Least Squares and Kalman Filtering on Forney Graphs

Hans-Andrea Loeliger, ISI, ETH Zürich

To appear in the Forney Festschrift

Revised, October 18, 2001

Abstract: General versions of Kalman filtering and recursive least-squares algorithms are derived as instances of the sum(mary)-product algorithm on Forney-style factor graphs.

1 Introduction

Factor graphs [3][6] are a unifying framework for a wide variety of system models, and the generic sum(mary)-product algorithm, which works by message passing in the factor graph, subsumes a wide variety of algorithms in coding, signal processing, and artificial intelligence [3]. It was pointed out in [3] that Kalman filtering can also be viewed as an instance of the sum-product algorithm, but this was explicitly demonstrated only for the scalar case; in the present paper, we discuss the vector case. We also demonstrate the equivalence of Kalman filtering and general recursive least-squares algorithms in this context. (For traditional state space models, this equivalence was shown in [4]).

In [2], Forney introduced a variation of factor graphs with a number of attractive properties. We will use these Forney graphs rather than the original factor graphs of [3].

The main result of the present paper are Tables 1 and 2, which state the message update rules for the building blocks of the classical linear state space models. Of course, these rules are just reformulations of the well established equations of Kalman filtering (e.g., [1]). Nevertheless, the whole paper is devoted to explain and to prove these tables.

The paper is structured as follows. Forney-style factor graphs are introduced in Section 2. In Section 3 some pertinent properties of Gaussian distributions are reviewed and the equivalence of the min-sum and the sum-product algorithm for Gaussian networks is pointed out. Kalman filtering as an instance of the sum(mary)-product algorithm is discussed in Section 4. The proofs of the message update rules of Tables 1 and 2, as well as some comments on these rules, are given in Section 5. Some conclusions are offered in Section 6. There is also an appendix with some background material in linear algebra, especially on the pseudo-inverse and on nonnegative definite matrices.

The following notation will be used. If z is a complex number, then $\overline{z}$ is its conjugate complex. If A is a matrix or a vector, then A^T is its transpose, $A^H \triangleq \overline{A^T}$, and $A^\#$ is the

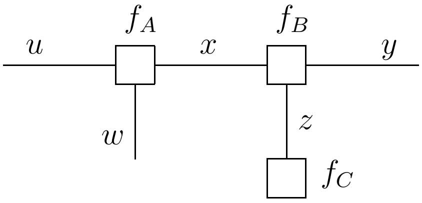


Figure 1: A Forney-style factor graph (FFG).

Moore-Penrose pseudo-inverse (see the appendix). The symbol "$\propto$" denotes equality of functions up to a scale factor. All vectors all column vectors.

2 Forney Graphs

A Forney-style factor graph (FFG) or "normal graph" [2] represents a factorization of a function of several variables. For example, assume that some function $f(u,w,x,y,z)$ can be factored as

$$f(u,w,x,y,z) = f_A(u,w,x)f_B(x,y,z)f_C(z). \tag{1}$$

This factorization is expressed by the graph of Fig. 1. In general, an FFG consists of nodes, edges, and "half edges", where "half edges" are connected to only one node. The rules are as follows:

- There is a node for every factor.
- There is an edge (or half edge) for every variable.
- The node representing some factor g is connected with the edge (or half edge) representing some variable x if and only if x is an argument of g.

Implicit in these rules is the assumption that no variable appears in more than two factors. We will see below that this condition is far less restrictive than might appear at first sight.

The factors of the factorization expressed by the FFG are also called *local functions;* the overall function (i.e., the product of all local functions) is called the *global function.*

We will now rephrase some basic facts about factor graphs for FFGs; for more details, see [3] and [2]. We will first assume that all variables take values in finite sets; the modifications for continuous variables are given at the end of this section.

In probability theory, factorizations of joint probability measures are expressions of independence. E.g., let X, Y, and Z be discrete random variables with joint probability mass function $p(x,y,z)$. Then X,Y,Z form a Markov chain if and only if $p(x,y,z)$ can be factored as

$$p(x,y,z) = p(x)p(y|x)p(z|y). \tag{2}$$

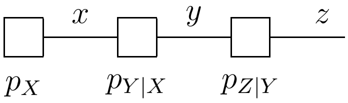


Figure 2: FFG of a Markov chain.

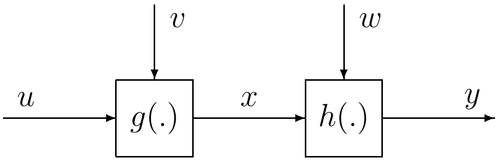


Figure 3: A block diagram.

This factorization is shown in Fig. 2. Upon removal of the edge y, the graph falls into two disconnected components, with x and z in different components, which expresses the conditional independence of X and Z given Y. It is easy to see that this generalizes to any FFG of a joint probability mass function: conditioned on the variables in any cut set of the graph, the variables in the two resulting components are independent.

A block diagram as in Fig. 3 may also be viewed as an FFG. A function block $x = g(u, v)$ in the block diagram is then interpreted as representing the factor $\delta(x - g(u, v))$, where δ is the Kronecker delta function. When viewed as an FFG, the block diagram of Fig. 3 thus represents the function

$$\delta\big(x - g(u, v)\big)\,\delta\big(y - h(x, w)\big) = \begin{cases} 1, & \text{if } x = g(u, v) \text{ AND } y = h(x, w) \\ 0, & \text{else.} \end{cases} \tag{3}$$

In other words, the global function evaluates to 1 if and only if the values of all variables are consistent with the equations of the block diagram. Note that the arrows in the block diagram have no influence on its interpretation as an FFG.

As illustrated by these examples, an FFG can be used to express the structure of a "system" or "model". In this context, the domain of the global function f is called the *configuration space.* A *configuration* is an element of the configuration space, i.e., a particular assignment of values to all variables. A configuration ω is *valid* if $f(\omega) \neq 0$.

In a block diagram, we usually find also branching points as in Fig. 4 (left). In an FFG, such branching points must be treated as factor nodes on their own, as is illustrated in Fig. 4 (right). In doing so, there arise new variables (x_1 and x_2 in Fig. 4) and a new factor

$$f_=(x, x_1, x_2) \stackrel{\triangle}{=} \delta(x - x_1)\,\delta(x - x_2). \tag{4}$$

Note that, in every valid configuration, the new auxiliary variables have the same value as the original variable. By this device of variable "cloning", it is always possible to enforce the condition that a variable may appear in at most two factors (local functions).

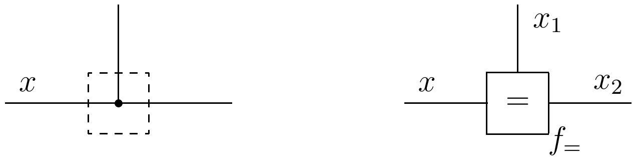


Figure 4: Branching point (left) becomes a replication node (right).

The fact that ordinary block diagrams can be viewed as FFGs is one of the advantages of FFGs over factor graphs as defined in [3]. With this comes also the advantage of a clear distinction between external ("visible") variables and internal ("latent" or "state") variables: the former are represented by half edges, the latter by normal edges. Moreover, the operations of dividing a system into subsystems ("tearing") and of "zooming" into the interior of some subsystem — both central to Willems' system theory [7] — are naturally expressed in an FFG (cf. Figures 5 and 6).

Another attraction of FFGs is that the sum-product algorithm [3][6] takes on a particulary simple form. Two messages are transmitted along each edge, one in each direction. (In practice, it often happens that only a subset of all these messages is actually needed.) Each message is, or represents, a *function* of the variable associated with that edge. Consider a node that represents some factor $f(x_1, \ldots, x_n)$. The message $\mu_{f \to x_k}$ out of this node along the edge x_k is the function

$$\mu_{f\to x_k}(x_k) = \sum_{x_1} \cdots \sum_{x_{k-1}} \sum_{x_{k+1}} \cdots \sum_{x_n} f(x_1, \ldots, x_n) \mu_{x_1 \to f}(x_1) \cdot \ldots \\ \cdot \mu_{x_{k-1}\to f}(x_{x-1}) \mu_{x_{k+1}\to f}(x_{k+1}) \cdot \ldots \cdot \mu_{x_n \to f}(x_n), \quad (5)$$

where $\mu_{x_j \to f}$ is the message incoming on edge x_j. In words, the message $\mu_{f \to x_k}$ is the product of f and all messages towards f along all edges except x_k, summed over all variables except x_k. In practice, the update rule (5) is often modified to include a scale factor.

The messages in the graph are computed, or iteratively recomputed, according to some schedule. As is well known, if the graph is cycle free, the (final) message out of some half edge representing some variable x is the marginal function

$$\mu(x) \triangleq \sum_{\omega:\, x \text{ fixed}} f(\omega), \quad (6)$$

where the sum goes over all configurations ω with fixed x; for details see [3].

In this paper, we will be primarily interested in real or complex variables, or variables that are real or complex vectors. In this case, the Kronecker delta in equations such as (3) and (4) should be replaced by the Dirac delta and the summation in (5) and (6) should be replaced by integration.

3 Gaussian Distributions and Quadratic Cost Functions

In Kalman filtering, the factors and messages are of the form

$$f(x) = e^{-q(x)} \tag{7}$$

with

$$\begin{aligned} q(x) &\stackrel{\triangle}{=} (x-m_x)^H W_x (x-m_x) + c_x & (8)\\ &= x^H W_x x - 2\operatorname{Re}\big(x^H W_x m_x\big) + m_x^H W_x m_x + c_x, & (9) \end{aligned}$$

where x is a real or complex vector, where W_x is a nonnegative definite $n \times n$ matrix, and where the constant $c_x \in \mathbb{R}$ amounts to a scale factor in (7) which can often be ignored. The case where all quantities in (8) are real-valued and $q(.)$ is a function $\mathbb{R}^n \rightarrow \mathbb{R}$ will be referred to as "the real case"; the case where $q(.)$ is a function $\mathbb{C}^n \rightarrow \mathbb{R}$ will be referred to as "the complex case".

If W_x is positive definite, (7) may be viewed as a multi-dimensional Gaussian probability density with mean m_x. In the real case, the corresponding covariance matrix is $\frac{1}{2}W_x^{-1}$ and the appropriate scale factor is $e^{-c_x} = \sqrt{\det(W_x)/\pi^n}$; in the complex case, the covariance matrix is W_x^{-1} and the scale factor is $e^{-c_x} = \det(W_x)/\pi^n$. Note that the integral $\int_{-\infty}^{\infty} e^{-q(x)+c_x}\,dx$ depends on W_x but neither on m_x nor on c_x.

The sum of terms of the form (8) is again of that form (cf. the appendix), and thus the product of terms of the form (7) is also of the form (7). Moreover, integrating a function of the form (7) over some of the variables also preserves this form. Indeed, if $q(x,y)$ is the quadratic form

$$q(x,y) \stackrel{\triangle}{=} \Big((x-m_x)^H,\, (y-m_y)^H\Big) \begin{pmatrix} W_{1,1} & W_{1,2} \\ W_{2,1} & W_{2,2} \end{pmatrix} \begin{pmatrix} x-m_x \\ y-m_y \end{pmatrix}, \tag{10}$$

then, considered as a function of x with parameter y, $q(x,y)$ is of the form (9) with $W_x = W_{1,1}$ and with both m_x and $c_x = \min_x q(x)$ depending on y. Assuming that $W_{1,1}$ is positive definite, we then have

$$\begin{aligned} \int_{-\infty}^{\infty} e^{-q(x,y)} dx &= e^{-c_x(y)} \int_{-\infty}^{\infty} e^{-\big(q(x,y)-c_x(y)\big)} dx & (11)\\ &\propto e^{-c_x(y)} & (12)\\ &= e^{-\min_x q(x,y)} & (13) \end{aligned}$$

because the integral on the right-hand side of (11) does not depend on y.

It is often convenient to view Dirac delta functions as limits of Gaussian distributions: in the real case as

$$\delta(x) = \lim_{\beta\to\infty} \frac{\sqrt{\beta}}{(2\pi)^{n/2}} e^{-\frac{\beta}{2}\|x\|^2} \tag{14}$$

and in the complex case as

$$\delta(x) = \lim_{\beta \to \infty} \frac{\beta}{\pi^n} e^{-\beta \|x\|^2}. \tag{15}$$

If all factors in some FFG are of the form (7), then both (5) and (6) (with summation replaced by integration) are of the form (13) and yield results of the form (7). In this case, all computations, and indeed the FFG itself, may be interpreted in the logarithmic domain. The FFG then represents a "global" quadratic cost function $\varphi(\omega) = -\log f(\omega)$ of the form (8) as the sum of "local" quadratic cost functions of the same form, the messages are also quadratic cost functions of this form, and the sum-product rule (5) becomes the min-sum rule

$$\begin{aligned} q_{f \to x_k}(x_k) = \min_{x_1} \ldots \min_{x_{k-1}} \min_{x_{k+1}} \ldots \min_{x_n} \Big(\varphi(x_1, \ldots, x_n) + q_{x_1 \to f}(x_1) + \ldots \\ + q_{x_{k-1} \to f}(x_{x-1}) + q_{x_{k+1} \to f}(x_{k+1}) + \ldots + q_{x_n \to f}(x_n) \Big) + c, \end{aligned} \tag{16}$$

with $\varphi(x_1, \ldots, x_n) \triangleq -\log f(x_1, \ldots, x_n)$, $q_{f \to x_k}(.) \triangleq -\log \mu_{f \to x_k}(.)$, $q_{x_j \to f}(.) \triangleq -\log \mu_{x_j \to f}(.)$, and where $c \in \mathbb{R}$ is a normalizing constant that corresponds to a scale factor in (5) and may be chosen freely. The sum-product algorithm thus becomes the min-sum algorithm [3]; if the graph has no cycles, the final message $q(x)$ out of some half edge x is

$$q(x) = \min_{\omega \colon x \text{ fixed}} \varphi(\omega) + \text{const.} \tag{17}$$

In other words, the marginalization (6) is equivalent to the least-squares problem (17) and the sum-product algorithm becomes the min-sum algorithm, which, in this case, is a general least-squares algorithm.

If the graph has cycles, the min-sum algorithm may fail to converge; however, if it converges, Weiss and Freeman [5] have shown that the mean vector of the message $q(x)$ is correct, i.e., it agrees with the mean vector of the quadratic form on the right-hand side of (17).

4 Least Squares and Kalman-Filtering on an FFG

Kalman filtering may be viewed as the sum-product algorithm applied to the state space model of Fig. 5 with composite nodes as in Fig. 6:

$$\begin{aligned} x[k] &= Ax[k-1] + Bu[k] \qquad (18) \\ y[k] &= Cx[k] + w[k], \qquad (19) \end{aligned}$$

where for $k \in \mathbb{Z}$, $u[k]$, $w[k]$, $x[k]$, and $y[k]$ are real or complex vectors and where A, B, and C are real or complex matrices of appropriate dimensions.

In the traditional setup, it is assumed that $y[.]$ is observed and that both $u[.]$ and $w[.]$ are white Gaussian noise. In its most narrow sense, Kalman filtering is then only the

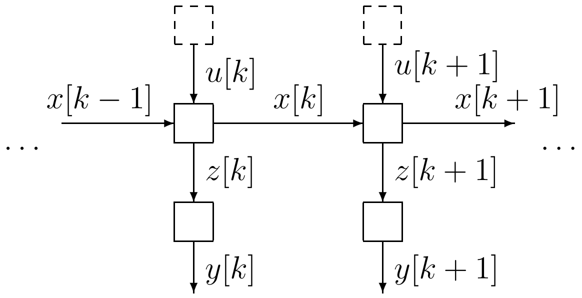


Figure 5: State space model.

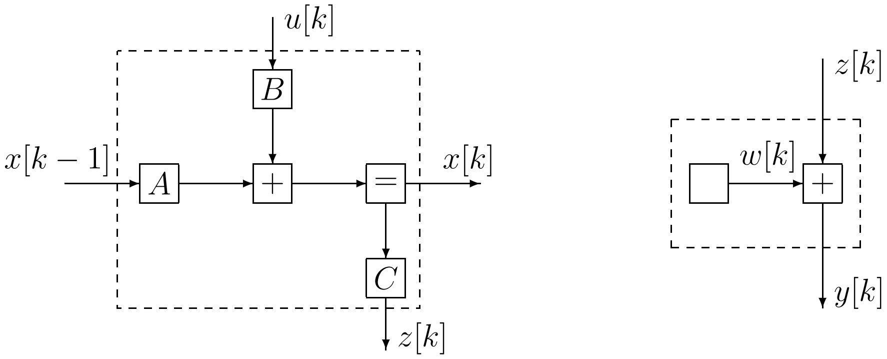


Figure 6: Details of linear state space model.

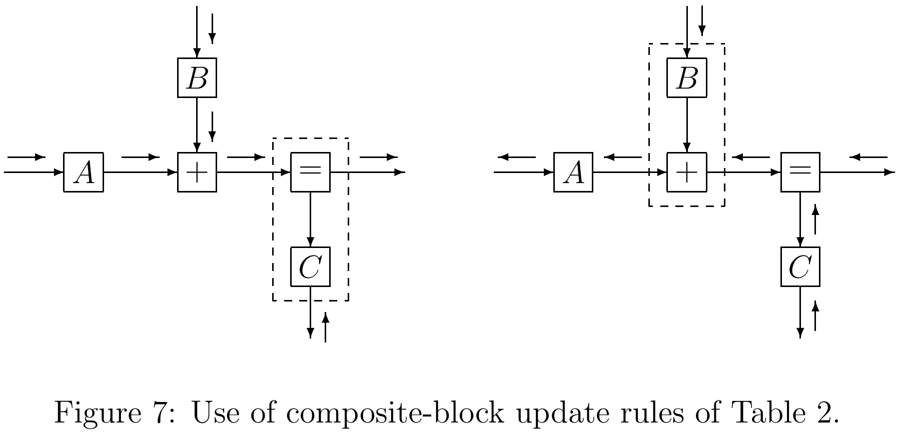


Figure 7: Use of composite-block update rules of Table 2.

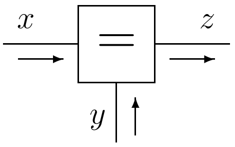
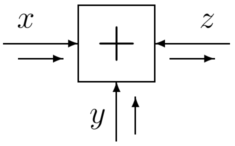
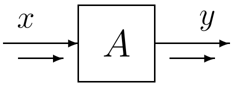
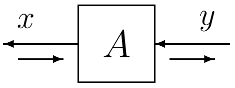
1	 $\delta(x-y)\delta(x-z)$	$m_z = \big(W_x + W_y\big)^{\#}\,(W_x m_x + W_y m_y)$ $W_z = W_x + W_y$ $V_z = V_x\big(V_x + V_y\big)^{\#}\,V_y$
2	 $\delta(x+y+z=0)$	$m_z = -m_x - m_y$ $V_z = V_x + V_y$ $W_z = W_x\big(W_x + W_y\big)^{\#}\,W_y$
3	 $\delta(y-Ax)$	$m_y = Am_x$ $V_y = AV_xA^H$ Problem with W_y if A has not full row rank!
4	 $\delta(x-Ay)$	$m_y = \big(A^H W_x A\big)^{\#} A^H W_x m_x$ $W_y = A^H W_x A$ If A has full row rank: $m_y = A^H\big(AA^H\big)^{-1} m_x$

Table 1: Update rules for messages consisting of mean vector m and covariance matrix V or $W = V^{-1}$.

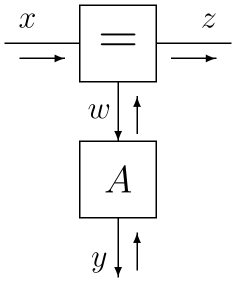
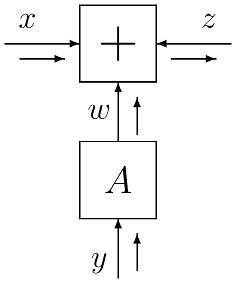
5		$m_z = m_x + V_x A^H G\left(m_y - Am_x\right)$ $V_z = V_x - V_x A^H G A V_x$ with $G \triangleq \left(V_y + AV_x A^H\right)^{-1}$
6		$m_z = -m_x - Am_y$ $W_z = W_x - W_x A H A^H W_x$ with $H \triangleq \left(W_y + A^H W_x A\right)^{-1}$

Table 2: Update rules for composite blocks.

forward sum-product recursion though the graph of Fig. 5 (cf. Fig. 7 left) and yields the a posteriori probability distribution of the state $x[k]$ given the observation sequence $y[.]$ up to time k. By computing also the backwards messages (cf. Fig. 7 right), the a posteriori probability of all quantities given the whole observation sequence $y[.]$ may be obtained.

More generally, Kalman filtering amounts to the sum-product algorithm on any FFG (or part of an FFG) that consists of the linear building blocks listed in Table 1. As discussed in Section 3, any message along some edge x represents a quadratic cost function of the form (8) or, equivalently, a function of the form (7); for the actual computation, each such message consists of a mean vector m_x and a nonnegative definite "cost" matrix W_x or its inverse $V_x = W_x^{-1}$. As pointed out in Section 3, Kalman filtering may thus be viewed as, and is equivalent to, a general least-squares algorithm.

The rules for the computation of such messages are given in Table 1. As only one of the two messages along any edge, say x, is considered, the corresponding mean vectors and matrices are simple denoted m_x, W_x, etc.. The derivation of these rules from the min-sum update rule (16) is given in the next section.

In general, we allow the matrices W or V to be nonnegative definite, which allows to express certainty in V and complete ignorance in W. However, whenever the inversion of such a matrix is required, it had better be positive definite; in practice, this may be achieved by the innocent device of adding a small multiple of an identity matrix.

The direct application of the update rules in Table 1 may lead to frequent matrix inversions. A key observation in Kalman filtering is that the inversion of large matrices can often be avoided. In the FFG, such simplifications may be achieved by using the update rules for the composite blocks given in Table 2. E.g., the vectors $u[k]$ and $z[k]$ in Fig. 5 have usually much smaller dimensions than the state vector $x[.]$; in fact, they are often scalars. By working with composite blocks as in Fig. 7, the forward recursion (left in Fig. 7) using the covariance matrix $V = W^{-1}$ then requires no inversion of a large matrix and the backward recursion (right in Fig. 7) using the cost matrix W requires only one such inversion for each discrete time index.

5 Derivation of Update Rules

We now derive the update rules of Tables 1 and 2 from the min-sum rule (16). As only one of the two messages along any edge, say x, is considered, the corresponding quadratic cost function will be denoted simply by $q_x(x) = (x - m_x)^H W_x (x - m_x)$. Frequent use will be made of facts stated in the appendix.

5.1 Equality Constraint

Rule 1 of Table 1 is proved as follows. Assume that both W_x and W_y are nonnegative definite. According to the min-sum rule (16), we have for $\beta \to \infty$

$$\begin{aligned} q_z(z) &= \min_x \min_y \Big(q_x(x) + q_y(y) + \beta \left(\|x-y\|^2 + \|x-z\|^2 \right) \Big) & (20) \\ &= q_x(z) + q_y(z) & (21) \\ &= (z - m_x)^H W_x (z - m_x) + (z - m_y)^H W_y (z - m_y). & (22) \end{aligned}$$

This is of the form (95) with $x = z$, $A = W_x$, $B = W_y$, $a = m_x$, and $b = m_y$. Thus

$$q_z(z) = (z - m_z)^H W_z (z - m_z) + \text{const} \qquad (23)$$

with

$$\begin{aligned} W_z &= W_x + W_y & (24) \\ m_z &= (W_x + W_y)^{\#} (W_x m_x + W_y m_y). & (25) \end{aligned}$$

5.2 Sum Constraint

Rule 2 of Table 1 is proved as follows. The rule $V_z = V_x + V_y$ for the covariance matrices follows directly from elementary probability theory. An alternative derivation from the min-sum rule goes as follows. Assume that both W_x and W_y are nonnegative definite.

According to the min-sum rule (16), we have for $\beta \rightarrow \infty$

$$\begin{aligned} q_z(z) &= \min_x \min_y \Big(q_x(x) + q_y(y) + \beta \|x+y+z\|^2 \Big) && (26) \\ &= \min_{x,y\colon x+y+z=0} \Big(q_x(x) + q_y(y) \Big) && (27) \\ &= \min_{x,y\colon x+y+z=0} \Big((x-m_x)^H W_x (x-m_x) + (y-m_y)^H W_y (y-m_y) \Big) && (28) \end{aligned}$$

Substituting $x+y=-z$ and $x-y=s$, i.e., $2x=s-z$ and $2y=-(s+z)$ yields

$$\begin{aligned} q_z(z) &= \min_{x,y\colon x+y+z=0} \frac{1}{4} \Big((2x-2m_x)^H W_x (2x-2m_x) + \\ &\qquad (2y-2m_y)^H W_y (2y-2m_y) \Big) && (29) \\ &= \min_s \frac{1}{4} \Big((s-z-2m_x)^H W_x (s-z-2m_x) + \\ &\qquad (s+z+2m_y)^H W_y (s+z+2m_y) \Big) && (30) \end{aligned}$$

This is of the form (95) with $x=s$, $A=W_x$, $B=W_y$, $a=z+2m_x$, and $b=-(z+2m_y)$. The minimum of (95) over x is c as in (107). With $a-b=2z+2m_x+2m_y$, we thus have

$$q_z(z) = (z+m_x+m_y)^H W_x (W_x+W_y)^{\#} W_y (z+m_x+m_y). \tag{31}$$

If both W_x and W_y are invertible, then W_x+W_y is invertible and (87) implies $W_x(W_x+W_y)^{-1}W_y = (W_x^{-1}+W_y^{-1})^{-1}$.

5.3 Matrix Multiplication: Forward Message

Rule 3 of Table 1 is proved as follows. The rule $V_y = AV_xA^H$ for the covariance matrices follows directly from elementary probability theory. An alternative derivation from the min-sum rule goes as follows. Assume that W_x is nonnegative definite. According to the min-sum rule (16), we have for $\beta \rightarrow \infty$

$$q_y(y) = \min_x \Big(q_x(x) + \beta \|y-Ax\|^2 \Big) \tag{32}$$

Clearly, if A does not have full row rank (i.e., if the rank of A is less than the number of rows), any vector y that is not in span(A) should be assigned infinite cost; the term $\beta\|y-Ax\|^2$ cannot be made to vanish in this case. Noting that $(I-AA^{\#})y$ is the projection of y onto span$(A)^{\perp}$, this case can be handled in practice by including in W_y a penalty term $\beta(I-AA^{\#})^H(I-AA^{\#})$ where β is some large positive real number.

Having settled that, we now consider the computation of $q_y(y)$ for those y that are in span(A). With this restriction in mind, we have

$$\begin{aligned} q_y(y) &= \min_{x\colon Ax=y} q_x(x) && (33) \\ &= \min_{x\colon Ax=y} (x-m_x)^H W_x (x-m_x) && (34) \end{aligned}$$

In general, this minimization is not easily handled. We therefore focus on two important special cases.

Case 1: If A is invertible, (34) becomes

$$\begin{aligned} q_x(A^{-1}y) &= (A^{-1}y - m_x)^H W_x (A^{-1}y - m_x) \qquad (35) \\ &= (y - Am_x)^H (A^{-1})^H W_x A^{-1} (y - Am_x). \qquad (36) \end{aligned}$$

Case 2: If W_x is invertible, it is positive definite, and so is its inverse $V_x = W_x^{-1}$. We then can write $V_x = BB^H$ for some matrix B that is itself invertible. Thus

$$\begin{aligned} q_x(x) &= (x - m_x)^H W_x (x - m_x) \qquad (37) \\ &= (x - m_x)^H (B^{-1})^H B^{-1} (x - m_x) \qquad (38) \\ &= (B^{-1}x - B^{-1}m_x)^H (B^{-1}x - B^{-1}m_x). \qquad (39) \end{aligned}$$

With $w \triangleq B^{-1}x - B^{-1}m_x$, we obtain

$$\begin{aligned} \min_{x\colon Ax=y} q_x(x) &= \min_{w\colon A(Bw+m_x)=y} \|w\|^2 \qquad (40) \\ &= \min_{w\colon ABw=y-Am_x} \|w\|^2 \qquad (41) \\ &= \|(AB)^{\#}(y - Am_x)\|^2 \qquad (42) \\ &= \big((AB)^{\#}(y - Am_x)\big)^H \, (AB)^{\#} \, (y - Am_x) \qquad (43) \\ &= (y - Am_x)^H \, \big((AB)^{\#}\big)^H \, (AB)^{\#} \, (y - Am_x) \qquad (44) \\ &= (y - Am_x)^H (ABB^H A^H)^{\#} \, (y - Am_x) \qquad (45) \\ &= (y - Am_x)^H (AW^{-1}A^H)^{\#} \, (y - Am_x), \qquad (46) \end{aligned}$$

where the step to (45) uses (81).

5.4 Matrix Multiplication: Backward Message

Rule 4 of Table 1 is proved as follows. Assume that W_x is nonnegative definite. According to the min-sum rule (16), we have for $\beta \to \infty$

$$\begin{aligned} q_y(y) &= \min_x \Big(q_x(x) + \beta \|x - Ay\|^2 \Big) \qquad (47) \\ &= q_x(Ay) \qquad (48) \\ &= (Ay - m_x)^H W_x (Ay - m_x) \qquad (49) \\ &= (Ay)^H W_x Ay - 2\,\mathrm{Re}(y^H A^H W_x m_x) + m_x^H W m_x \qquad (50) \\ &= y^H A^H W_x A y - 2\,\mathrm{Re}\big(y^H (A^H W_x A)(A^H W_x A)^{\#} A^H W_x m_x\big) + \\ &\quad\; m_x^H W_x m_x \qquad (51) \end{aligned}$$

where the equality $A^H W_x = (A^H W_x A)(A^H W_x A)^{\#} A^H W_x$ used in the last step follows from (76) and Property 9 of nonnegative definite matrices.

If the rank of A equals the number of rows, then $AA^{\#} = I$ and

$$\begin{aligned} A^H W_x &= A^H W_x A A^{\#} && (52)\\ &= \left(A^H W_x A\right) A^H \left(AA^H\right)^{-1}. && (53) \end{aligned}$$

5.5 Composite Blocks: Rule 5

Rule 5 of Table 2 is the combination of rules 1 and 4 of Table 1. We assume that both V_x and V_y (and thus W_x and W_y) are positive definite.

$$\begin{aligned} W_z &= W_x + W_w && (54)\\ &= W_x + A^H W_y A && (55) \end{aligned}$$

With the matrix inversion lemma (89), we obtain

$$\begin{aligned} V_z &= W_z^{-1} && (56)\\ &= V_x - V_x A^H \left(V_y + A V_x A^H\right)^{-1} A V_x. && (57) \end{aligned}$$

According to the rules 1 and 4, we further have

$$\begin{aligned} m_z &= \left(W_x + W_w\right)^{-1} \left(W_x m_x + W_w m_w\right) && (58)\\ &= V_z \Big(W_x m_x + A^H W_y A \left(A^H W_y A\right)^{\#} A^H W_y m_y\Big) && (59)\\ &= V_z \left(V_x^{-1} m_x + A^H V_y^{-1} m_y\right). && (60) \end{aligned}$$

Inserting (57) yields

$$\begin{aligned} m_z &= \Big(I - V_x A^H \left(V_y + A V_x A^H\right)^{-1} A\Big) \left(m_x + V_x A^H V_y^{-1} m_y\right) && (61)\\ &= m_x + V_x A^H V_y^{-1} m_y - \\ &\quad\; V_x A^H \left(V_y + A V_x A^H\right)^{-1} A \left(m_x + V_x A^H V_y^{-1} m_y\right) && (62)\\ &= m_x + V_x A^H \left(V_y + A V_x A^H\right)^{-1} && (63)\\ &\quad\; \cdot \left(\left(V_y + A V_x A^H\right) V_y^{-1} m_y - \left(A m_x + A V_x A^H V_y^{-1} m_y\right)\right) && (64)\\ &= m_x + V_x A^H \left(V_y + A V_x A^H\right)^{-1} (m_y - A m_x). && (65) \end{aligned}$$

5.6 Composite Blocks: Rule 6

Rule 6 of Table 2 is the combination of rules 2 and 3 of Table 1. We assume that both W_x and W_y are positive definite.

$$\begin{aligned} V_z &= V_x + V_w && (66)\\ &= V_x + A V_y A^H && (67) \end{aligned}$$

With the matrix inversion lemma (89), we obtain

$$\begin{aligned} W_z &= V_z^{-1} && (68)\\ &= W_x - W_x A \left(W_y + A^H W_x A \right)^{-1} A^H W_x. && (69) \end{aligned}$$

According to the rules 2 and 3, we further have

$$\begin{aligned} m_z &= -m_x - m_w && (70)\\ &= -m_x - A m_y. && (71) \end{aligned}$$

6 Conclusions

Forney-style factor graphs ("normal graphs") are a natural extension of block diagrams that allows for both probabilistic and behavioral modelling. Kalman filtering and recursive least-squares algorithms may both be viewed as message passing in the graph that represents the underlying state space model. In this paper, we have discussed the corresponding message update rules for the building blocks of such models. In particular, we have shown in detail how these rules can be derived from the update rules of the general min-sum or sum-product algorithm.

Recursive least squares and Kalman-filtering are important tools in problems like combined decoding, equalization, channel estimation, etc. The open-system spirit of Forney-style factor graphs makes it easy to use these tools for suitable subtasks of an overall iterative algorithm to solve such problems.

In 1967, Dave Forney introduced trellis diagrams, which have long since become deeply ingrained in the mind of every communications engineer. This writer finds Forney-style factor graphs equally helpful and wishes them an equally happy future.

Appendix

A Singular Value Decomposition and Pseudo-Inverse

A square complex matrix A is *unitary* if $AA^H = I$, i.e., $A^{-1} = A^H$. Any matrix A (not necessarily square) can be written as

$$A = USV^H, \tag{72}$$

where U and V are unitary matrices, and where the matrix S (not necessarily square) is real and diagonal with positive entries only:

$$S = \begin{pmatrix} S^\dagger & 0 \\ 0 & 0 \end{pmatrix} \tag{73}$$

with $S^\dagger \triangleq \text{diag}(\sigma_1, \ldots, \sigma_\ell)$, $\sigma_k > 0$, $k = 1 \ldots \ell$. The decomposition (72) is called the *singular value decomposition* of A.

The *Moore-Penrose generalized inverse* or *pseudo-inverse* of a complex matrix A (not necessarily square) with singular value decomposition (72) is the matrix

$$A^\# \triangleq VS^\# U^H \tag{74}$$

with

$$S^\# \triangleq \begin{pmatrix} \text{diag}(\sigma_1^{-1}, \ldots, \sigma_\ell^{-1}) & 0 \\ 0 & 0 \end{pmatrix}. \tag{75}$$

Note that $S^\#$ has the same dimensions as S^H and $A^\#$ has the same dimensions as A^H. It is easy to see that

$$AA^\# A = A. \tag{76}$$

and

$$A^\# AA^\# = A^\#. \tag{77}$$

Proof: $AA^\# A = USV^H VS^\# U^H USV^H = USS^\# SV^H = USV^H = A$.
$A^\# AA^\# = VS^\# U^H USV^H VS^\# U^H = VS^\# SS^\# U^H = VS^\# U^H = A^\#$. □

It is also easy to see that

$$A^\# x = 0 \Longleftrightarrow x^H A = 0. \tag{78}$$

Proof: $x^H A = 0 \Leftrightarrow x^H US = 0 \Leftrightarrow S^\# U^H x = 0 \Leftrightarrow A^\# x = 0$. □

It follows from (76) and (78) that $AA^{\#}$ is the projection onto span(A), the space spanned by the columns of A.

If the rank of A equals the number of rows, then $AA^{\#} = I$ and

$$A^{\#} = A^H(AA^H)^{-1}; \tag{79}$$

if the rank of A equals the number of columns, then $A^{\#}A = I$ and

$$A^{\#} = (A^HA)^{-1}A^H. \tag{80}$$

Proof: We prove only (79); the proof of (80) is analogous. Under the stated assumptions, we have
$A^H(AA^H)^{-1} = VS^HU^H(USV^HVS^HU^H)^{-1} = VS^HU^H(USS^HU^H)^{-1} = VS^HU^HU(SS^H)^{-1}U^H = VS^H(SS^H)^{-1}U^H = VS^{\#}U^H = A^{\#}$. □

Another useful formula is

$$(AA^H)^{\#} = (A^{\#})^HA^{\#}, \tag{81}$$

which follows from
$(AA^H)^{\#} = (USV(USV)^H)^{\#} = (USVV^HS^HU^H)^{\#} = (USS^HU^H)^{\#} = U(SS^H)^{\#}U^H = U(S^{\#})^HS^{\#}U^H = U(S^{\#})^HV^HVS^{\#}U^H = (A^{\#})^HA^{\#}$.

B Nonnegative Definite Matrices

A square complex matrix A is *Hermitian* if $A^H = A$. If A is Hermitian, then $x^HAx = (x^HAx)^H = \overline{x^HAx}$ is real for any vector x. A Hermitian matrix A is *positive definite (nonnegative definite)* if $x^HAx > 0$ ($x^HAx \geq 0$) for all complex vectors $x \neq 0$. Some basic facts:

1. If A is positive definite (nonnegative definite), then so is aA for any real number $a > 0$.

2. If A and B are nonnegative definite matrices of the same dimensions, the $A + B$ is also nonnegative definite. If, in addition, A (or B) is positive definite, then $A + B$ is positive definite.

3. If A is positive definite, then A is invertible and A^{-1} is also positive definite. (See also Property 10.)

4. The unit matrix I (of any dimension) is positive definite.

5. If B is nonnegative definite, then so is ABA^H for any matrix A (of suitable dimensions, not necessarily square). In particular, AA^H is nonnegative definite.

6. If A is nonnegative definite, then it has a singular value decomposition $A = USU^H$ with unitary U and $S = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$. $A = USU^H$ is positive definite if and only if all singular values σ_k, $k = 1 \ldots n$, are nonzero (i.e., positive).

7. If A is nonnegative definite, then $x^H Ax = 0$ implies $Ax = 0$.

8. If A and B are nonnegative definite matrices of the same dimensions, then we have $\operatorname{span}(A+B) = \operatorname{span}(A, B)$. (Here $\operatorname{span}(W)$ denotes the linear space spanned by the columns of W; $\operatorname{span}(A, B)$ is the space spanned by the columns of A and B.)

9. If B is nonnegative definite, then for any matrix A (of suitable dimensions, not necessarily square), $\operatorname{span}(A^H BA) = \operatorname{span}(A^H B)$.

10. If A is nonnegative definite, then so is its pseudo-inverse $A^\#$. Moreover, $AA^\# = A^\# A$.

Proof:

1. $x^H(aA)x = a(x^H Ax)$.

2. $x^H(A + B)x = x^H Ax + x^H Bx$.

3. Let A be positive definite. Clearly, A is invertible if and only if $Ax = 0$ implies $x = 0$. But $Ax = 0$ implies $x^H Ax = 0$, which in turn implies $x = 0$; thus A is invertible. It remains to show that A^{-1} is also positive definite. Assume $x^H A^{-1} x = 0$. With $y \triangleq A^{-1}x$, we then have $x^H A^{-1} x = y^H Ay = 0$. Thus $y = 0$, which implies $x = 0$.

4. $x^H Ix = x^H x = \|x\|^2$.

5. $x^H(ABA^H)x = (A^H x)^H B(A^H x) \geq 0$.

7. Let $A = USU^H$ be the singular value decomposition of A according to Property 6 and let $y \triangleq U^H x$. Then $x^H Ax = (U^H x)^H S(U^H x) = y^H Sy = \sum_{k=1}^{n} \sigma_k |y_k|^2$. Now assume $x^H Ax = 0$. Since all σ_k are real and nonnegative, this implies $\sigma_k y_k = 0$, $k = 1 \ldots n$, and thus $Sy = 0$. We therefore have $Ax = USU^H x = USy = 0$.

8. Clearly, we always have $\operatorname{span}(A + B) \subseteq \operatorname{span}(A, B)$. We prove equality by showing that the orthogonal complements of these spaces are equal. Some vector x is orthogonal to $\operatorname{span}(A + B)$ if and only if $x^H(A + B) = 0$, and orthogonal to $\operatorname{span}(A, B)$ if and only if $x^H A = x^H B = 0$. But with Property 7, we have $x^H(A + B) = 0 \iff x^H(A + B)x = 0 \iff x^H Ax + x^H Bx = 0 \iff x^H Ax = x^H Bx = 0 \iff x^H A = x^H B = 0$.

9. Clearly, we always have $\operatorname{span}(A^H BA) \subseteq \operatorname{span}(A^H B)$. As above, we prove equality by showing that the orthogonal complements of these spaces are equal. This is established by the following chain of equivalent conditions, where Property 7 is used twice: $x \in \operatorname{span}(A^H B)^\perp \iff x^H A^H B = 0 \iff (Ax)^H B = 0 \iff (Ax)^H B(Ax) = 0 \iff x^H(A^H BA)x = 0 \iff x^H A^H BA = 0 \iff x \in \operatorname{span}(A^H BA)^\perp$.

10. Let $A = USU^H$ be the singular value decomposition of A according to Property 6. Then $A^\# = US^\# U^H$ and $x^H A^\# x = x^H US^\# U^H x = (U^H x)^H S^\# (U^H x) \geq 0$. Moreover, $AA^\# = USU^H US^\# U^H = USS^\# U^H = US^\# SU^H = A^\# A$.
□

If A and B are nonnegative definite matrices of the same dimensions, we also have

$$\begin{aligned} (A+B)(A+B)^\# A &= A && (82) \\ &= A(A+B)^\#(A+B) && (83) \end{aligned}$$

Proof: The two equalities are obtained from each other by Hermitian transposition; it thus suffices to prove one of them. Now (76) implies $(A+B)(A+B)^\# x = x$ for any vector $x \in \operatorname{span}(A+B)$. But according to property 8 above, $\operatorname{span}(A+B) = \operatorname{span}(A, B) \supseteq \operatorname{span}(A)$. Thus $(A+B)(A+B)^\# x = x$ holds for any vector $x \in \operatorname{span}(A)$, which implies (82).
□

From (83) follows $A(A+B)^\# A + A(A+B)^\# B = A$ and thus

$$A(A+B)^\# B = A - A(A+B)^\# A; \tag{84}$$

from (82) follows $A(A+B)^\# A + B(A+B)^\# A = A$ and thus

$$B(A+B)^\# A = A - A(A+B)^\# A. \tag{85}$$

In particular, $A(A+B)^\# B = B(A+B)^\# A = \big(A(A+B)^\# B\big)^H$.

If both A and B are invertible, then $(A+B)$ is invertible and

$$\begin{aligned} A(A+B)^{-1}B &= \big(B^{-1}(A+B)A^{-1}\big)^{-1} && (86) \\ &= \big(A^{-1} + B^{-1}\big)^{-1}. && (87) \end{aligned}$$

Finally, we have the

Matrix Inversion Lemma. Let B and D be positive definite matrices. If, for some matrices A and C, we have

$$A = B^{-1} + CD^{-1}C^H, \tag{88}$$

then

$$A^{-1} = B - BC(D + C^H BC)^{-1} C^H B. \tag{89}$$

Proof:

$$\begin{aligned}
&\left(B^{-1}+CD^{-1}C^H\right)\Big(B-BC\left(D+C^HBC\right)^{-1}C^HB\Big) && (90)\\
&= I+CD^{-1}C^HB-\left(B^{-1}+CD^{-1}C^H\right)BC\left(D+C^HBC\right)^{-1}C^HB && (91)\\
&= I+CD^{-1}C^HB-\left(C+CD^{-1}C^HBC\right)\left(D+C^HBC\right)^{-1}C^HB && (92)\\
&= I+CD^{-1}C^HB-CD^{-1}\left(D+C^HBC\right)\left(D+C^HBC\right)^{-1}C^HB && (93)\\
&= I. && (94)
\end{aligned}$$

□

C Sum of Quadratic Forms

Let both A and B be nonnegative definite matrices (which implies that they are Hermitian). Using (82) in the step to (98), we have

$$\begin{aligned}
&(x-a)^HA(x-a)+(x-b)^HB(x-b) && (95)\\
&= x^HAx-2\,\mathrm{Re}(x^HAa)+a^HAa+x^HBx-2\,\mathrm{Re}(x^HBb)+b^HBb && (96)\\
&= x^H(A+B)x-2\,\mathrm{Re}\big(x^H(Aa+Bb)\big)+a^HAa+b^HBb && (97)\\
&= x^H(A+B)x-2\,\mathrm{Re}\big(x^H(A+B)(A+B)^{\#}(Aa+Bb)\big)+\\
&\quad a^HAa+b^HBb && (98)\\
&= x^HWx-2\,\mathrm{Re}(x^HWm)+m^HWm+c && (99)
\end{aligned}$$

with

$$\begin{aligned}
W &= A+B && (100)\\
m &= (A+B)^{\#}(Aa+Bb) && (101)
\end{aligned}$$

and the scalar

$$
\begin{aligned}
c &= a^H A a + b^H B b - m^H W m && (102)\\
&= a^H A a + b^H B b - (Aa + Bb)^H (A+B)^{\#} (A+B) (A+B)^{\#} (Aa + Bb) \\
& && (103)\\
&= a^H A a + b^H B b - (a^H A + b^H B)(A+B)^{\#}(Aa + Bb) && (104)\\
&= a^H \big(A - A(A+B)^{\#}A\big)\, a - a^H A (A+B)^{\#} B b + \\
&\quad b^H \big(B - B(A+B)^{\#}B\big)\, b - b^H B (A+B)^{\#} A a && (105)\\
&= a^H A (A+B)^{\#} B a - a^H A (A+B)^{\#} B b + \\
&\quad b^H B (A+B)^{\#} A b - b^H B (A+B)^{\#} A a && (106)\\
&= (a-b)^H A (A+B)^{\#} B (a-b), && (107)
\end{aligned}
$$

where the step to (106) follows from (84).

References

[1] B. D. O. Anderson and J. B. Moore, *Optimal Filtering.* Englewood Cliffs, N.J.: Prentice-Hall, 1979.

[2] G. D. Forney, Jr., "Codes on graphs: normal realizations," *IEEE Trans. Inform. Theory,* vol. 47, pp. 520–548, Feb. 2001.

[3] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory,* vol. 47, pp. 498–519, Feb. 2001.

[4] A. H. Sayed and T. Kailath, "A state-space approach to adaptive RLS filtering," *IEEE Signal Proc. Mag.,* vol. 11, pp. 18–60, 1994.

[5] Y. Weiss and W. T. Freeman, "On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs," *IEEE Trans. Inform. Theory,* vol. 47, pp. 736–744, Feb. 2001.

[6] N. Wiberg, *Codes and Decoding on General Graphs.* Linköping Studies in Science and Technology, Ph.D. Thesis No. 440, Univ. Linköping, Sweden, 1996.

[7] J. C. Willems, "Paradigms and puzzles in the theory of dynamical systems," *IEEE Trans. Aut. Contr.,* vol. 36, pp. 259–294, March 1991.