ for $\tau = 14, 13, \ldots, 8$. Here $\tilde{T}$ is learned incrementally using the procedure described above for Z, but with a constant learning rate $\gamma_T = 1/N_{\rm p}$. The values of $\tilde{F}$ and Z at $t = 8$ are used for the KC calculation. The $\tilde{T}_\tau$ matrices that are learned after incrementing through all p at each τ are stored. Then, for $t = 8, 9, \ldots, 14$, the stored $\tilde{T}_t$ is used (for all p) to compute u_t^p and thus drive the execution of the control process, while $\tilde{F}$ and Z learning continue. Figure 3 (bottom panel) shows the (1,1) component of the incrementally learned $\tilde{T}_\tau^p$ plotted vs. $(\tau + 1 - p/N_{\rm p})$ (jagged line), and for comparison the classical KC result (denoted by circles at integer τ connected by line segments).

VI. Constraints on the Choice of Signal Flows and NN Circuitry

Both the signal flows and the NN architecture and circuitry appear to be constrained (apart from small variations) by the requirements of the Kalman neural algorithm and of local (in our case, Hebbian) learning rules; that is, they do not appear to represent merely one among a large number of disparate possible choices of architecture or signal flow. Among the constraints are the following:

(a) Each of the matrices R, $\tilde{g}$, Z (or Z^{-1}), and T (or T^{-1}) is learned by a Hebb rule that contains a product of the form vz' where the source and target activity vectors v and z are equal; each matrix is thus represented by a set of lateral connections within a distinct layer.

[6]The learning rates for Z and $\tilde{F}$ are preferably allowed to be time-varying for efficient learning, using the adaptive method of [11]. In that reference's notation, the values of the rate control parameters, which we have made no attempt to optimize, are $\{\alpha, \beta, \gamma, \delta\} = \{0.5, 30, 0.05, 0.1\}$ for Z, and $\{0.1, 3, 0.05, 0.04\}$ for $\tilde{F}$.

(b) For local (Hebbian) learning, each activity vector must be present at the input to the connection matrix that is learned using that vector; thus η must be present at the input to the Z (or Z^{-1}) connections, and similarly for the (vector; matrix) pairs $(w; T$ or $T^{-1})$, $(y \equiv n; R)$ during R learning mode, and $(\nu^g; \tilde{g})$ during $\tilde{g}$ learning mode.

(c) For Hebbian learning of $\tilde{F}$, activities $\hat{y}_{t-1}$ and η_t must be present simultaneously at the two ends of the $\tilde{F}$ matrix. Thus $\hat{y}_{t-1}$ must be held as the activity of one set of nodes in layer R until η_t has been computed at layer Z. $\tilde{F}$ is *updated* at the time indicated by links LF1 and LF2, but is *used* later in the signal flow, at the link labeled $\tilde{F}$ (Fig. 1).

(d) Matrix $\tilde{F}$ and its transpose $\tilde{F}'$, required for Kalman control, are learned using the same Hebbian rule, and thus must connect the same two layers (in opposite directions).

(e) The measurement vector y_t is required twice as input: to layer Z, where it contributes to η_t, and to layer R, where it combines with $RZ^{-1}\eta_t$ to yield $\hat{y}_t$.

Thus the arrangement shown in Figs. 1 and 2 is not an arbitrary way of laying out the NN algorithms we have derived; rather, the four-layer organization and its signal flows appear to be substantially determined (apart from small variations) by the algorithms and the requirements of a NN implementation.

VII. Comparison with biology

A. Background

It has been proposed by many workers[1,12-14] that mammalian cerebral cortex is characterized, at the $50 \sim 100\mu$ horizontal scale, by an arrangement of cells and local interconnections – sometimes called the minicolumn, or local cortical circuit (LCC), or canonical microcircuit – that is substantially similar (although with significant and well-known variations) across a variety of cortical areas subserving vision, hearing, motor control, and other functions. These findings have suggested that the LCC might perform a set of core functions, as yet unknown, that underlie the variety of observed higher-level area-specific functions. Various proposals have explored possible connections among visual processing, learning, and development[15]; between sensory and motor function[16]; and especially between the processing of 'bottom-up' sensory input and 'top-down' model-driven expectations. The latter approach has made use of Bayesian inference and generative models, and various NN methods are motivated by, approximate, or perform a portion of the Bayesian inference process[17-27]. Kalman filtering is, given certain assumptions, an exactly solvable special linear case of Bayesian inference.

In particular, I consider it a plausible and attractive hypothesis to think of the LCC's functions as including prediction, the estimation or inference of missing or noisy sensory data, and the goal-driven generation of control signals, albeit using nonlinear methods that enable the discovery of higher-level sensory and motor-control features, and that go beyond (linear) Kalman estimation and control, and also beyond (nonlinear) 'extended Kalman filtering.' In view of the constraints that our algorithm places on the NN's design (previous section), this hypothesis motivates a comparison between the derived NN architecture and that of the LCC.

B. Resemblances to the biological LCC, and caveats

We compare diagrammatically the interlaminar signal flow of the KPC NN with the putative principal signal flows[14] of the LCC. The signal flow for Kalman estimation (learning and execution) and Kalman control (execution only) (Fig. 1a) may be schematized, considering layers g and T as a unit, as Dgm. D1:

$$
\begin{array}{ccccccccc}
\{ & & \mathrm{R} & & \} & \rightarrow & (\mathrm{g,T}) & \rightarrow & \mathrm{Z} \quad \rightarrow \mathrm{R} \\
 & \uparrow & & \downarrow & & & \downarrow & & \uparrow \\
 & y & & \hat{y} & & & \tilde{u} & & y \quad \textbf{[D1]}
\end{array}
$$

$$
\begin{array}{ccccccccc}
 & 4 & \rightarrow & 2/3 & & \rightarrow & 5 & \rightarrow & 6 \quad \rightarrow 4 \\
 & \uparrow & & \downarrow & & & \downarrow & & \uparrow \\
 & \mathrm{in} & & \mathrm{out}_S & & & \mathrm{out}_M & & \mathrm{in} \quad \textbf{[D2]}
\end{array}
$$

(The KC learning of Fig. 3b adds a g $\rightarrow$ R path.) By comparison, Gilbert's proposal[14] for the principal LCC signal flow among the layers 6, 5, 4, and 2/3 is shown in Dgm. D2 (above). More recent work is consistent with, and expands upon, this basic flow[12,13,28]. The similarities are consistent with a rough correspondence between NN layer R and LCC layers 4 and 2/3, between NN layer Z and LCC layer 6, and between NN layers (g,T) and LCC layer 5 (important in motor processing).

These and other resemblances are *consistent* with the view – although they do not imply – that the LCC may be performing prediction, filtering, and control functions (albeit involving nonlinear operations including the discovery of higher-level features, use of context, etc.). There are several important caveats that prevent one from making any stronger claim than that of consistency, and from expecting any highly-detailed correspondence. For example: (a) We have considered only one type of NN representation – real-valued activities with linear-combination processing units[7] – rather than other coding schemes and neuronal dynamics (e.g., population codes, spike codes, sparse codes, phase-locking schemes, etc.). (b) Biological prediction and control are more sophisticated than KF or EKF, e.g., involving Bayesian inference or some approximation thereof. (c) Current knowledge of detailed LCC connectivity and of signal flows (and their sequencing) within the circuit remains incomplete. Owing to space limitations, we must refer the interested reader to [5] for a more extensive discussion of both the resemblances and caveats.

[7]We could have used units having sigmoid nonlinearities, but the nonlinearity appears to offer no advantage for implementing (linear) KF and KC. Also, the fact that the 'activities' in our artificial NN can be of either sign, and that weights can change sign, is inessential; the units and weights can be replaced by more biologically plausible ones without affecting the results.

VIII. Conclusion

We have shown that Kalman estimation (including prediction), Kalman control, and system identification can be fully implemented within an artificial neural network. The NN's simultaneous learning of KF, KC, and system identification is illustrated by a numerical example. The resulting network is a multilayer recurrent NN that may be useful for engineering applications, and that also bears certain resemblances to the putative 'local circuit' of mammalian cerebral cortex. The possible connection with neuroscience fits into a broader context of ongoing NN research by many workers, exploring a range of neural coding strategies and a range of computational tasks in prediction and control (e.g., Bayesian inference and generative models), as well as experimental work to elucidate both the functional connectivity and signal flows in the LCC. It will be important to discover which types of computational tasks are associated with specific architectural features in NNs and, if so, whether such mappings have useful implications for understanding biological neural function.

Such continuing interaction between theory and experiment may not only contribute to elucidating core aspects of cortical function, but may also lead to insights into new methods for nonlinear estimation and control.

References

[1] Mountcastle, V. B. (1998). *Perceptual neuroscience: the cerebral cortex* (Harvard Univ. Press).

[2] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Trans. ASME – J. Basic Eng., 82*, 35-45.

[3] Rao, R. P. N. & Ballard, D. H. (1997) Dynamic model of visual recognition predicts neural response properties in the visual cortex. *Neural Computation, 9*, 721-763.

[4] Rao, R. P. N. (1999). An optimal estimation approach to visual perception and learning. *Vision Res., 39*, 1963-89.

[5] Linsker, R. (2008). Neural network learning of optimal Kalman prediction and control. *Neural Networks, 21*, 1328-1343.

[6] Szirtes, G., Póczos, B., & Lőrincz, A. (2005). Neural Kalman-filter. *Neurocomputing, 65-66*, 349-355.

[7] Szita, I. & Lőrincz, A. (2004). Kalman filter control embedded into the reinforcement learning framework. *Neural Computation, 16*, 491-499.

[8] Linsker, R. (1992). Local synaptic learning rules suffice to maximize mutual information in a linear network. *Neural Computation, 4*, 691-702.

[9] Linsker, R. (2005). Improved local learning rule for information maximization and related applications. *Neural Networks, 18*, 261-265.

[10] Linsker, R. (2008) Neural networks for prediction and control. *U. S. Patent 7,395,251*.

[11] Murata, N., Müller, K.-R., Ziehe, A., & Amari, S.-i. (1997). Adaptive on-line learning in changing environments. *Adv. Neural Info. Proc. Systems, 9*, 599-605.

[12] Callaway, E. M. (1998). Local circuits in primary visual cortex of the macaque monkey. *Annu. Rev. Neurosci., 21*, 47-74.

[13] Douglas, R. J. & Martin, K. A. C. (2004). Neuronal circuits of the neocortex. *Annu. Rev. Neurosci., 27*, 419-51.

[14] Gilbert, C. D. (1983). Microcircuitry of the visual cortex. *Annu. Rev. Neurosci., 6*, 217-247.

[15] Grossberg, S. & Williamson, J. R. (2001). A neural model of how horizontal and interlaminar connections of visual cortex develop into adult circuits that carry out perceptual grouping and learning. *Cerebral Cortex, 11*, 37-58.

[16] Poggio, T. & Bizzi, E. (2004). Generalization in vision and motor control. *Nature, 431*, 768-74.

[17] Hinton, G. E. & Ghahramani, Z. (1997). Generative models for discovering sparse distributed representations. *Philos. Trans. R. Soc. London, B, 352*, 1177-90.

[18] Lewicki, M. S. & Sejnowski, T. J. (1997). Bayesian unsupervised learning of higher order structure. *Adv. Neural Info. Proc. Systems, 9*, 529-35.

[19] Rao, R. P. N. & Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience, 2*, 79-87.

[20] Lee, T. S. & Mumford, D. (2003). Hierarchical Bayesian inference in the visual cortex. *J. Opt. Soc. Am. A, 20*, 1434-48.

[21] Rao, R. P. N. (2004). Bayesian computation in recurrent neural circuits. *Neural Computation, 16*, 1-38.

[22] George, D. & Hawkins, J. (2005). A hierarchical Bayesian model of invariant pattern recognition in the visual cortex. *Proc. 2005 IEEE Int. Joint Conf. Neural Networks, 3*, 1812-17.

[23] Rao, R. P. N. (2005). Bayesian inference and attentional modulation in the visual cortex. *NeuroReport, 16*, 1843-48.

[24] Todorov, E. (2005). Stochastic optimal control and estimation methods adapted to the noise characteristics of the sensorimotor system. *Neural Computation, 17*, 1084-1108.

[25] Yu, A. J. & Dayan, P. (2005). Inference, attention, and decision in a Bayesian neural architecture. *Adv. Neural Info. Proc. Systems, 17*, 1577-84.

[26] Zemel, R. S., Huys, Q. J. M., Natarajan, R., & Dayan, P. (2005). Probabilistic computation in spiking populations. *Adv. Neural Info. Proc. Systems, 17*, 1609-16.

[27] Hinton, G. E., Osindero, S. & Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation, 18*, 1527-1554.

[28] Raizada, R. D. S. & Grossberg, S. (2001). Context-sensitive binding by the laminar circuits of V1 and V2: A unified model of perceptual grouping, attention, and orientation contrast. *Visual Cognition, 8*, 431-466; see refs. cited in Table 1.