

The Factor Graph Approach to Model-Based Signal Processing

Factor graphs can model complex systems and help to design effective algorithms for detection and estimation problems.

By HANS-ANDREA LOELIGER, *Fellow IEEE*, JUSTIN DAUWELS, *Member IEEE*, JUNLI HU, *Member IEEE*, SASCHA KORL, *Member IEEE*, LI PING, *Senior Member IEEE*, AND FRANK R. KSCHISCHANG, *Fellow IEEE*

ABSTRACT | The message-passing approach to model-based signal processing is developed with a focus on Gaussian message passing in linear state-space models, which includes recursive least squares, linear minimum-mean-squared-error estimation, and Kalman filtering algorithms. Tabulated message computation rules for the building blocks of linear models allow us to compose a variety of such algorithms without additional derivations or computations. Beyond the Gaussian case, it is emphasized that the message-passing approach encourages us to mix and match different algorithmic techniques, which is exemplified by two different approaches—steepest descent and expectation maximization—to message passing through a multiplier node.

KEYWORDS | Estimation; factor graphs; graphical models; Kalman filtering; message passing; signal processing

I. INTRODUCTION

Graphical models such as factor graphs allow a unified approach to a number of topics in coding, signal processing, machine learning, statistics, and statistical

physics. In particular, many algorithms in these fields have been shown to be special cases of the basic sum-product and max-product algorithms that operate by message passing in a factor graph [1]–[3]. In this paper, we elaborate on this topic with an emphasis on signal processing. We hope to convey that factor graphs continue to grow more useful for the design of practical algorithms for model-based detection and estimation problems involving many (discrete and/or continuous) variables. In particular, the factor graph approach allows us to compose nontrivial algorithms for such problems from tabulated rules for “local” computations corresponding to the building blocks of the system model; it also encourages us to mix and match a large variety of techniques ranging from classical Gaussian and gradient techniques over expectation maximization (EM) to sequential Monte Carlo methods (particle filters).

Factor graphs are graphical models [4]–[7]. In many respects, the different notation systems for graphical models (Bayesian networks [7], [8], Markov random fields [7], [9], junction graphs [10], [11], etc.) are essentially equivalent, but there are some real differences when it comes to practical use. Indeed, some of the material of this paper (in particular, in Sections IV, V, and Appendix III) is not easily expressed in other notation systems.

While much interesting recent work on graphical models specifically addresses graphs with cycles (e.g., [12]–[21]), the present paper is mostly concerned with cycle-free graphs, or with cycle-free subgraphs of complex system models. We will encounter some factor graphs with cycles, though.

The message-passing approach to signal processing was suggested in [2], [22] and has been used, e.g., in [23]–[39] for tasks like equalization, multi-user detection, multiple-input multiple-output (MIMO) detection, channel estimation, etc. The literature on graphical models for general inference problems is vast. For example, factor graphs have

Manuscript received May 9, 2006; revised March 13, 2007. The work of J. Dauwels was supported in part by Swiss NF under Grant 200021-101955. The work of S. Korl was supported in part by Phonak AG.

H.-A. Loeliger and **J. Hu** are with the Signal Processing Laboratory (ISI), Department of Information Technology and Electrical Engineering, ETH Zurich, 8092 Zurich, Switzerland (e-mail: loeliger@isi.ee.ethz.ch; hu@isi.ee.ethz.ch).

J. Dauwels was with the Department of Information Technology and Electrical Engineering, ETH Zurich, CH-8092 Zurich, Switzerland. He is now with Amari Research Unit, RIKEN Brain Science Institute, Saitama-ken 351-0106, Japan (e-mail: justin@dauwels.com).

S. Korl was with the Department of Information Technology and Electrical Engineering, ETH Zurich, CH-8092 Zurich, Switzerland. He is now with Phonak AG, CH-8712 Stäfa, Switzerland (e-mail: sascha.korl@phonak.ch).

L. Ping is with the Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong (e-mail: eelping@cityu.edu.hk).

F. R. Kschischang is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: frank@comm.utoronto.ca).

Digital Object Identifier: 10.1109/JPROC.2007.896497

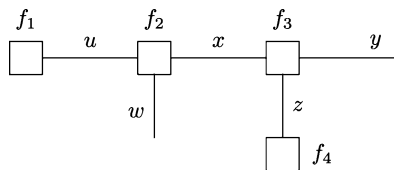


Fig. 1. Forney-style factor graph.

also been used for link monitoring in wireless networks [40], for genome analysis [41], and for clustering [42].

This paper begins with an introduction to factor graphs that complements [2] and [3]. We then turn to an in-depth discussion of Gaussian message passing for linear models, i.e., Kalman filtering and some of its ramifications. In particular, we will present tabulated message computation rules for multivariate Gaussian messages that are extensions and refinements of similar tables in [3] and [33]. With these tables, it is possible to write down (essentially without any computation) efficient Gaussian/linear minimum-mean-squared-error (LMMSE) estimation algorithms for a wide variety of applications, as is illustrated by several nontrivial examples.

Beyond the Gaussian case, we will address the representation of messages for continuous variables and of suitable message computation rules. A wide variety of algorithmic techniques can be put into message-passing form, which helps to mix and match these techniques in complex system models. To illustrate this point, we demonstrate the application of two different techniques, steepest descent and expectation maximization, to message passing through a multiplier node.

Some background material on multivariate Gaussian distributions and LMMSE estimation is summarized in Appendix I. Appendix II contains the proofs for Section V. Appendix III reviews some results by Forney on the Fourier transform on factor graphs [43] and adapts them to the setting of the present paper.

The following notation will be used. The transpose of a matrix (or vector) A is denoted by A^T ; A^H denotes the complex conjugate of A^T ; $A^\#$ denotes the Moore–Penrose pseudo-inverse of A ; and “ \propto ” denotes equality of functions up to a scale factor.

II. FACTOR GRAPHS

We review some basic notions of factor graphs. For complementary introductions to factor graphs and their history and their relation to other graphical models, we refer the interested reader to [2] and [3]. Other than in [2], we will use Forney-style factor graphs (also known as “normal factor graphs”) as in [3]. (The original factor graphs [2] have both variable nodes and factor nodes. Forney-style factor graphs were introduced in [43], but we deviate in some details from the notation of [43].)

Assume, for example, that some function $f(u, w, x, y, z)$ can be factored as

$$f(u, w, x, y, z) = f_1(u)f_2(u, w, x)f_3(x, y, z)f_4(z). \quad (1)$$

This factorization is expressed by the factor graph shown in Fig. 1. In general, a (Forney-style) factor graph consists of nodes, edges, and “half-edges” (which are connected only to one node), and there are the following rules.

- 1) There is a (unique) node for every factor.
- 2) There is a (unique) edge or half-edge for every variable.
- 3) The node representing some factor g is connected with the edge (or half-edge) representing some variable x if and only if g is a function of x .

Implicit in these rules is the assumption that no variable appears in more than two factors. We will see below how this restriction is easily circumvented.

A main application of factor graphs are stochastic models. For example, let X be a real-valued random variable and let Y_1 and Y_2 be two independent real-valued noisy observations of X . The joint probability density of these variables is

$$f(x, y_1, y_2) = f(x)f(y_1|x)f(y_2|x) \quad (2)$$

which we claim is represented by the factor graph of Fig. 2. Literally, Fig. 2 represents an extended model with auxiliary variables X' and X'' and with joint density

$$f(x, x', x'', y_1, y_2) = f(x)f(y_1|x')f(y_2|x'')f_=(x, x', x'') \quad (3)$$

where the equality constraint “function”

$$f_=(x, x', x'') \triangleq \delta(x - x')\delta(x - x'') \quad (4)$$

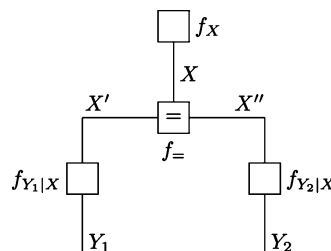


Fig. 2. Factor graph of (2) and (3).

(with $\delta(\cdot)$ denoting the Dirac delta) enforces $P(X = X') = P(X = X'') = 1$. Clearly, (2) is a marginal of (3)

$$f(x, y_1, y_2) = \int_{x'} \int_{x''} f(x, x', x'', y_1, y_2) dx' dx'' \quad (5)$$

For most practical purposes, equality constraint nodes (such as the node labeled “=” in Fig. 2) may be viewed simply as branching points that allow more than two factors to share some variable.

Assume now that Y_1 and Y_2 are observed and that we are interested in the *a posteriori* probability $f(x | y_1, y_2)$. For fixed y_1 and y_2 , we have

$$f(x | y_1, y_2) \propto f(x, y_1, y_2) \quad (6)$$

where “ \propto ” denotes equality up to a scale factor. It follows that $f(x | y_1, y_2)$ is also represented by the factor graph of Fig. 2 (up to a scale factor). This clearly holds in general: passing from some *a priori* model to an *a posteriori* model [based on fixing the value of some variable(s)] does not change the factor graph.

We will usually denote unknown variables by capital letters and known (observed) variables by small letters. (Formally, a B -valued known variable is an element of B while a B -valued unknown variable is a function from the configuration space into B [3].) For the factor graph of $f(x | y_1, y_2)$, we would thus modify Fig. 2 by replacing Y_1 by y_1 and Y_2 by y_2 if these variables are known (observed).

A more detailed version of Fig. 2 is shown in Fig. 3, where we assume

$$Y_1 = X + Z_1 \quad (7)$$

$$Y_2 = X + Z_2 \quad (8)$$

with random variables Z_1 and Z_2 that are independent of each other and of X . The nodes labeled “+” represent the factors $\delta(x + z_1 - y_1)$ and $\delta(x + z_2 - y_2)$, respectively. As illustrated by this example, the (Forney-style) factor graph notation naturally supports modular and hierarchical system modeling. Fig. 3 also illustrates the use of arrows and of special symbols (such as “=” and “+”) to define factors and to make factor graphs readable as block diagrams.

As a nonstochastic example, consider the following system identification problem. Let U_k and \tilde{Y}_k , $k \in \mathbb{Z}$, be the real-valued input signal and the real-valued output signal, respectively, of some unknown system that we wish to approximate by a linear finite impulse response (FIR) filter of order M

$$Y_k = \sum_{\ell=0}^M h_\ell U_{k-\ell} \quad (9)$$

$$\tilde{Y}_k = Y_k + Z_k. \quad (10)$$

The slack variable Z_k absorbs the difference between the filter output Y_k and the observed output \tilde{Y}_k . Assume that we know both the input signal $U_k = u_k$, $k = -M + 1, -M + 2, \dots, N$ and the output signal $\tilde{Y}_k = \tilde{y}_k$, $k = 1, 2, \dots, N$. We wish to determine the unknown filter coefficients h_0, \dots, h_M such that the squared error $\sum_{k=1}^N Z_k^2$ is as small as possible.

We first note that minimizing $\sum_{k=1}^N Z_k^2$ is equivalent to maximizing

$$\prod_{k=1}^N e^{-z_k^2/2\sigma^2} = e^{-\sum_{k=1}^N z_k^2/2\sigma^2} \quad (11)$$

(for arbitrary positive σ^2) subject to the constraints (9) and (10).

For later use, we rewrite (9) as

$$Y_k = [u]_k H \quad (12)$$

with the row vector $[u]_k \triangleq (u_k, u_{k-1}, \dots, u_{k-M})$ and the column vector $H \triangleq (h_0, h_1, \dots, h_M)^T$.

A factor graph of this example is shown in Fig. 4. The lower part represents the unconstrained cost function (11) and the linear constraint (10); the upper part represents the linear constraint (12), i.e., the factors $\delta(y_k - [u]_k H)$ for $k = 1, 2, \dots, N$.

Note that the original least-squares problem to determine the filter coefficients is equivalent to maximizing the function represented by Fig. 4 over all assignments of values to all unknown variables. We will see in Section V

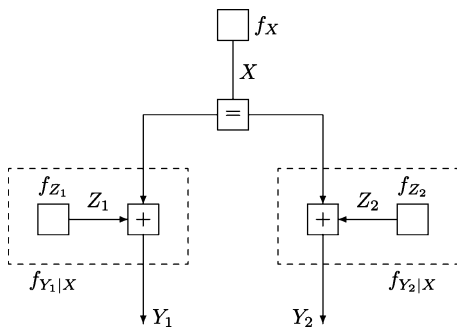


Fig. 3. More detailed version of Fig. 2.

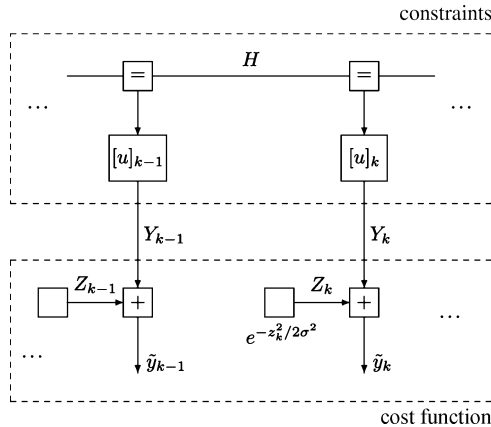


Fig. 4. Two sections of factor graph for FIR filter identification problem.

how efficient recursive least-squares algorithms (RLS) for this maximization may be obtained from tabulated rules for Gaussian message-passing algorithms.

III. MESSAGE-PASSING ALGORITHMS

We briefly review the basic sum-product and max-product algorithms [2], [44]. In this section, the emphasis will be on cycle-free factor graphs.

A. Sum-Product Algorithm

For some given function $f(x_1, \dots, x_n)$, assume that we wish to compute

$$\bar{f}_k(x_k) \triangleq \sum_{\substack{x_1, \dots, x_n \\ \text{except } x_k}} f(x_1, \dots, x_n). \quad (13)$$

If $f(x_1, \dots, x_n)$ is a probability mass function of discrete random variables X_1, \dots, X_n , then (13) is the probability mass function of X_k .

In later sections, we will primarily be interested in continuous variables. In this case, the summation in (13) is replaced by integration.

If $f(x_1, \dots, x_n)$ has a cycle-free factor graph, the function (13) can be computed by the sum-product algorithm, which splits the “big” marginalization (13) into a sequence of “small” marginalizations. For example, assume that $f(x_1, \dots, x_7)$ can be written as

$$f(x_1, \dots, x_7) = f_1(x_1)f_2(x_2)f_3(x_1, x_2, x_3)f_4(x_4) \cdot f_5(x_3, x_4, x_5)f_6(x_5, x_6, x_7)f_7(x_7) \quad (14)$$

as in the factor graph in Fig. 5. Assume that we wish to compute $\bar{f}_3(x_3)$. It is easily verified that

$$\bar{f}_3(x_3) = \mu_C(x_3)\mu_D(x_3) \quad (15)$$

with

$$\mu_C(x_3) \triangleq \sum_{x_1, x_2} f_1(x_1)f_2(x_2)f_3(x_1, x_2, x_3) \quad (16)$$

and

$$\mu_D(x_3) \triangleq \sum_{x_4, x_5} f_4(x_4)f_5(x_3, x_4, x_5)\mu_F(x_5) \quad (17)$$

with

$$\mu_F(x_5) \triangleq \sum_{x_6, x_7} f_6(x_5, x_6, x_7)f_7(x_7). \quad (18)$$

The quantities μ_C , μ_D , and μ_F in (15)–(18) may be viewed as summaries of the dashed boxes in Fig. 5, which are obtained by eliminating the variables inside the box by summation. Such summaries may be thought of as messages in the factor graph, as is shown in Fig. 5.

With the trivial summaries/messages $\mu_A(x_1) \triangleq f_1(x_1)$, and $\mu_B(x_2) \triangleq f_2(x_2)$, we can write (16) as

$$\mu_C(x_3) \triangleq \sum_{x_1, x_2} f_3(x_1, x_2, x_3)\mu_A(x_1)\mu_B(x_2). \quad (19)$$

Similarly, with the trivial summaries $\mu_E(x_4) \triangleq f_4(x_4)$ and $\mu_G(x_7) \triangleq f_7(x_7)$, we can write (17) as

$$\mu_D(x_3) \triangleq \sum_{x_4, x_5} f_5(x_3, x_4, x_5)\mu_E(x_4)\mu_G(x_7) \quad (20)$$

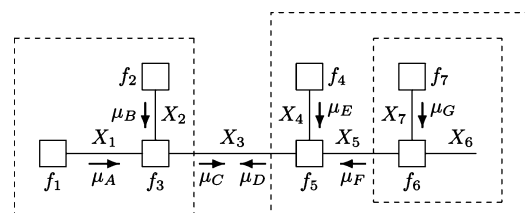


Fig. 5. “Summarized” factors as messages in factor graph.

and (18) as

$$\mu_F(x_5) \triangleq \sum_{x_6, x_7} f_6(x_5, x_6, x_7) \mu_G(x_7). \quad (21)$$

All these messages/summaries (19)–(21) are formed according to the following rule.

Sum-Product Rule: The message out of some node/factor f_ℓ along some edge X_k is formed as the product of f_ℓ and all incoming messages along all edges except X_k , summed over all involved variables except X_k .

From this example, it is obvious that the marginal $\bar{f}_k(x_k)$ may be obtained simultaneously for all k by computing two messages, one in each direction, for every edge in the factor graph. The function \bar{f}_k is the product of these two messages as in (15).

We also observe the following.

- 1) The sum-product algorithm works for any cycle-free factor graph.
- 2) Open half-edges such as X_6 in Fig. 5 do not carry an incoming message. Equivalently, they may be thought as carrying the constant function $\mu(x) = 1$ as incoming message.
- 3) Known variables such as \bar{y}_k in Fig. 4 are simply plugged into the corresponding factors; they are not otherwise involved in the algorithm.
- 4) It is usually (but not always!) sufficient to know the “marginal” (13) up to a scale factor. In such cases, it suffices to know the messages only up to a scale factor. The option to freely scale messages is often essential for numerically reliable computations.

Since the sum-product rule is a “local” computation, it can be applied also to factor graphs with cycles. The sum-product algorithm then becomes an iterative algorithm where messages are recomputed according to some schedule until some stopping criterion is satisfied or until the available time is over. The algorithm may be initialized by assigning to all messages the neutral constant function $\mu(x) = 1$.

B. Max-Product Algorithm

Assume we wish to maximize some function $f(x_1, \dots, x_n)$, i.e., we wish to compute

$$(\hat{x}_1, \dots, \hat{x}_n) = \arg \max_{x_1, \dots, x_n} f(x_1, \dots, x_n) \quad (22)$$

where we assume that f has a maximum. Note that

$$\hat{x}_k = \arg \max_{x_k} \hat{f}_k(x_k) \quad (23)$$

with

$$\hat{f}_k(x_k) \triangleq \max_{\substack{x_1, \dots, x_n \\ \text{except } x_k}} f(x_1, \dots, x_n). \quad (24)$$

If $f(x_1, \dots, x_n)$ has a cycle-free factor graph, the function (24) can be computed by the max-product algorithm. For example, assume that $f(x_1, \dots, x_7)$ can be written as in (14), which corresponds to the factor graph of Fig. 5. Assume that we wish to compute $\hat{f}_3(x_3)$. It is easily verified that

$$\hat{f}_3(x_3) = \mu_C(x_3) \mu_D(x_3) \quad (25)$$

with $\mu_A \dots \mu_G$ defined as in (16)–(21) except that summation is everywhere replaced by maximization. In other words, the max-product algorithm is almost identical to the sum-product algorithm except that the messages are computed as follows.

Max-Product Rule: The message out of some node/factor f_ℓ along some edge X_k is formed as the product of f_ℓ and all incoming messages along all edges except X_k , maximized over all involved variables except X_k .

The remarks at the end of Section III-A apply also to the max-product algorithm. In particular, the “max-marginals” $\hat{f}_k(x_k)$ may be obtained simultaneously for all k by computing two messages, one in each direction, for every edge in the factor graph; $\hat{f}_k(x_k)$ is the product of these two messages as in (25).

The analogies between the sum-product algorithm and the max-product algorithm are obvious. Indeed, the sum-product algorithm can be formulated to operate with abstract addition and multiplication operators “ \oplus ” and “ \otimes ”, respectively, and setting “ \oplus ” = “max” then yields the max-product algorithm [22, Sec. 3.6], [10]. Translating the max-product algorithm into the logarithmic domain yields the max-sum (or min-sum) algorithm [2], [10], [22].

C. Arrows and Notation for Messages

The use of *ad hoc* names for the messages (such as μ_A, \dots, μ_G in Fig. 5) is often unsatisfactory. We will therefore use the following systematic notation. Let X be a variable that is represented by a directed edge (i.e., an edge depicted with an arrow). Then, $\bar{\mu}_X$ denotes the message that flows in the direction of the edge and $\overleftarrow{\mu}_X$ denotes the message in the opposite direction. We will sometimes draw the edges with arrows just for the sake of this notation (e.g., the edge X in Fig. 6 and all edges in Fig. 7).

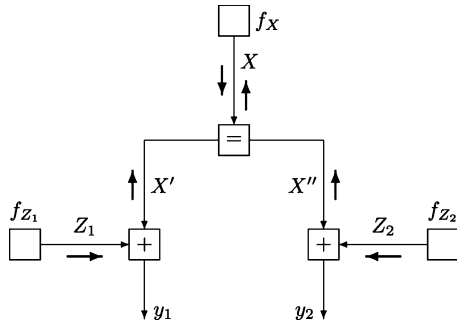


Fig. 6. Message passing in Fig. 3.

D. Example

For a simple example, consider sum-product message passing in the factor graph of Fig. 3 to compute the *a posteriori* probability $f(x|y_1, y_2)$ for known (observed) $Y_1 = y_1$ and $Y_2 = y_2$. The required messages are indicated in Fig. 6. Note that

$$f(x|y_1, y_2) \propto f(x, y_1, y_2) \quad (26)$$

$$= \vec{\mu}_X(x) \overleftarrow{\mu}_X(x). \quad (27)$$

By the rules of the sum-product algorithm, we have

$$\vec{\mu}_{z_1}(z_1) = f_{z_1}(z_1) \quad (28)$$

$$\vec{\mu}_{z_2}(z_2) = f_{z_2}(z_2) \quad (29)$$

$$\overleftarrow{\mu}_{X'}(x') = \int_{z_1} \delta(x' + z_1 - y_1) \vec{\mu}_{z_1}(z_1) dz_1 \quad (30)$$

$$= f_{z_1}(y_1 - x') \quad (31)$$

$$\overleftarrow{\mu}_{X''}(x'') = \int_{z_2} \delta(x'' + z_2 - y_2) \vec{\mu}_{z_2}(z_2) dz_2 \quad (32)$$

$$= f_{z_2}(y_2 - x'') \quad (33)$$

$$\begin{aligned} \overleftarrow{\mu}_X(x) &= \int_{x'} \int_{x''} \delta(x - x') \delta(x - x'') \\ &\quad \cdot \overleftarrow{\mu}_{X'}(x') \overleftarrow{\mu}_{X''}(x'') dx' dx'' \quad (34) \end{aligned}$$

$$= \overleftarrow{\mu}_{X'}(x) \overleftarrow{\mu}_{X''}(x) \quad (35)$$

$$= f_{z_1}(y_1 - x) f_{z_2}(y_2 - x) \quad (36)$$

$$\vec{\mu}_X(x) = f_X(x). \quad (37)$$

In this example, the max-product algorithm computes exactly the same messages.

E. Marginals and Output Edges

Both in the sum-product algorithm and in the max-product algorithm, the final results are the marginals $\overrightarrow{\mu}_X \overleftarrow{\mu}_X$ of the variables (such as X) of interest. It is often convenient to think of such marginals as messages out of a half-edge \bar{X} (without incoming message) connected to $X (= X')$ as shown in Fig. 7; both by the sum-product rule and by the max-product rule, we have

$$\begin{aligned} \overrightarrow{\mu}_{\bar{X}}(x) &= \int_{x'} \int_{x''} \delta(x - x') \delta(x - x'') \\ &\quad \cdot \overrightarrow{\mu}_X(x'') \overleftarrow{\mu}_{X'}(x') dx' dx'' \quad (38) \end{aligned}$$

$$= \overrightarrow{\mu}_X(x) \overleftarrow{\mu}_{X'}(x). \quad (39)$$

It follows that the message computation rules for marginals coincide with the message computation rules out of equality constraint nodes (as, e.g., in Table 2).

IV. LINEAR STATE-SPACE MODELS

Linear state-space models are important in many applications; some examples will be given in the following. In Section V, we will discuss Gaussian sum-product and max-product message passing in such models.

The general linear state-space model may be described as follows. For $k = 1, 2, 3, \dots, N$, the input U_k , the output Y_k , and the state X_k are (real or complex) scalars or vectors coupled by the equations

$$X_k = A_k X_{k-1} + B_k U_k \quad (40)$$

$$Y_k = C_k X_k \quad (41)$$

where A_k , B_k , and C_k are (real or complex) matrices of suitable dimensions. The factor graph corresponding to these equations is given in Fig. 8.

Note that the upper part of Fig. 4 is a special case of Fig. 8 without input (i.e., $B_k = 0$), with $X_k = H$, with A_k an identity matrix, and with $C_k = [u]_k$.

As exemplified by Fig. 4, linear state-space models are often combined with quadratic cost functions or

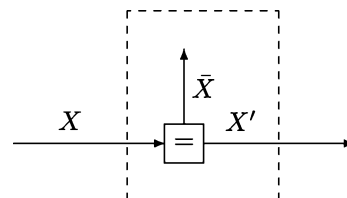


Fig. 7. Adding an output half-edge \bar{X} to some edge X .

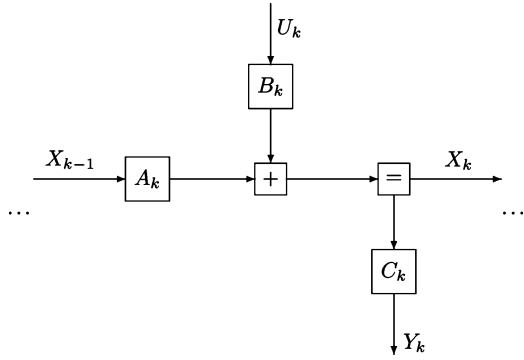


Fig. 8. Factor graph of general linear state-space model (40), (41).

(equivalently) with Gaussian noise models. The factor graph approach to this classical happy combination is the core of the present paper. A key feature of this combination is that the sum-product algorithm coincides with the max-product algorithm (up to a scale factor) and that all messages are Gaussians (or degenerate Gaussians), as will be discussed in Section V and Appendix I.

We will now give two more examples of this kind.

A. Equalization

Consider the transmission of real symbols U_k , $k = 1, \dots, N$, over a discrete-time linear intersymbol interference channel with additive white Gaussian noise. The received real values \tilde{Y}_k , $k = 1, \dots, N$, are given by

$$\tilde{Y}_k = \sum_{\ell=0}^M h_{\ell} U_{k-\ell} + Z_k \quad (42)$$

where Z_k , $k = 1, \dots, N$ are i.i.d. zero-mean Gaussian random variables with variance σ^2 and where h_0, \dots, h_M are known real coefficients. (The initial channel state $U_0, U_{-1}, \dots, U_{-M+1}$ may be known or unknown, depending on the application.)

We bring (42) into the state-space form (40), (41) by defining $Y_k \triangleq \tilde{Y}_k - Z_k$ (the noise-free output), $X_k \triangleq (U_k, \dots, U_{k-M})^T$, and the matrices

$$A_k \triangleq A \triangleq \begin{pmatrix} 0 & 0 \\ I_M & 0 \end{pmatrix} \quad (43)$$

(where I_M denotes the $M \times M$ identity matrix) and

$$B_k \triangleq B \triangleq (1, 0, \dots, 0)^T \quad (44)$$

$$C_k \triangleq C \triangleq (h_0, h_1, \dots, h_M). \quad (45)$$

The factor graph of the model (42) is shown in Fig. 9. The dashed box at the top of Fig. 9 represents a code constraint (e.g., the factor graph of a low-density parity check code) or another model for U_k , if available.

The factor graph in Fig. 9 (without the dashed box) represents the likelihood function $f(\tilde{y}, y, z|u)$ for $u \triangleq (u_1, \dots, u_N)$, $y = (y_1, \dots, y_N)$, etc. If the dashed box in Fig. 9 represents an *a priori* density $f(u)$, then the total graph in Fig. 9 represents

$$f(u)f(\tilde{y}, y, z|u) = f(u, \tilde{y}, y, z) \quad (46)$$

$$\propto f(u, y, z|\tilde{y}). \quad (47)$$

If, in addition, the dashed box in Fig. 9 does not introduce cycles, then the sum-product messages along U_k satisfy

$$\vec{\mu}_{U_k}(u_k) \overleftarrow{\mu}_{U_k}(u_k) \propto f(u_k|\tilde{y}) \quad (48)$$

from which the symbol-wise MAP (maximum *a posteriori*) estimate of U_k may be obtained as

$$\hat{u}_k = \arg \max_{u_k} \vec{\mu}_{U_k}(u_k) \overleftarrow{\mu}_{U_k}(u_k). \quad (49)$$

If the dashed box introduces cycles, we may use iterative (“turbo”) message passing and still use (49) to obtain an estimate of U_k . Efficient algorithms to compute the messages $\vec{\mu}_{U_k}$ will be discussed in Section V-B.

B. Separation of Superimposed Signals

Let $U = (U_1, \dots, U_K)^T$ be a K -tuple of real-valued random variables, let H be a real $N \times K$ matrix, let

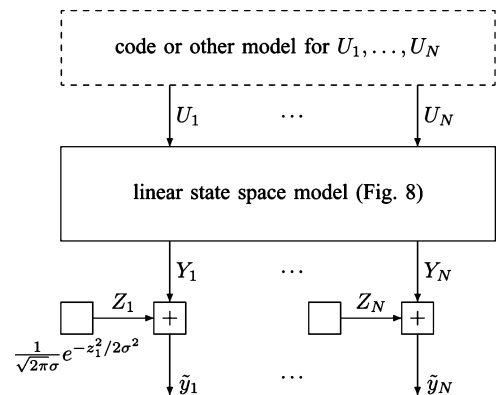


Fig. 9. Factor graph for equalization.

$Y = HU$, and let

$$\tilde{Y} = HU + Z \tag{50}$$

$$= Y + Z \tag{51}$$

where $Z = (Z_1, \dots, Z_N)^T$ is an N -tuple of real zero-mean i.i.d. Gaussian random variables with variance σ^2 . Based on the observation $\tilde{Y} = \tilde{y}$, we wish to compute $p(u_k|\tilde{y})$ and/or a LMMSE estimate of U_k for $k = 1, 2, \dots, K$.

Two different factor graphs for this system model are shown in Figs. 10 and 11. In Fig. 10, h_k denotes the k th column of the matrix H ; in Fig. 11, $e_1 \triangleq (1, 0, \dots, 0)^T$, $e_2 \triangleq (0, 1, 0, \dots, 0)^T$, etc. In Fig. 10, most of the factor graph just represents the decomposition

$$HU = \sum_{k=1}^K h_k U_k. \tag{52}$$

In Fig. 11, most of the factor graph just represents the decomposition

$$U = \sum_{k=1}^K e_k U_k. \tag{53}$$

Although these decompositions appear trivial, the resulting message-passing algorithms in these factor graphs (using tabulated rules for Gaussian messages) are not trivial, cf. Section V-C. The complexity of these algorithms depends mainly on N and K ; Fig. 10 may be preferable for $K > N$ while Fig. 11 may be preferable for $K < N$.

V. GAUSSIAN MESSAGE PASSING IN LINEAR MODELS

Linear models as in Section IV consist of equality constraints (branching points), adders, and multipliers

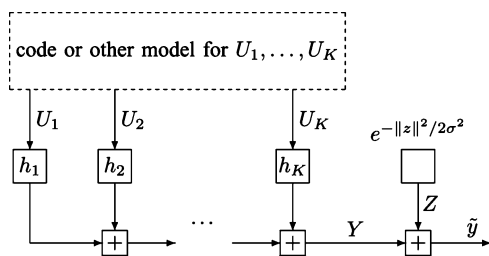


Fig. 10. Factor graph of (50) and (51) (superimposed signals).

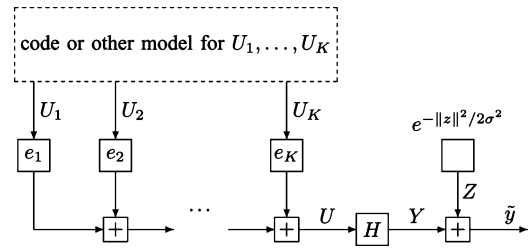


Fig. 11. Another factor graph of (50) and (51) (superimposed signals).

with a constant (scalar or vector or matrix) coefficient. The sum-product and max-product message computation rules for such nodes preserve Gaussianity: if the incoming messages are members of the exponential family, then so are the outgoing messages.

In this section, the computation of such messages will be considered in detail. For scalar variables, there is not much to say; for vector variables, however, the efficient computation of messages and marginals is not trivial. The at heart of this section are Tables 2–6 with formulas for such messages and marginals. These tables allow us to compose a variety of efficient algorithms—both classic and recent variations of Kalman filtering—without additional computations or derivations.

However, before entering into the details of these tables, it should be pointed out that the algorithms considered here are many different things at the same time.

- 1) For linear Gaussian factor graphs, the sum-product algorithm and the max-product algorithm coincide (up to a scale factor). (This follows from Theorem 4 in Appendix I.)
- 2) The Gaussian assumption does not imply a stochastic setting. For example, the least-squares problem of Fig. 4 may also be viewed as a linear Gaussian problem.
- 3) In a stochastic setting, the max-product (= sum-product) algorithm may be used to compute both maximum *a posteriori* (MAP) and maximum likelihood (ML) estimates.
- 4) MAP estimation in a linear Gaussian model coincides both with minimum-mean-squared-error (MMSE) estimation and with LMMSE estimation, cf. Appendix I.
- 5) MAP estimation with *assumed* Gaussians coincides with *true* LMMSE estimation (cf., Theorem 3 in Appendix I). It follows that LMMSE estimation may be carried out by Gaussian message passing in an appropriate linear model.
- 6) If the sum-product algorithm converges in a Gaussian factor graph with cycles, then the means of the marginals are correct (despite the cycles) [12], [13].

It is thus obvious that Gaussian message passing in linear models encompasses much of classical signal processing.

We now turn to the actual message computations. In this section, all messages will be (multivariate) Gaussian distributions, up to scale factors. Gaussian distributions will be described either by the mean vector \vec{m} and the covariance matrix V or by the weight matrix $W \triangleq V^{-1}$ and the transformed mean Wm , cf. Appendix I. (It happens quite frequently that either V or W are singular for certain messages, but this is seldom a serious problem.)

We will use the notation for messages of Section III-C, which we will extend to the parameters m , V , and W . If some directed edge represents the variable X , the forward message has mean \vec{m}_X , covariance matrix \vec{V}_X , and weight matrix $\vec{W}_X = \vec{V}_X^{-1}$; the backward message has mean \overleftarrow{m}_X , covariance matrix \overleftarrow{V}_X , and weight matrix $\overleftarrow{W}_X = \overleftarrow{V}_X^{-1}$. The product of these two messages—the marginal of the global function if the factor graph has no cycles—is the Gaussian with mean m_X and covariance matrix $V_X = W_X^{-1}$ given by

$$W_X = \vec{W}_X + \overleftarrow{W}_X \quad (54)$$

and

$$W_X m_X = \vec{W}_X \vec{m}_X + \overleftarrow{W}_X \overleftarrow{m}_X. \quad (55)$$

[Equations (54) and (55) are equivalent to (II.1) and (II.3) in Table 2, cf. Section III-E.]

An open half-edge without an incoming message may be viewed as carrying as incoming message the constant function 1, which is the limit of a Gaussian with $W = 0$ and mean $m = 0$. A half-edge representing some known variable $X = x_0$ may be viewed as carrying the incoming message $\delta(x - x_0)$, which is the limit of a Gaussian with $V = 0$ and mean $m = x_0$.

We will also use the auxiliary quantity

$$\tilde{W}_X \triangleq (\vec{V}_X + \overleftarrow{V}_X)^{-1} \quad (56)$$

which is dual to $V_X = (\vec{W}_X + \overleftarrow{W}_X)^{-1}$. Some relations among these quantities and the corresponding message parameters are given in Table 1. These relations are proved and then used in other proofs, in Appendix II.

Computation rules for the parameters of messages (such as \vec{m}_X , \vec{V}_X , etc.) and marginals (such as m_X , V_X , etc.) are listed in Tables 2–4 and 6. The equation numbers in these tables are prefixed with the table number; for example, (II.3) denotes equation 3 in Table 2. The proofs are given in Appendix II.

In principle, Tables 2 and 3 suffice to compute all messages in a linear model. However, using only the rules

of Tables 2 and 3 leads to frequent transformations of \vec{W} and \vec{m} into $\vec{V} = \vec{W}^{-1}$ and \vec{m} , and vice versa; if \vec{V} and \vec{W} are large matrices, such conversions are costly.

The inversion of large matrices can often be avoided by using the message computation rules given in Table 4 (which follow from the Matrix Inversion Lemma [47], cf. Theorem 7 in Appendix II). The point of these rules is that the dimension of Y may be much smaller than the dimension of X and Z ; in particular, Y may be a scalar.

Table 3 shows the propagation of \vec{V} and \vec{m} forward through a matrix multiplication node as well as the

Table 1 Single-Edge Relations Involving \vec{W}

$\vec{W}_X = \vec{W}_X V_X \vec{W}_X$	(I.1)
$= \vec{W}_X - \vec{W}_X V_X \vec{W}_X$	(I.2)
$V_X = \vec{V}_X \vec{W}_X \overleftarrow{V}_X$	(I.3)
$= \vec{V}_X - \vec{V}_X \vec{W}_X \overleftarrow{V}_X$	(I.4)
$m_X = V_X \vec{W}_X \vec{m}_X + V_X \overleftarrow{W}_X \overleftarrow{m}_X$	(I.5)
$= \vec{m}_X - \vec{V}_X \vec{W}_X \vec{m}_X + V_X \overleftarrow{W}_X \overleftarrow{m}_X.$	(I.6)
The direction of the arrows may be reversed in all these relations.	

Table 2 Gaussian Messages: Elementary Nodes

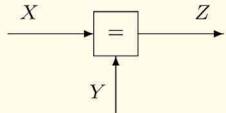
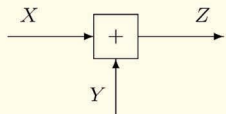
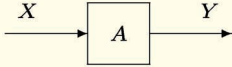
	
$\vec{W}_Z = \vec{W}_X + \vec{W}_Y$	(II.1)
$\overleftarrow{W}_X = \overleftarrow{W}_Z + \overleftarrow{W}_Y$	(II.2)
$\vec{W}_Z \vec{m}_Z = \vec{W}_X \vec{m}_X + \vec{W}_Y \vec{m}_Y$	(II.3)
$\overleftarrow{W}_X \overleftarrow{m}_X = \overleftarrow{W}_Z \overleftarrow{m}_Z + \overleftarrow{W}_Y \overleftarrow{m}_Y$	(II.4)
$m_X = m_Y = m_Z$	(II.5)
$V_X = V_Y = V_Z$	(II.6)
	
$\vec{V}_Z = \vec{V}_X + \vec{V}_Y$	(II.7)
$\overleftarrow{V}_X = \overleftarrow{V}_Z + \overleftarrow{V}_Y$	(II.8)
$\vec{m}_Z = \vec{m}_X + \vec{m}_Y$	(II.9)
$\overleftarrow{m}_X = \overleftarrow{m}_Z - \overleftarrow{m}_Y$	(II.10)
$m_X + m_Y - m_Z = 0$	(II.11)
$\vec{W}_X = \vec{W}_Y = \vec{W}_Z$	(II.12)

Table 3 Gaussian Messages: Matrix Multiplication Node

	
Forward:	$\vec{V}_Y = A \vec{V}_X A^H \quad (\text{III.1})$ $\vec{m}_Y = A \vec{m}_X \quad (\text{III.2})$ $m_Y = A m_X \quad (\text{III.3})$ $V_Y = A V_X A^H \quad (\text{III.4})$
For \vec{W}_Y , see also Tables 5 and 6.	
Backward:	$\overleftarrow{W}_X = A^H \overleftarrow{W}_Y A \quad (\text{III.5})$ $\overleftarrow{W}_X \overleftarrow{m}_X = A^H \overleftarrow{W}_Y \overleftarrow{m}_Y \quad (\text{III.6})$ $\overleftarrow{W}_X m_X = A^H \overleftarrow{W}_Y m_Y \quad (\text{III.7})$ $\tilde{W}_X = A^H \tilde{W}_Y A \quad (\text{III.8})$ $\tilde{W}_X \overleftarrow{m}_X = A^H \tilde{W}_Y \overleftarrow{m}_Y \quad (\text{III.9})$
For \overleftarrow{V}_X and \overleftarrow{m}_X , see also Tables 5 and 6.	

propagation of \overleftarrow{W} and $\overleftarrow{W}\overleftarrow{m}$ backward through such a node. In the other direction, Table 5 may help: Table 5 (top) together with Table 4 (bottom) allows the propagation of \overleftarrow{W} and $\overleftarrow{W}\overleftarrow{m}$ forward through a matrix multiplication node; Table 5 (bottom) together with Table 4 (top) allows the propagation of \overleftarrow{V} and \overleftarrow{m} backward through a matrix multiplication node. The new “internal” open input in Table 5 (top) may be viewed as carrying an incoming message that is a degenerate Gaussian with $\overleftarrow{W}=0$ and mean $\overleftarrow{m}=0$; the new “internal” output in Table 5 (bottom) may be viewed as carrying an incoming message that is a degenerate Gaussian with $\overleftarrow{m}=0$ and $\overleftarrow{V}=0$.

The point of the groupings in Table 6 is to create invertible matrices out of singular matrices.

We now demonstrate the use of these tables by three examples, each of which is of interest on its own.

A. RLS Algorithms

Consider a factor graph as in Fig. 12, where $X_0 = X_1 = X_2 = \dots$ are (unknown) real column vectors and where c_1, c_2, \dots , are (known) real row vectors. The classic recursive least-squares (RLS) algorithm [47] may be viewed as forward-only (left-to-right) Gaussian message passing through this factor graph, with an extra twist.

Note that Fig. 4 is a special case of Fig. 12 (with $X_k = H$ and with $c_k = [u]_k$). It follows that the RLS algorithm may

be used, in particular, to solve the FIR filter identification problem stated at the end of Section II.

We begin by noting that the max-product message $\vec{\mu}_{X_k}$ (as shown in Fig. 12) is

$$\vec{\mu}_{X_k}(x) \propto \max_{z_1, \dots, z_k} \prod_{\ell=1}^k e^{-z_\ell^2 / (2\sigma^2)} \quad (\text{57})$$

$$= e^{-\min_{z_1, \dots, z_k} \sum_{\ell=1}^k z_\ell^2} \quad (\text{58})$$

subject to the constraints $\tilde{y}_\ell = c_\ell x + z_\ell$, $\ell = 1, \dots, k$. It follows that the least-squares estimate (or, in a stochastic

Table 4 Gaussian Messages: Composite Blocks. (Most Useful if Y Is Scalar)

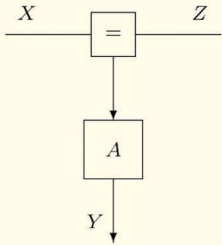
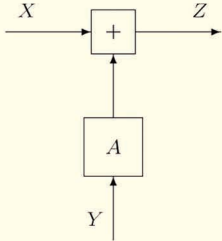
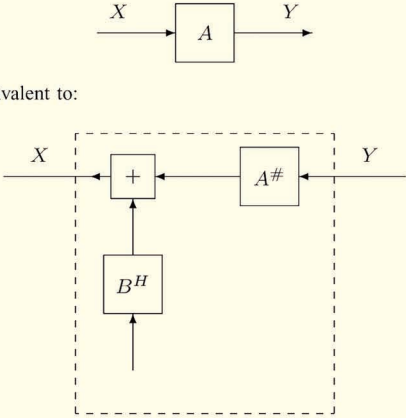
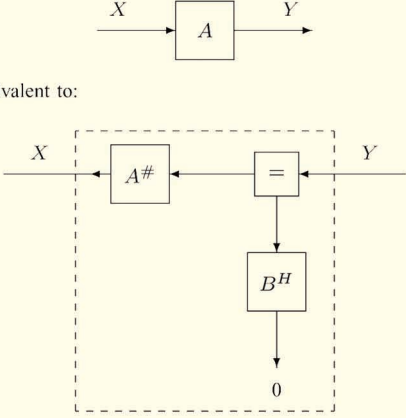
	
	$\vec{m}_Z = \vec{m}_X + \vec{V}_X A^H G (\overleftarrow{m}_Y - A \vec{m}_X) \quad (\text{IV.1})$
	$\vec{V}_Z = \vec{V}_X - \vec{V}_X A^H G A \vec{V}_X \quad (\text{IV.2})$
	$\text{with } G \triangleq (\overleftarrow{V}_Y + A \vec{V}_X A^H)^{-1} \quad (\text{IV.3})$
For $\vec{\mu}_Y$, consider using (III.3) and (III.4).	
	
	$\vec{m}_Z = \vec{m}_X + A \vec{m}_Y \quad (\text{IV.4})$
	$\overleftarrow{m}_X = \overleftarrow{m}_Z - A \vec{m}_Y \quad (\text{IV.5})$
	$\overleftarrow{W}_Z = \overleftarrow{W}_X - \overleftarrow{W}_X A H A^H \overleftarrow{W}_X \quad (\text{IV.6})$
	$\text{with } H \triangleq (\overleftarrow{W}_Y + A^H \overleftarrow{W}_X A)^{-1} \quad (\text{IV.7})$
	$\overleftarrow{W}_Z \overleftarrow{m}_Z = \overleftarrow{W}_X \overleftarrow{m}_X + \overleftarrow{W}_X A H (\overleftarrow{W}_Y \overleftarrow{m}_Y - A^H \overleftarrow{W}_X \overleftarrow{m}_X) \quad (\text{IV.8})$
For \overleftarrow{W}_X , change \overleftarrow{W}_X to \overleftarrow{W}_Z and \overleftarrow{W}_Z to \overleftarrow{W}_X in (IV.6) and (IV.7); for $\overleftarrow{W}_X \overleftarrow{m}_X$, change also the sign of \overleftarrow{m}_Y in (IV.8).	
For $\vec{\mu}_Y$, consider using (II.12), (III.8), (III.9).	

Table 5 Reversing a Matrix Multiplication

<p>If rank $A = \text{number of rows} < \text{number of columns}$:</p>  <p>is equivalent to:</p> <p>where $A^\# = A^H(AA^H)^{-1}$ and where B is a matrix such that $\begin{pmatrix} A \\ B \end{pmatrix}$ is a nonsingular square matrix and $AB^H = 0$.</p> <p>This decomposition allows to compute \vec{W}_Y and $\vec{W}_Y \vec{m}_Y$ by means of (IV.6)–(IV.8) and (III.5)–(III.6).</p>
<p>If rank $A = \text{number of columns} < \text{number of rows}$:</p>  <p>is equivalent to:</p> <p>where $A^\# = (A^H A)^{-1} A^H$ and where B is a matrix such that $\begin{pmatrix} A \\ B \end{pmatrix}$ is a nonsingular square matrix and $B^H A = 0$.</p> <p>This decomposition allows to compute \vec{V}_X and \vec{m}_X by means of (IV.1)–(IV.2) and (III.1)–(III.2).</p>

setting, the ML estimate) of $X_k (= X_0)$ based on $\tilde{y}_1, \dots, \tilde{y}_k$ is

$$\hat{x}_k \triangleq \arg \max_x \vec{\mu}_{X_k}(x) \quad (59)$$

$$= \arg \min_x \min_{z_1, \dots, z_k} \sum_{\ell=1}^k z_\ell^2 \Big|_{\tilde{y}_\ell = c_\ell x + z_\ell} \quad (60)$$

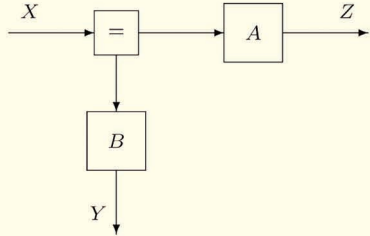
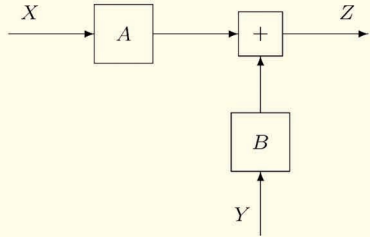
$$= \vec{m}_k \quad (61)$$

the mean of the (properly normalized) Gaussian $\vec{\mu}_{X_k}$.

Now, we consider the actual computation of the messages shown in Fig. 12. The message $\vec{\mu}_{Z_k}$ is simply the factor $e^{-z_k^2/2\sigma^2}$, i.e., a Gaussian with mean $\vec{m}_{Z_k} = 0$ and variance $\vec{V}_{Z_k} = \sigma^2$. The fact that $\tilde{Y}_k = \tilde{y}_k$ is known may be expressed by an incoming degenerate-Gaussian message $\vec{\mu}_{\tilde{Y}}$ with mean $\vec{m}_{\tilde{Y}} = \tilde{y}_k$ and variance $\vec{V}_{\tilde{Y}} = 0$. It then follows from (II.10) and (II.8) in Table 2 that the message $\vec{\mu}_{Y_k}$ is Gaussian with mean $\vec{m}_Y = \tilde{y}_k$ and variance $\vec{V}_Y = \sigma^2$.

So much for the trivial (scalar-variable) messages in Fig. 12. As for the messages $\vec{\mu}_{X_k}$, we note that the RLS algorithm comes in two versions. In one version, these messages are represented by the mean vector \vec{m}_{X_k} and the

Table 6 Combining Rectangular Matrices to Form a Nonsingular Square Matrix

 <p>If $\begin{pmatrix} A \\ B \end{pmatrix}$ is a nonsingular square matrix:</p> $\vec{m}_X = \begin{pmatrix} A \\ B \end{pmatrix}^{-1} \begin{pmatrix} \vec{m}_Z \\ \vec{m}_Y \end{pmatrix} \quad (VI.1)$ $\vec{V}_X = \begin{pmatrix} A \\ B \end{pmatrix}^{-1} \begin{pmatrix} \vec{V}_Z & 0 \\ 0 & \vec{V}_Y \end{pmatrix} (A^H, B^H)^{-1} \quad (VI.2)$
 <p>If (A, B) is a nonsingular square matrix:</p> $\vec{W}_Z \vec{m}_Z = \begin{pmatrix} A^H \\ B^H \end{pmatrix}^{-1} \begin{pmatrix} \vec{W}_X \vec{m}_X \\ \vec{W}_Y \vec{m}_Y \end{pmatrix} \quad (VI.3)$ $\vec{W}_Z = \begin{pmatrix} A^H \\ B^H \end{pmatrix}^{-1} \begin{pmatrix} \vec{W}_X & 0 \\ 0 & \vec{W}_Y \end{pmatrix} (A, B)^{-1} \quad (VI.4)$ $\begin{pmatrix} m_X \\ m_Y \end{pmatrix} = (A, B)^{-1} m_Z \quad (VI.5)$ $\begin{pmatrix} V_X & V_{XY} \\ V_{YX} & V_Y \end{pmatrix} = (A, B)^{-1} V_Z \begin{pmatrix} A^H \\ B^H \end{pmatrix}^{-1} \quad (VI.6)$

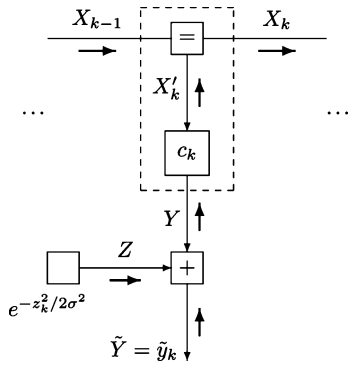


Fig. 12. RLS as Gaussian message passing.

covariance matrix \vec{V}_{X_k} ; in the other version, these messages are represented by the weight matrix $\vec{W}_{X_k} = \vec{V}_{X_k}^{-1}$ and the transformed mean $\vec{W}_{X_k} \vec{m}_{X_k}$. In the latter case, the message computation rules are immediate from Tables 2 and 3: we first obtain $\vec{\mu}_{X_k'}$ (the message inside the dashed box) from (III.5) and (III.6), and then $\vec{\mu}_{X_k}$ from (II.1) and (II.3). Note that no matrix inversion is required to compute these messages. However, extracting the estimate $\hat{x}_k = \vec{m}_{X_k}$ from $\vec{W}_{X_k} \vec{m}_{X_k}$ amounts to solving a system of linear equations. For the initial message $\vec{\mu}_{X_0}$, we set $\vec{W}_{X_0} = 0$ (the all-zeros matrix).

The other version of the RLS algorithm is obtained by grouping the two nodes inside the dashed box in Fig. 12 and using (IV.1) and (IV.2) of Table 4 to propagate \vec{m}_{X_k} and \vec{V}_{X_k} . Again, no matrix inversion is required to compute these messages: the inversion in (IV.3) is only a scalar division in this case. In this version of the RLS algorithm, the estimate $\hat{x}_k = \vec{m}_{X_k}$ is available at every step without extra computations. As for the initial message $\vec{\mu}_{X_0}$, an obvious practical approach is to set $\vec{m}_{X_0} = 0$ and $\vec{V}_{X_0} = \rho I$,

where I is the identity matrix of appropriate dimensions and where ρ is some “large” real number. (In a stochastic setting, this initialization amounts to a prior on X_0 , which turns $\hat{x}_k = \vec{m}_{X_k}$ into a MAP estimate.)

We have now fully described max-product (= sum-product) Gaussian message passing through Fig. 12. However, as mentioned, the classical RLS algorithm introduces an extra twist: in every step, the covariance matrix \vec{V}_{X_k} (as computed above) is multiplied by a scale factor $\gamma > 1$, $\gamma \approx 1$, or (equivalently) \vec{W}_{X_k} is multiplied by $1/\gamma$. In this way, the algorithm is made to slowly forget the past. In the adaptive-filter problem of Fig. 4, this allows the algorithm to track slowly changing filter coefficients and it improves its numerical stability.

B. Gaussian Message Passing in General Linear State-Space Model

We now consider Gaussian message passing in the general linear state-space model of Fig. 8, which encompasses Kalman filtering (= forward-only message passing), Kalman smoothing, LMMSE turbo equalization, etc. The discussion will be based on Fig. 13, which is a more detailed version of Fig. 8 with named variables assigned to every edge and with an optional decomposition of the state transition matrix A_k (if A_k is not square or not regular) into

$$A_k = A_k'' A_k' \tag{62}$$

with matrices A_k' and A_k'' such that the rank of A_k' equals the number of rows of A_k' and the rank of A_k'' equals the number of columns of A_k'' . (In other words, the multiplication by A_k' is a surjective mapping and the multiplication by A_k'' is an injective mapping.) Such a decomposition is always possible. In the example of

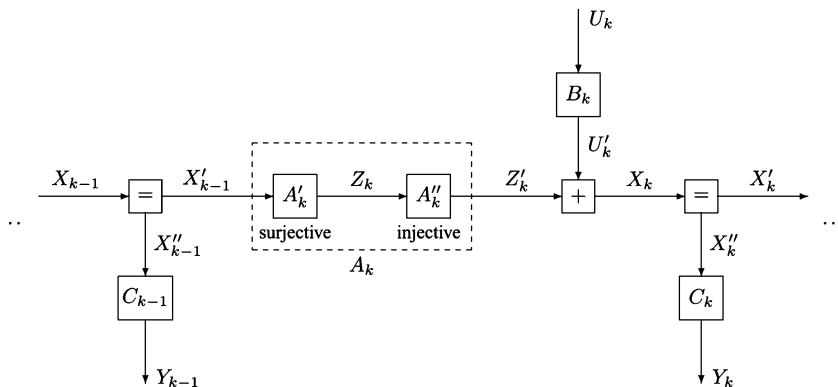


Fig. 13. Factor graph of general linear state-space model (extended version of Fig. 8).

Section IV-A (equalization of an FIR channel), the decomposition (62) yields $A'_k = (I_M, 0)$ and $A''_k = \begin{pmatrix} 0 \\ I_M \end{pmatrix}$; the pseudo-inverses of these matrices (which are used below) are $(A'_k)^\# = (A'_k)^T$ and $(A''_k)^\# = (A''_k)^T$.

The inputs U_k and the outputs Y_k in Fig. 13 are usually scalars (while the state variables X_k are vectors). If U_k is a vector, it may be advantageous to decompose it (and accordingly B_k) into parallel scalar inputs; similarly, if Y_k is a vector, it may be advantageous to decompose it (and C_k) into parallel scalar outputs.

If all inputs U_k and all outputs Y_k are scalars, then none of the algorithms given below requires repeated matrix inversions. In fact, some of the remarks on complexity in the algorithms that follow implicitly assume that U_k and Y_k are scalars.

Let us recall that the point of all the following algorithms is efficiency; if we do not mind inverting matrices all the time, then Tables 2 and 3 suffice. There are also, of course, issues with numerical stability, but they are outside the scope of this paper.

Algorithm A—Forward Recursion With \vec{m}_{X_k} and \vec{V}_{X_k} : (This algorithm is known as “covariance matrix Kalman filter” [47].) From $\vec{m}_{X_{k-1}}$ and the incoming message $\overleftarrow{\mu}_{Y_{k-1}}$, the message $\vec{\mu}_{X'_{k-1}}$ is obtained by (IV.1) and (IV.2). (X''_{k-1} is skipped.) The message $\overleftarrow{\mu}_{Z'_k}$ is obtained by (III.1) and (III.2). (Z_k may be skipped.) The message $\overleftarrow{\mu}_{U'_k}$ is obtained from the incoming message $\overleftarrow{\mu}_{U_k}$ by (III.1) and (III.2). The message $\vec{\mu}_{X_k}$ is then obtained by (II.7) and (II.9).

Algorithm B—Backward Recursion With \overleftarrow{W}_{X_k} and $\overleftarrow{W}_{X_k} \overleftarrow{m}_{X_k}$: (This algorithm is known as “information matrix Kalman filter” [47].) From the incoming message $\overleftarrow{\mu}_{Y_k}$, the message $\overleftarrow{\mu}_{X'_k}$ is obtained by (III.5) and (III.6). From this and from the backward message $\overleftarrow{\mu}_{X'_k}$, the message $\overleftarrow{\mu}_{X_k}$ is obtained by (II.1) and (II.3). The message at $\overleftarrow{\mu}_{Z'_k}$ is then obtained by (IV.4) and (IV.6). (U'_k is skipped.) The message $\overleftarrow{\mu}_{X'_{k-1}}$ is obtained by (III.5) and (III.6). (Z_k may be skipped.)

If the state transition matrix A_k is square and nonsingular, then the algorithms can, of course, be applied in the reverse direction. However, even if A_k is singular (as, e.g., for FIR channel equalization), it is possible to forward propagate \vec{W}_{X_k} and $\vec{W}_{X_k} \vec{m}_{X_k}$ and to backward propagate \overleftarrow{m}_{X_k} and \overleftarrow{V}_{X_k} without matrix inversions. These algorithms rely on the decomposition (62), which allows us to group A'_k with C_{k-1} as in Table 6 (top) and A''_k with B_k as in Table 6 (bottom).

Algorithm C—Forward With \vec{W}_{X_k} and $\vec{W}_{X_k} \vec{m}_{X_k}$ if A_k is Singular: From the incoming message $\overleftarrow{\mu}_{Y_{k-1}}$, the message $\overleftarrow{\mu}_{X''_{k-1}}$ is obtained by (III.5) and (III.6). From this and from $\overleftarrow{\mu}_{X_{k-1}}$, the message $\vec{\mu}_{X'_{k-1}}$ is obtained by (II.1) and (II.3).

The message $\overleftarrow{\mu}_{Z'_k}$ is obtained by the decomposition of A'_k as in Table 5 (top) and using first (IV.4) and (IV.6) and then (III.5) and (III.6). The message $\overleftarrow{\mu}_{X_k}$ is obtained by grouping A''_k with B_k as in Table 6 (bottom) and using (VI.3) and (VI.4). (Z'_k is skipped.)

Algorithm D—Backward With \overleftarrow{m}_{X_k} and \overleftarrow{V}_{X_k} if A_k is Singular: The message $\overleftarrow{\mu}_{U'_k}$ is obtained from the incoming message $\overleftarrow{\mu}_{U_k}$ by (III.1) and (III.2). From this and from $\overleftarrow{\mu}_{X_k}$, the message $\overleftarrow{\mu}_{Z'_k}$ is obtained by (II.8) and (II.10). The message $\overleftarrow{\mu}_{Z_k}$ is obtained by the decomposition of A''_k as in Table 5 (bottom) and using first (IV.1) and (IV.2) and then (III.1) and (III.2). The message $\overleftarrow{\mu}_{X_{k-1}}$ is obtained by grouping A'_k with C_{k-1} as in Table 6 (top) and using (VI.1) and (VI.2). (X'_{k-1} is skipped.)

By combining Algorithm B either with its own forward version (if A_k is nonsingular for all k) or with Algorithm C (if A_k is singular), we can get W_{X_k} ($= \overleftarrow{W}_{X_k} + \overleftarrow{W}_{X_k}$) as well as

$$W_{X_k} m_{X_k} = \overleftarrow{W}_{X_k} \overleftarrow{m}_{X_k} + \overleftarrow{W}_{X_k} \overleftarrow{m}_{X_k} \quad (63)$$

for all k . The estimate $\hat{x}_k = m_{X_k}$ may then be obtained by solving (63) for m_{X_k} .

The computation of V_{X_k} and/or of output messages $\overleftarrow{\mu}_{U_k}$ and/or $\overleftarrow{\mu}_{Y_k}$ requires more effort. In fact, the computation of $\overleftarrow{\mu}_{Y_k}$ may be reduced to the computation of m_{X_k} and V_{X_k} , and the computation of $\overleftarrow{\mu}_{U_k}$ may be reduced to the computation of $m_{U'_k}$ and \overleftarrow{W}_{X_k} . Specifically, the output message $\overleftarrow{\mu}_{Y_k}$ may be extracted from the incoming message $\overleftarrow{\mu}_{Y_k}$ and from V_{Y_k} and m_{Y_k} , and the latter two quantities are easily obtained from V_{X_k} and m_{X_k} by means of (II.5), (II.6), and (III.3), (III.4). The output message $\overleftarrow{\mu}_{U_k}$ may be extracted from the incoming message $\overleftarrow{\mu}_{U_k}$ and from \overleftarrow{W}_{U_k} and m_{U_k} , and the latter two quantities are easily obtained from \overleftarrow{W}_{X_k} and $m_{U'_k}$ [$= m_{X_k} - m_{Z'_k}$ by (II.11)] by means of (II.12), (III.8), and (III.7).

In the example of Section IV-A (FIR channel equalization), m_{U_k} and V_{U_k} may actually be read off directly from m_{X_ℓ} and V_{X_ℓ} , respectively, for any ℓ , $k - M \leq \ell \leq k$. In this case, computing V_{X_k} only every M steps suffices to compute all outgoing messages. These few computations of V_{X_k} are perhaps best carried out directly according to $V_{X_k} = (\overleftarrow{W}_{X_k} + \overleftarrow{W}_{X_k})^{-1}$ or $V_{X_k} = (\overleftarrow{V}_{X_k}^{-1} + \overleftarrow{W}_{X_k})^{-1}$.

However, it is possible to compute V_{X_k} , W_{X_k} , and m_{X_k} (and thus also all outgoing messages) in a general linear state-space model without any matrix inversions (assuming that U_k and Y_k are scalars):

Algorithm E—All Marginals and Output Messages by Forward-Backward Propagation: Forward pass: with \vec{m}_{X_k} and \vec{V}_{X_k} according to Algorithm A. Backward pass: with

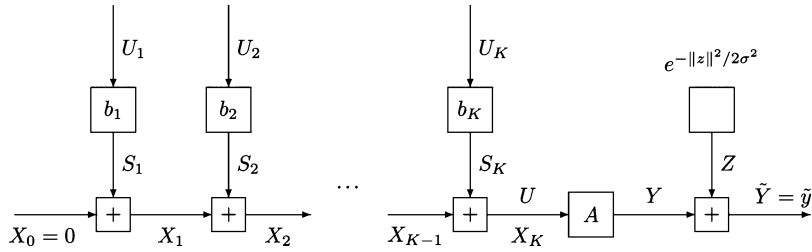


Fig. 14. Unified version of Figs. 10 and 11 with additional variables to explain the algorithms of Section V-C.

$\overleftarrow{W}_{X_k} \overleftarrow{m}_{X_k}$ and $\overleftarrow{W}_{X_k} \overleftarrow{m}_{X_k}$ according to Algorithm B, augmented by the simultaneous computation of V_{X_k} , m_{X_k} , and \tilde{W}_{X_k} as follows. From $m_{X'_k}$ and $V_{X'_k}$, we trivially obtain m_{X_k} and V_{X_k} by (II.5) and (II.6). We then obtain \tilde{W}_{X_k} (from V_{X_k} and W_{X_k}) by (I.2), and we also have $\tilde{W}_{Z'_k}$ by (II.12). We further obtain $V_{Z'_k}$ by (I.4) and $m_{Z'_k}$ by (I.6). From (III.8), we obtain $\tilde{W}_{X'_{k-1}}$. Finally, $V_{X'_{k-1}}$ and $m_{X'_{k-1}}$ are obtained by (I.4) and (I.6).

The outgoing messages $\overrightarrow{\mu}_{Y_k}$ and/or $\overleftarrow{\mu}_{U_k}$ may be obtained as described.

The following algorithm is a variation of Algorithm E that may be used for the example of Section IV-A (FIR channel equalization). [The algorithm assumes that A_k is singular and that A'_k may be grouped with B_k as in Table 6 (bottom).]

Algorithm F—All Marginals and Output Messages by Forward-Backward Propagation: Forward pass: with \overleftarrow{m}_{X_k} and V_{X_k} according to Algorithm A. Backward pass: with \overleftarrow{W}_{X_k} and $\overleftarrow{W}_{X_k} \overleftarrow{m}_{X_k}$ according to Algorithm B, augmented by the simultaneous computation of V_{X_k} , m_{X_k} , and \tilde{W}_{X_k} as follows. From $m_{X'_k}$ and $V_{X'_k}$, we trivially have m_{X_k} and V_{X_k} . By grouping A'_k with B_k as in Table 6 (bottom), we obtain m_{Z_k} and V_{Z_k} as well as m_{U_k} and V_{U_k} by (VI.5) and (VI.6). We then obtain \tilde{W}_{Z_k} (from V_{Z_k} and \overleftarrow{W}_{Z_k}) by (I.2). We then compute $\tilde{W}_{X'_{k-1}}$ and $\tilde{W}_{X'_{k-1}} m_{X'_{k-1}}$ by (III.8) and (III.7), respectively. Finally, $V_{X'_{k-1}}$ and $m_{X'_{k-1}}$ are obtained by (I.4) and (I.6).

The outgoing message $\overleftarrow{\mu}_{U_k}$ may be extracted from V_{U_k} , m_{U_k} , and $\overrightarrow{\mu}_{U_k}$.

More such algorithms seem possible and are worth exploring. For FIR channel equalization, the method described just before Algorithm E may be the most attractive.

C. Wang/Poor Algorithm

Factor graphs as in Fig. 10 and Fig. 11 give rise to yet another fundamental algorithm. The application of this algorithm to Fig. 11 (combined with the conversion from Gaussian to binary as described in Section V-D) is essentially equivalent to the algorithm by Wang and Poor in [48].

We describe the algorithm in terms of Fig. 14, which subsumes Figs. 10 and 11. Note that U_k , $1 \leq k \leq K$ are real scalars, b_k are real column vectors, and A is a real matrix. (We will assume that $A^H A$ is nonsingular, which means that the rank of A equals the number of its columns.) The column vectors S_k and X_k are defined as $S_k \triangleq b_k U_k$ and $X_k \triangleq S_k + X_{k-1}$, respectively. Note also that $X_0 = 0$ and $X_K = U$.

From the observation $\tilde{Y} = \tilde{y}$ and the incoming messages \overrightarrow{U}_k ($1 \leq k \leq K$) (with scalar mean \overrightarrow{m}_{U_k} and variance \overrightarrow{V}_{U_k}), we wish to compute the outgoing messages at U_k , i.e., the scalar means \overleftarrow{m}_{U_k} and the variances \overleftarrow{V}_{U_k} .

The pivotal quantity of the algorithm is \tilde{W}_U which, according to (56), is defined as

$$\tilde{W}_U = (\overrightarrow{V}_U + \overleftarrow{V}_U)^{-1}. \quad (64)$$

(The actual computation of \tilde{W}_U will be discussed in the following.) This quantity is useful here because, according to (II.12), it stays constant around a “+” node. In particular, we have

$$\tilde{W}_{S_k} = \tilde{W}_U \quad (65)$$

for $0 \leq k \leq K$. By means of (III.8), we then obtain \tilde{W}_{U_k} , from which the variance of the outgoing messages is easily extracted

$$\overleftarrow{V}_{U_k} = \tilde{W}_{U_k}^{-1} - \overrightarrow{V}_{U_k} \quad (66)$$

$$= (b_k^H \tilde{W}_U b_k)^{-1} - \overrightarrow{V}_{U_k}. \quad (67)$$

Note that $b_k^H \tilde{W}_U b_k$ is a scalar. Note also that, if $b_k = e_k$ as in Fig. 11, then $b_k^H \tilde{W}_U b_k$ is simply the k th diagonal element of \tilde{W}_U .

From (III.9), the mean of the outgoing messages is

$$\overleftarrow{m}_{U_k} = \tilde{W}_{U_k}^{-1} b_k^H \tilde{W}_U \overleftarrow{m}_{S_k} \quad (68)$$

$$= (b_k^H \tilde{W}_U b_k)^{-1} b_k^H \tilde{W}_U \overleftarrow{m}_{S_k} \quad (69)$$

which depends on both \tilde{W}_U and \overleftarrow{m}_{S_k} . The latter is easily obtained from (II.9), (II.10), and (III.2)

$$\overleftarrow{m}_{S_k} = \overleftarrow{m}_U - \sum_{\ell=1}^K \overrightarrow{m}_{S_\ell} + \overrightarrow{m}_{S_k} \quad (70)$$

$$= \overleftarrow{m}_U - \sum_{\ell=1}^K b_\ell \overrightarrow{m}_{U_\ell} + b_k \overrightarrow{m}_{U_k}. \quad (71)$$

Note that the first two terms on the right-hand side of (71) do not depend on k .

The mean vector \overleftarrow{m}_U , which is used in (71), may be obtained as follows. Using (III.6), we have

$$\tilde{W}_U \overleftarrow{m}_U = A^H \overleftarrow{W}_Y \overleftarrow{m}_Y \quad (72)$$

$$= A^H \sigma^{-2} \tilde{y} \quad (73)$$

and from (III.5), we have

$$\overleftarrow{W}_U = A^H \overleftarrow{W}_Y A \quad (74)$$

$$= \sigma^{-2} A^H A. \quad (75)$$

We thus obtain

$$A^H A \overleftarrow{m}_U = A^H \tilde{y} \quad (76)$$

and

$$\overleftarrow{m}_U = (A^H A)^{-1} A^H \tilde{y}. \quad (77)$$

[In an actual implementation of the algorithm, solving (76) by means of, e.g., the Cholesky factorization of $A^H A$ is usually preferable to literally computing (77).]

It remains to describe the computation of \tilde{W}_U ($= \tilde{W}_{X_k}$, $0 \leq k \leq K$). We will actually describe two methods. The first method goes as follows. From (III.8), we have

$$\tilde{W}_U = A^H \tilde{W}_Y A \quad (78)$$

$$= A^H (\overrightarrow{V}_Y + \overleftarrow{V}_Y)^{-1} A. \quad (79)$$

But $\overleftarrow{V}_Y = \sigma^2 I$ is known and \overrightarrow{V}_Y is straightforward to compute using (III.1) and (II.7)

$$\overrightarrow{V}_Y = A \overrightarrow{V}_U A^H \quad (80)$$

$$= A \left(\sum_{k=1}^K b_k \overrightarrow{V}_{U_k} b_k^H \right) A^H \quad (81)$$

$$= \sum_{k=1}^K (A b_k) \overrightarrow{V}_{U_k} (A b_k)^H. \quad (82)$$

So, evaluating (79) is a viable method to compute \tilde{W}_U .

An alternative method to compute \tilde{W}_U goes as follows. First, we compute \overleftarrow{W}_U by (75) and then we recursively compute $\overleftarrow{W}_{X_{K-1}}, \overleftarrow{W}_{X_{K-2}}, \dots, \overleftarrow{W}_{X_0}$ using (IV.6). Then, we have

$$\tilde{W}_U = \tilde{W}_{X_0} \quad (83)$$

$$= \left(\overrightarrow{V}_{X_0} + \overleftarrow{V}_{X_0} \right)^{-1} \quad (84)$$

$$= \overleftarrow{V}_{X_0}^{-1} \quad (85)$$

$$= \overleftarrow{W}_{X_0}. \quad (86)$$

D. Gaussian to Binary and Vice Versa

Linear Gaussian models are often subsystems of larger models involving discrete variables. In digital communications, for example, binary codes usually coexist with linear Gaussian channel models, cf. Figs. 9–11. In such cases, the conversion of messages from the Gaussian domain to the finite-alphabet domain and vice versa is an issue. Consider, for example, the situation in Fig. 15, where X is a $\{+1, -1\}$ -valued variable and Y is a real variable. The “=”-node in Fig. 15 denotes the factor $\delta(x - y)$, which is a Kronecker delta in x and a Dirac delta in y .

The conversion of a Gaussian message $\overrightarrow{\mu}_Y$ into a binary (“soft bit”) message $\overrightarrow{\mu}_X$ is straightforward: according to the sum-product rule, we have

$$\overrightarrow{\mu}_X(x) = \int_y \overrightarrow{\mu}_Y(y) \delta(x - y) dy \quad (87)$$

$$= \overrightarrow{\mu}_Y(x). \quad (88)$$

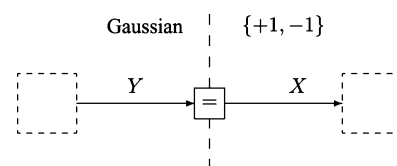


Fig. 15. Conversion of (real scalar) Gaussian to binary ($\{+1, -1\}$) variables and vice versa.

In the popular log-likelihood representation of soft-bit messages, we thus have

$$\vec{L}_X \triangleq \ln \frac{\vec{\mu}_X(+1)}{\vec{\mu}_X(-1)} \quad (89)$$

$$= 2\vec{m}_Y / \vec{\sigma}_Y^2. \quad (90)$$

In the opposite direction, an obvious and standard approach is to match the mean and the variance

$$\overleftarrow{m}_Y = \overleftarrow{m}_X \quad (91)$$

$$= \frac{\overleftarrow{\mu}_X(+1) - \overleftarrow{\mu}_X(-1)}{\overleftarrow{\mu}_X(+1) + \overleftarrow{\mu}_X(-1)} \quad (92)$$

and

$$\overleftarrow{\sigma}_Y^2 = \overleftarrow{\sigma}_X^2 \quad (93)$$

$$= 1 - \overleftarrow{m}_X^2. \quad (94)$$

It should be noted, however, that (92) and (94) need not be optimal even for graphs without cycles. An alternative way to compute a Gaussian message μ_Y is proposed in [18] and [49].

VI. BEYOND GAUSSIANS

We have seen in the previous section that, for continuous variables, working out the sum-product or max-product message computation rules for particular nodes/factors is not always trivial. In fact, literal implementation of these two basic algorithms is often infeasible when continuous variables are involved. Moreover, other algorithms may be of interest for several reasons, e.g., to yield better marginals or to guarantee convergence on graphs with cycles [14]–[21]. However, it appears that most useful algorithms for structured models with many variables can be put into message-passing form, and the factor graph approach helps us to mix and match different techniques.

A key issue with all message-passing algorithms is the representation of messages for continuous variables. In some cases, a closed family of functions with a small number of parameters works nicely, the prime example being linear Gaussian models as in Section V. However, beyond the Gaussian case, this does not seem to happen often. (An interesting exception is [27], which uses a family of Tikhonov distributions.)

In general, therefore, one has to resort to simplified messages for continuous variables.

The following message types are widely applicable.

- 1) *Quantization* of continuous variables. This approach is essentially limited to 1-D real variables.
- 2) *Single point*: the message $\mu(x)$ is replaced by a single point \hat{x} , which may be viewed as a temporary or final decision on the value of the variable X .
- 3) *Function value and derivative/gradient* at a point selected by the receiving node [50], [51] (to be described in Section VII).
- 4) *Gaussians* (cf., Section V and Appendix I).
- 5) *Gaussian mixtures*.
- 6) *List of samples*: A probability density can be represented by a list of samples. This message type allows us to describe particle filters [52], [53] as message-passing algorithms (see, e.g. [35], [36], [38], [39], [54]–[56]).
- 7) *Compound messages* consist of the “product” of other message types.

All these message types, and many different message computation rules, can coexist in large system models. The identification of suitable message types and message computation rules for particular applications remains a large area of research. Some illustrative examples will be given in the next two sections.

With such “local” approximations, and with the “global” approximation of allowing cycles in the factor graph, practical detection/estimation algorithms may be obtained for complex system models that cannot be handled by “optimal” methods.

In the next two sections, we will illustrate the use of message computation rules beyond the sum-product and max-product rules by the following example. Assume that, in some linear state-space model (as in Fig. 8), one of the matrices (A_k, B_k, C_k) is not known. In this case, this matrix becomes a variable itself. For example, if C_k is unknown, but constant over time (i.e., $C_k = C$), we obtain the factor graph of Fig. 16, which should be thought to be a part of some larger factor graph as, e.g., in Fig. 9.

The key difficulty in such cases is the multiplier node. We will outline two approaches to deal with such cases: steepest descent and (a “local” version of) expectation maximization. However, more methods are known (e.g., particle methods), and better methods may yet be found.

It should also be noted that the graph of Fig. 16 has cycles, which implies that message passing algorithms will be iterative. The convergence of such algorithms is not, in general, guaranteed, but robust (almost-sure) convergence is often observed in practice.

VII. STEEPEST DESCENT AS MESSAGE PASSING

The use of steepest descent as a “local” method in factor graphs is illustrated in Fig. 17, which represents the global function $f(\theta) \triangleq f_A(\theta)f_B(\theta)$. The variable Θ is assumed to

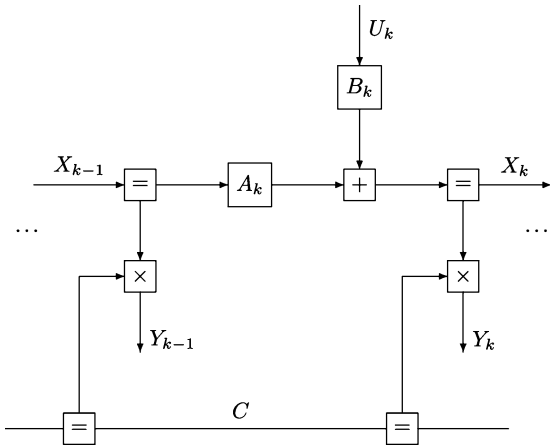


Fig. 16. Linear state-space model with unknown coefficient vector C .

take values in \mathbb{R} or in \mathbb{R}^n . Fig. 17 may be a part of some bigger factor graph, and the nodes f_A and f_B may be summaries of (i.e., messages out of) subsystems/subgraphs.

Suppose we wish to find

$$\theta_{\max} \triangleq \arg \max_{\theta} f(\theta) \quad (95)$$

by solving

$$\frac{d}{d\theta} (\ln f(\theta)) = 0. \quad (96)$$

Note that

$$\frac{d}{d\theta} (\ln f(\theta)) = \frac{f'(\theta)}{f(\theta)} \quad (97)$$

$$= \frac{f_A(\theta)f_B'(\theta) + f_B(\theta)f_A'(\theta)}{f_A(\theta)f_B(\theta)} \quad (98)$$

$$= \frac{d}{d\theta} (\ln f_B(\theta)) + \frac{d}{d\theta} (\ln f_A(\theta)). \quad (99)$$

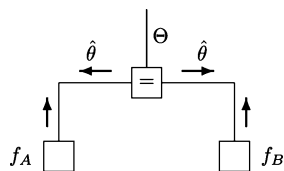


Fig. 17. Steepest descent as message passing.

The functions f_A and f_B may be infeasible to represent, or to compute, in their entirety, but it may be easy to evaluate $(d/d\theta)(\ln f_A(\theta))$ (and likewise for f_B) at any given point θ .

One method to find a solution $\hat{\theta}$ of (96) is steepest descent. The message passing view of this method can be described as follows.

- 1) An initial estimate $\hat{\theta}$ is broadcast to the nodes f_A and f_B . The node f_A replies by sending

$$\left. \frac{d}{d\theta} (\ln f_A(\theta)) \right|_{\theta=\hat{\theta}}$$

and the node f_B replies accordingly.

- 2) A new estimate $\hat{\theta}$ is computed as

$$\hat{\theta}_{\text{new}} = \hat{\theta}_{\text{old}} + s \cdot \left. \frac{d}{d\theta} (\ln f(\theta)) \right|_{\theta=\hat{\theta}_{\text{old}}} \quad (100)$$

where $s \in \mathbb{R}$ is a positive step-size parameter.

- 3) The procedure is iterated as one pleases.

As always with message-passing algorithms, there is much freedom in the scheduling of the individual operations.

The application of this method to cases as in Fig. 16 amounts to understanding its application to a multiplier node as in Fig. 18. The coefficient (or coefficient vector/matrix) C in Fig. 16 takes the role of Θ in Fig. 18.

Due to the single-point message $\hat{\theta}$, the messages along the X - and Y -edges work as if Θ were known. In particular, if the incoming messages on these edges are Gaussians, then so are the outgoing messages.

As described earlier, the outgoing message along the edge Θ is the quantity

$$\left. \frac{d}{d\theta} (\ln \bar{\mu}_{\Theta}(\theta)) \right|_{\theta=\hat{\theta}} = \left. \frac{\frac{d}{d\theta} \bar{\mu}_{\Theta}(\theta)}{\bar{\mu}_{\Theta}(\theta)} \right|_{\theta=\hat{\theta}} \quad (101)$$

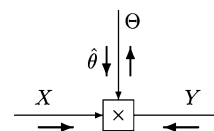


Fig. 18. Multiplier node.

where $\overleftarrow{\mu}_\Theta(\theta)$ is the sum-product (or max-product) message and

$$\overleftarrow{\mu}_\Theta(\theta) = \int_x \int_y \overrightarrow{\mu}_X(x) \overleftarrow{\mu}_Y(y) \delta(y - \theta x) dx dy \quad (102)$$

$$= \int_x \overrightarrow{\mu}_X(x) \overleftarrow{\mu}_Y(\theta x) dx. \quad (103)$$

The gradient message (101) can be evaluated in closed form, even in the vector/matrix case, if the incoming messages $\overrightarrow{\mu}_X$ and $\overleftarrow{\mu}_Y$ are both Gaussians. For the sake of clarity, we now focus on the case where Θ , X , and Y are all real-valued scalars. In this case, using

$$\frac{d}{d\theta} \overleftarrow{\mu}_Y(\theta x) = \frac{d}{d\theta} (\text{const}) \exp\left(-\frac{(\theta x - \overleftarrow{m}_Y)^2}{2\overleftarrow{\sigma}_Y^2}\right) \quad (104)$$

$$= \overleftarrow{\mu}_Y(\theta x) \left(-\frac{\theta x^2}{\overleftarrow{\sigma}_Y^2} + \frac{x\overleftarrow{m}_Y}{\overleftarrow{\sigma}_Y^2}\right) \quad (105)$$

and (103), we can write (101) as

$$\left. \frac{d}{d\theta} \overleftarrow{\mu}_\Theta(\theta) \right|_{\theta=\hat{\theta}} = \frac{\int_x \overrightarrow{\mu}_X(x) \left(\frac{d}{d\theta} \overleftarrow{\mu}_Y(\theta x) \right) dx}{\int_x \overrightarrow{\mu}_X(x) \overleftarrow{\mu}_Y(\theta x) dx} \Bigg|_{\theta=\hat{\theta}} \quad (106)$$

$$= \frac{\int_x \overrightarrow{\mu}_X(x) \overleftarrow{\mu}_Y(\hat{\theta} x) \left(-\frac{\hat{\theta} x^2}{\overleftarrow{\sigma}_Y^2} + \frac{x\overleftarrow{m}_Y}{\overleftarrow{\sigma}_Y^2} \right) dx}{\int_x \overrightarrow{\mu}_X(x) \overleftarrow{\mu}_Y(\hat{\theta} x) dx} \quad (107)$$

$$= \frac{1}{\overleftarrow{\sigma}_Y^2} \mathbb{E}_{\tilde{p}(x|\hat{\theta})} [X(\overleftarrow{m}_Y - \hat{\theta}X)]. \quad (108)$$

The expectation in (108) is with respect to the (local) probability density

$$\tilde{p}(x|\hat{\theta}) \triangleq \frac{\overrightarrow{\mu}_X(x) \overleftarrow{\mu}_Y(\hat{\theta}x)}{\int_x \overrightarrow{\mu}_X(x) \overleftarrow{\mu}_Y(\hat{\theta}x) dx} \quad (109)$$

$$\propto \exp\left(-x^2 \left(\frac{1}{2\overleftarrow{\sigma}_X^2} + \frac{\hat{\theta}^2}{2\overleftarrow{\sigma}_Y^2}\right) + x \left(\frac{\overleftarrow{m}_X}{\overleftarrow{\sigma}_X^2} + \frac{\hat{\theta}\overleftarrow{m}_Y}{\overleftarrow{\sigma}_Y^2}\right)\right) \quad (110)$$

which is a Gaussian density with mean \tilde{m} and variance $\tilde{\sigma}^2$ given by

$$\frac{1}{\tilde{\sigma}^2} = \frac{1}{\overleftarrow{\sigma}_X^2} + \frac{\hat{\theta}^2}{\overleftarrow{\sigma}_Y^2} \quad (111)$$

$$\frac{\tilde{m}}{\tilde{\sigma}^2} = \frac{\overleftarrow{m}_X}{\overleftarrow{\sigma}_X^2} + \frac{\hat{\theta}\overleftarrow{m}_Y}{\overleftarrow{\sigma}_Y^2}. \quad (112)$$

From (108), we finally obtain the outgoing gradient message (101) as

$$\left. \frac{d}{d\theta} \left(\ln \overleftarrow{\mu}_\Theta(\theta) \right) \right|_{\theta=\hat{\theta}} = \frac{\overleftarrow{m}_Y}{\overleftarrow{\sigma}_Y^2} \mathbb{E}_{\tilde{p}(x|\hat{\theta})} [X] - \frac{\hat{\theta}}{\overleftarrow{\sigma}_Y^2} \mathbb{E}_{\tilde{p}(x|\hat{\theta})} [X^2] \quad (113)$$

$$= \frac{\overleftarrow{m}_Y}{\overleftarrow{\sigma}_Y^2} \tilde{m} - \frac{\hat{\theta}}{\overleftarrow{\sigma}_Y^2} (\tilde{\sigma}^2 + \tilde{m}^2). \quad (114)$$

VIII. EXPECTATION MAXIMIZATION AS MESSAGE PASSING

Another classical method to deal with the estimation of coefficients like C in Fig. 16 is expectation maximization (EM) [57]–[59]. It turns out that EM can be put into message-passing form, where it essentially boils down to a message computation rule that differs from the sum-product and max-product rules [60].

For example, consider the factor graph of Fig. 19 with the single node/factor $g(x, y, \theta)$ (which should be considered as a part of some larger factor graph). Along the edge Θ , we receive only the single-point message $\hat{\theta}$. The messages along the X - and Y -edges are the standard sum-product messages (computed under the assumption $\Theta = \hat{\theta}$). The outgoing message along Θ is not the sum-product message, but

$$\mu_{\text{EM}}(\theta) = e^{h(\theta)} \quad (115)$$

with

$$h(\theta) \triangleq \mathbb{E}_{\tilde{p}(x,y|\hat{\theta})} [\ln g(X, Y, \theta)] \quad (116)$$

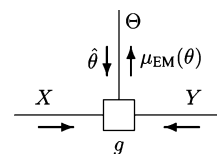


Fig. 19. Messages corresponding to expectation maximization.

where the expectation is with respect to the (local) probability density

$$\tilde{p}(x, y | \hat{\theta}) \propto g(x, y, \hat{\theta}) \overrightarrow{\mu}_X(x) \overleftarrow{\mu}_Y(y) \quad (117)$$

based on the incoming sum-product messages $\overrightarrow{\mu}_X$ and $\overleftarrow{\mu}_Y$. For the justification of (115), (116), and the corresponding perspective on the EM algorithm, the reader is referred to [60] and [61]. The main point is that the message (115) is compatible with, and may be further processed by, the sum-product or max-product algorithms.

One way to apply the general message computation rule (115), (116) to a multiplier node as in Fig. 16 is illustrated in Fig. 20. We assume that X and Θ are real (column) vectors and Y is a real scalar (as in Fig. 16). Instead of defining $g(x, y, \theta) = \delta(y - \theta^T x)$ (which turns out not to work [61]), we define

$$g(x, \theta) = \int_y \delta(y - \theta^T x) \overleftarrow{\mu}_Y(y) dy \quad (118)$$

as indicated by the dashed box in Fig. 20. (It then follows that the expectation in (116) is over X alone.) If the incoming messages $\overrightarrow{\mu}_X$ and $\overleftarrow{\mu}_Y$ are Gaussian densities, the outgoing message $\mu_{EM}(\theta)$ turns out to be a Gaussian density with inverse covariance matrix

$$\overleftarrow{W}_\Theta = \frac{V_X + m_X m_X^T}{\overleftarrow{\sigma}_Y^2} \quad (119)$$

and with mean \overleftarrow{m}_Θ given by

$$\overleftarrow{W}_\Theta \overleftarrow{m}_\Theta = \frac{m_X \overleftarrow{m}_Y}{\overleftarrow{\sigma}_Y} \quad (120)$$

(cf., [61]). The required sum-product marginals V_X and m_X may be obtained from the standard Gaussian rules (54),

(55), (III.5), and (III.6), which yield

$$V_X^{-1} = W_X = \overrightarrow{W}_X + \hat{\theta} \hat{\theta}^T / \overleftarrow{\sigma}_Y^2 \quad (121)$$

and

$$W_X m_X = \overrightarrow{W}_X \overrightarrow{m}_X + \hat{\theta} \overleftarrow{m}_Y / \overleftarrow{\sigma}_Y^2. \quad (122)$$

By replacing the unwieldy sum-product message $\overleftarrow{\mu}_\Theta$ by the Gaussian message μ_{EM} , we have thus achieved a completely Gaussian treatment of the multiplier node.

IX. CONCLUSION

The factor graph approach to signal processing involves the following steps.

- 1) Choose a factor graph to represent the system model.
- 2) Choose the message types and suitable message computation rules. Use and maintain tables of such computation rules.
- 3) Choose a message update schedule.

In this paper, we have elaborated on this approach with an emphasis on Gaussian message passing in cycle-free factor graphs of linear models, i.e., Kalman filtering and some of its ramifications. Tables of message computation rules for the building blocks of such models allow us to write down a variety of efficient algorithms without additional computations or derivations.

Beyond the Gaussian case, the factor graph approach encourages and facilitates mixing and matching different algorithmic techniques, which we have illustrated by two different approaches to deal with multiplier nodes: steepest descent and “local” expectation maximization.

The identification of suitable message types and message computation rules for continuous variables remains a large area of research. However, even the currently available tools allow us to derive practical algorithms for a wide range of nontrivial problems. ■

APPENDIX I ON GAUSSIAN DISTRIBUTIONS, QUADRATIC FORMS, AND LMMSE ESTIMATION

We briefly review some basic and well-known facts about Gaussian distributions, quadratic forms, and LMMSE estimation.

Let $F = \mathbb{R}$ or $F = \mathbb{C}$. A general Gaussian random (column) vector $X = (X_1, \dots, X_n)^T$ over F with mean vector $m = (m_1, \dots, m_n)^T \in F^n$ can be written as

$$X = AU + m \quad (123)$$

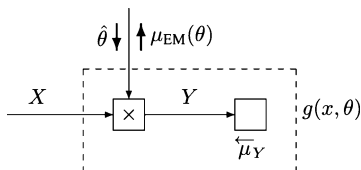


Fig. 20. Gaussian message passing through a multiplier node using EM.

where A is a nonsingular $n \times n$ matrix over F and where $U = (U_1, \dots, U_n)^T$ consists of independent F -valued Gaussian random variables U_1, \dots, U_n with mean zero and variance one. The covariance matrix of X is $V = AA^H$. The probability density of X is

$$f_X(x) \propto e^{-\beta(x-m)^H W(x-m)} \quad (124)$$

$$\propto e^{-\beta(x^H W_X - 2\text{Re}(x^H W_m))} \quad (125)$$

for $W = V^{-1} = (A^{-1})^H A^{-1}$ and with $\beta = 1/2$ in the real case ($F = \mathbb{R}$) and $\beta = 1$ in the complex case ($F = \mathbb{C}$). Conversely, any function of the form (124) with positive definite W may be obtained in this way with some suitable matrix A .

Now, let Z be a Gaussian random (column) vector, which we partition as

$$Z = \begin{pmatrix} X \\ Y \end{pmatrix} \quad (126)$$

where X and Y are themselves (column) vectors. The density of Z is $f_Z(z) \propto e^{-\beta q(x,y)}$ with

$$q(x,y) = ((x - m_X)^H, (y - m_Y)^H) \begin{pmatrix} W_X & W_{XY} \\ W_{YX} & W_Y \end{pmatrix} \begin{pmatrix} x - m_X \\ y - m_Y \end{pmatrix} \quad (127)$$

with positive definite W_X and W_Y and with $W_{YX} = W_{XY}^H$.

For fixed y , considered as a function of x alone, (127) becomes

$$\begin{aligned} q(x,y) &= x^H W_X x \\ &\quad - 2\text{Re}(x^H W_X (m_X - W_X^{-1} W_{XY} (y - m_Y))) \\ &\quad + \text{const.} \end{aligned} \quad (128)$$

Comparing this with (125) yields the following theorem.

Theorem 1 (Gaussian Conditioning Theorem): If X and Y are jointly Gaussian with joint distribution $\propto e^{-\beta q(x,y)}$ as above, then *conditioned on* $Y = y$ (for any fixed y), X is Gaussian with mean

$$E[X|Y = y] = m_X - W_X^{-1} W_{XY} (y - m_Y) \quad (129)$$

and covariance matrix W_X^{-1} . \square

Note that $E[X|Y = y]$ is both the MAP estimate and the MMSE estimate of X given the observation $Y = y$.

According to (129), $E[X|Y = y]$ is an *affine* (= linear with offset) function of the observation y . We thus have the following theorem.

Theorem 2: For jointly Gaussian random variables or vectors X and Y , the MAP estimate of X from the observation $Y = y$ is an affine function of y and coincides both with the MMSE estimate and the LMMSE estimate. \square

Note that, in this theorem as well as in the following theorem, the ‘‘L’’ in LMMSE must be understood as ‘‘affine’’ (= linear with offset).

Theorem 3 (LMMSE via Gaussian MAP Estimation): Let X and Y be random variables (or vectors) with arbitrary distributions but with finite means and with finite second-order moments. Then, the LMMSE estimate of X based on the observation $Y = y$ may be obtained by pretending that X and Y are jointly Gaussian (with their actual means and second-order moments) and forming the corresponding MAP estimate. \square

The proof follows from noting that, according to the orthogonality principle [47], the LMMSE estimate of X based on $Y = y$ depends only on the means and second-order moments.

In a different direction, we also note the following fact.

Theorem 4 (Gaussian Max/Int Theorem): Let $q(x,y)$ be a quadratic form as in (127) with W_X positive definite. Then

$$\int_{-\infty}^{\infty} e^{-q(x,y)} dx \propto \max_x e^{-q(x,y)} \quad (130)$$

$$= e^{-\min_x q(x,y)}. \quad (131)$$

\square

Note that the theorem still holds if $q(x,y)$ is replaced with $\beta q(x,y)$ for any positive real β .

Proof: We first note the following fact. If W is a positive definite matrix and

$$\tilde{q}(x) \triangleq (x - m)^H W (x - m) + c \quad (132)$$

$$= x^H W x - 2\text{Re}(x^H W m) + m^H W m + c \quad (133)$$

then

$$\int_{-\infty}^{\infty} e^{-\tilde{q}(x)} dx = e^{-c} \int_{-\infty}^{\infty} e^{-(\tilde{q}(x)-c)} dx \quad (134)$$

$$= e^{-c} \int_{-\infty}^{\infty} e^{-(x-m)^H W (x-m)} dx \quad (135)$$

$$= e^{-c} \int_{-\infty}^{\infty} e^{-x^H W x} dx \quad (136)$$

$$= e^{-\min_x \tilde{q}(x)} \int_{-\infty}^{\infty} e^{-x^H W x} dx. \quad (137)$$

Now consider (127) as a function of x with parameter y , as in (128). This function is of the form (133) with W_X taking the role of W . It thus follows from (137) that

$$\int_{-\infty}^{\infty} e^{-q(x,y)} dx = e^{-\min_x q(x,y)} \int_{-\infty}^{\infty} e^{-x^H W_X x} dx. \quad (138)$$

But the integral on the right-hand side does not depend on y , which proves (131). ■

The minimization in (131) is given by the following theorem.

Theorem 5 (Quadratic-Form Minimization): Let $q(x, y)$ be defined as in (127). Then

$$\min_x q(x, y) = (y - m_Y)^H (W_Y - W_{XY} W_X^{-1} W_{XY}) (y - m_Y). \quad (139)$$

□

This may be proved by noting from (128) or (129) that

$$\arg \min_x q(x, y) = m_X - W_X^{-1} W_{XY} (y - m_Y). \quad (140)$$

Plugging this into (127) yields (139).

Finally, we also have the following fact.

Theorem 6 (Sum of Quadratic Forms): Let both A and B be positive semi-definite matrices. Then

$$\begin{aligned} (x - a)^H A (x - a) + (x - b)^H B (x - b) \\ = x^H W x - 2 \operatorname{Re}(x^H W m) + m^H W m + c \end{aligned} \quad (141)$$

with

$$W = A + B \quad (142)$$

$$W m = A a + B b \quad (143)$$

$$m = (A + B)^{\#} (A a + B b) \quad (144)$$

and with the scalar

$$c = (a - b)^H A (A + B)^{\#} B (a - b). \quad (145)$$

□

The verification of (142) and (143) is straightforward. A proof of (144) and (145) may be found in [33].

APPENDIX II PROOFS OF TABLES 1–6

Proof of (I.1): From (56) and (54), we have

$$\tilde{W}_X^{-1} = \vec{V}_X + \overleftarrow{V}_X \quad (146)$$

$$= \overleftarrow{V}_X (\vec{W}_X + \overleftarrow{W}_X) \vec{V}_X \quad (147)$$

$$= \overleftarrow{V}_X W_X \vec{V}_X \quad (148)$$

and thus

$$\tilde{W}_X = (\overleftarrow{V}_X W_X \vec{V}_X)^{-1} \quad (149)$$

$$= \overleftarrow{W}_X \overleftarrow{V}_X \vec{W}_X. \quad (150)$$

Proof of (I.2): From (150) and (54), we have

$$\tilde{W}_X = \overleftarrow{W}_X \overleftarrow{V}_X (W_X - \overleftarrow{W}_X) \quad (151)$$

$$= \overleftarrow{W}_X - \overleftarrow{W}_X \overleftarrow{V}_X \vec{W}_X. \quad (152)$$

Proof of (I.3) and (I.4): Are analogous to the proofs of (I.1) and (I.2), respectively.

Proof of (I.5): Follows from multiplying both sides of (55) by V_X .

Proof of (I.6): Using (I.4), we have

$$\vec{V}_X \vec{W}_X \vec{m}_X = (\vec{V}_X - \overleftarrow{V}_X \overleftarrow{W}_X \vec{V}_X) \vec{W}_X \vec{m}_X \quad (153)$$

$$= \vec{m}_X - \overleftarrow{V}_X \overleftarrow{W}_X \vec{m}_X. \quad (154)$$

Inserting this into (I.5) yields (I.6).

Proof of (II.1) and (II.3): From the sum-product rule, and we immediately have

$$\vec{\mu}_Z(z) = \iint_{x \ y} \vec{\mu}_X(x) \vec{\mu}_Y(y) \delta(x-z) \delta(y-z) dx dy \quad (155)$$

$$= \vec{\mu}_X(z) \vec{\mu}_Y(z). \quad (156)$$

Plugging in

$$\vec{\mu}_X(x) \propto e^{-\beta(x-\vec{m}_X)^H \vec{W}_X(x-\vec{m}_X)} \quad (157)$$

(and analogously for $\vec{\mu}_Y$) and then using Theorem 6 yields

$$\vec{\mu}_Z(z) \propto e^{-\beta(z-\vec{m}_Z)^H \vec{W}_Z(z-\vec{m}_Z)} \quad (158)$$

with \vec{W}_Z and $\vec{W}_Z \vec{m}_Z$ as in (II.1) and (II.3), respectively.

Proof of (II.5) and (II.6): The proof follows from the fact that the marginals at all three edges coincide:

$$\vec{\mu}_X(s) \overleftarrow{\mu}_X(s) = \vec{\mu}_X(s) \vec{\mu}_Y(s) \overleftarrow{\mu}_Z(s) \quad (159)$$

$$= \vec{\mu}_Y(s) \overleftarrow{\mu}_Y(s) \quad (160)$$

$$= \vec{\mu}_Z(s) \overleftarrow{\mu}_Z(s). \quad (161)$$

Proof of (II.7) and (II.9): The computation of $\vec{\mu}_Z$ amounts to closing the box in the factor graph of Fig. 21. In this figure, by elementary probability theory, the mean of Z is the sum of the means of X and Y , and the variance of Z is the sum of the variances of X and Y .

Proof of (II.12): From (II.7), we have

$$\vec{V}_Z + \overleftarrow{V}_Z = (\vec{V}_X + \overleftarrow{V}_Y) + \overleftarrow{V}_Z \quad (162)$$

and from (II.8) we have

$$\vec{V}_X + \overleftarrow{V}_X = \vec{V}_X + (\overleftarrow{V}_Y + \overleftarrow{V}_Z) \quad (163)$$

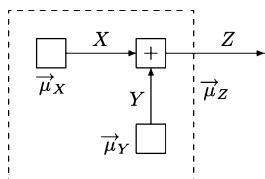


Fig. 21. Proof of (II.7) and (II.9).

$$\overleftarrow{V}_Y + \overleftarrow{V}_Y = \overleftarrow{V}_Y + (\overleftarrow{V}_X + \overleftarrow{V}_Z). \quad (164)$$

We thus have

$$\overleftarrow{V}_X + \overleftarrow{V}_X = \overleftarrow{V}_Y + \overleftarrow{V}_Y = \overleftarrow{V}_Z + \overleftarrow{V}_Z \quad (165)$$

which by (56) implies (II.12).

Proof of (III.1)–(III.4): By elementary probability theory.

Proof of (III.5) and (III.6): By the sum-product rule, we have

$$\overleftarrow{\mu}_X(x) = \int_y \delta(y - Ax) \overleftarrow{\mu}_Y(y) dy \quad (166)$$

$$= \overleftarrow{\mu}_Y(Ax) \quad (167)$$

$$\propto e^{-\beta(Ax - \vec{m}_Y)^H \vec{W}_Y(Ax - \vec{m}_Y)} \quad (168)$$

$$\propto e^{-\beta(x^H A^H \vec{W}_Y A x - 2\text{Re}(x^H A^H \vec{W}_Y \vec{m}_Y))} \quad (169)$$

and comparison with (125) completes the proof.

Proof of (III.8): Using (I.2), (III.5), and (III.4), we have

$$\tilde{W}_X = \overleftarrow{W}_X - \overleftarrow{W}_X V_X \overleftarrow{W}_X \quad (170)$$

$$= A^H \overleftarrow{W}_Y A - A^H \overleftarrow{W}_Y A V_X A^H \overleftarrow{W}_Y A \quad (171)$$

$$= A^H (\overleftarrow{W}_Y - \overleftarrow{W}_Y V_Y \overleftarrow{W}_Y) A \quad (172)$$

$$= A^H \tilde{W}_Y A. \quad (173)$$

Proof of (III.7): Using (III.8) and (III.3), we have

$$\tilde{W}_X m_X = A^H \tilde{W}_Y A m_X \quad (174)$$

$$= A^H \tilde{W}_Y m_Y. \quad (175)$$

Proof of (III.9): Using (I.2), (III.6), (III.5), and (III.4), we have

$$\tilde{W}_X \overleftarrow{m}_X = (\overleftarrow{W}_X - \overleftarrow{W}_X V_X \overleftarrow{W}_X) \overleftarrow{m}_X \quad (176)$$

$$= A^H \overleftarrow{W}_Y \overleftarrow{m}_Y - A^H \overleftarrow{W}_Y A V_X A^H \overleftarrow{W}_Y \overleftarrow{m}_Y \quad (177)$$

$$= A^H (\overleftarrow{W}_Y - \overleftarrow{W}_Y V_Y \overleftarrow{W}_Y) \overleftarrow{m}_Y \quad (178)$$

$$= A^H \tilde{W}_Y \overleftarrow{m}_Y. \quad (179)$$

We will now need the following well known fact.

Theorem 7 (Matrix Inversion Lemma [47]): Assume that the (real or complex) matrices A, B, C, D satisfy

$$A = B^{-1} + CD^{-1}C^H \quad (180)$$

and assume that both B and D are positive definite. Then A is positive definite and

$$A^{-1} = B - BC(D + C^HBC)^{-1}C^HB. \quad (181)$$

□

Proof of (IV.2): By (II.1) and (III.5), we have

$$\vec{W}_Z = \vec{W}_X + A^H \vec{W}_Y A. \quad (182)$$

Using the Matrix Inversion Lemma (Theorem 7 above) yields

$$\vec{V}_Z = \vec{V}_X - \vec{V}_X A^H (\vec{V}_Y + A \vec{V}_X A^H)^{-1} A \vec{V}_X. \quad (183)$$

Proof of (IV.1): By (II.3) and (III.6), we have

$$\vec{m}_Z = \vec{W}_Z^{-1} (\vec{W}_X \vec{m}_X + A^H \vec{W}_Y \vec{m}_Y) \quad (184)$$

$$= \vec{V}_Z \vec{V}_X^{-1} (\vec{m}_X + \vec{V}_X A^H \vec{W}_Y \vec{m}_Y). \quad (185)$$

Inserting (183) yields

$$\vec{m}_Z = \left(I - \vec{V}_X A^H (\vec{V}_Y + A \vec{V}_X A^H)^{-1} A \right) (\vec{m}_X + \vec{V}_X A^H \vec{W}_Y \vec{m}_Y) \quad (186)$$

$$= \vec{m}_X + \vec{V}_X A^H \vec{W}_Y \vec{m}_Y - \vec{V}_X A^H \cdot (\vec{V}_Y + A \vec{V}_X A^H)^{-1} A (\vec{m}_X + \vec{V}_X A^H \vec{W}_Y \vec{m}_Y) \quad (187)$$

$$= \vec{m}_X + \vec{V}_X A^H (\vec{V}_Y + A \vec{V}_X A^H)^{-1} \cdot \left((\vec{V}_Y + A \vec{V}_X A^H) \vec{W}_Y \vec{m}_Y - (A \vec{m}_X + A \vec{V}_X A^H \vec{W}_Y \vec{m}_Y) \right) \quad (188)$$

$$= \vec{m}_X + \vec{V}_X A^H (\vec{V}_Y + A \vec{V}_X A^H)^{-1} (\vec{m}_Y - A \vec{m}_X). \quad (189)$$

Proof of (IV.6): By (II.7) and (III.1), we have

$$\vec{V}_Z = \vec{V}_X + A \vec{V}_Y A^H. \quad (190)$$

Using the Matrix Inversion Lemma (Theorem 7 above) yields

$$\vec{W}_Z = \vec{W}_X - \vec{W}_X A (\vec{W}_Y + A^H \vec{W}_X A)^{-1} A^H \vec{W}_X. \quad (191)$$

Proof of (IV.4): By (II.9) and (III.2).

Proof of (IV.8): From (IV.4) and (IV.6), we have

$$\vec{W}_Z \vec{m}_Z = \vec{W}_Z (\vec{m}_X + A \vec{m}_Y) \quad (192)$$

$$= (\vec{W}_X - \vec{W}_X A A^H \vec{W}_X) (\vec{m}_X + A \vec{m}_Y) \quad (193)$$

$$= \vec{W}_X \vec{m}_X + \vec{W}_X A^H \cdot (H^{-1} \vec{m}_Y - A^H \vec{W}_X \vec{m}_X - A^H \vec{W}_X A \vec{m}_Y) \quad (194)$$

$$= \vec{W}_X \vec{m}_X + \vec{W}_X A^H (\vec{W}_Y \vec{m}_Y - A^H \vec{W}_X \vec{m}_X). \quad (195)$$

Proof of Table 5 (Top): Let U^\perp be the kernel of the surjective mapping $\varphi : x \mapsto Ax$ and let U be its orthogonal complement (in the space of x). Let $x = x_U + x_{U^\perp}$ be the decomposition of x into $x_U \in U$ and $x_{U^\perp} \in U^\perp$. The condition $y = Ax$ is equivalent to the condition

$$x_U = A^\# y. \quad (196)$$

We next note that U is spanned by the columns of A^H and that U^\perp is spanned by the columns of B^H . It follows that (196) is equivalent to

$$x = A^\# y + B^H (\text{arbitrary}). \quad (197)$$

Proof of Table 5 (Bottom): Let U be the image of the injective mapping $\varphi : x \mapsto Ax$ and let U^\perp be its orthogonal complement (in the space of y). Let $y = y_U + y_{U^\perp}$ be the decomposition of y into $y_U \in U$ and $y_{U^\perp} \in U^\perp$. The condition $y = Ax$ is equivalent to the two conditions

$$x = A^\# y \quad (198)$$

and

$$y_{U^\perp} = 0. \quad (199)$$

We next note that U^\perp is the kernel of the mapping $y \mapsto A^\# y$ and U is the kernel of the mapping $y \mapsto B^H y$. It follows that (199) is equivalent to $B^H y = 0$.

Proofs of (VI.1) and (VI.2): Are immediate from (III.2) and (III.1), respectively.

Proofs of (VI.3) and (VI.4): Are immediate from (III.6) and (III.5), respectively.

Proofs of (VI.5) and (VI.6): Are immediate from (III.2) and (III.1), respectively.

APPENDIX III FOURIER TRANSFORM ON FACTOR GRAPHS

We review some key results of Forney [43] and adapt (and simplify) them to the setting of the present paper.

The Fourier transform of a function $f: \mathbb{R}^n \rightarrow \mathbb{C}$ is the function $\tilde{f}: \mathbb{R}^n \rightarrow \mathbb{C}$ given by

$$\tilde{f}(\omega_1, \dots, \omega_n) = \int_{x_1} \dots \int_{x_n} f(x_1, \dots, x_n) e^{-i\omega_1 x_1} \dots e^{-i\omega_n x_n} dx_1 \dots dx_n. \quad (200)$$

With $x = (x_1, \dots, x_n)^T$ and $\omega = (\omega_1, \dots, \omega_n)^T$, this may be written as

$$\tilde{f}(\omega) = \int_x f(x) e^{-i\omega^T x} dx. \quad (201)$$

The function f may be recovered from its Fourier transform \tilde{f} by means of the inverse Fourier transform

$$f(x) = (2\pi)^{-n} \int_x \tilde{f}(\omega) e^{i\omega^T x} d\omega. \quad (202)$$

We will use both $\mathcal{F}(f)$ and \tilde{f} (as above) to denote the Fourier transform of f .

It is immediate from (200) that

$$\mathcal{F}\left(\int_{x_3} f(x_1, x_2, x_3) dx_3\right) = \tilde{f}(\omega_1, \omega_2, \omega_3)|_{\omega_3=0}. \quad (203)$$

In words: marginalization in one domain corresponds to zeroing in the other domain.

The Fourier transform of a 1-D real zero-mean Gaussian density is

$$\mathcal{F}\left(\frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}\right) = e^{-\omega^2\sigma^2/2}. \quad (204)$$

The Fourier transform of a general n -dimensional real Gaussian density with covariance matrix $V = W^{-1}$ is

$$\mathcal{F}\left(\gamma e^{-\frac{1}{2}(x-m)^T W (x-m)}\right) = e^{-i\omega^T m} e^{-\frac{1}{2}\omega^T V \omega} \quad (205)$$

(where γ is the required scale factor to normalize the density).

The Fourier transform may be carried out directly in a factor graph as follows. Consider a factor graph for (some factorization of) $f(x_1, \dots, x_n)$ such that x_1, \dots, x_k are represented by half-edges and x_{k+1}, \dots, x_n are represented by full edges. Create a new factor graph (with the same topology) by the following procedure.

- 1) Replace each variable x_ℓ by its dual (“frequency”) variable ω_ℓ .
- 2) Replace each node/factor by its Fourier transform.
- 3) For each full edge, introduce a minus sign into one of the adjacent factors.

The resulting factor graph will be called a dual factor graph of the original factor graph. (The dual factor graph is not unique due to the sign issue in step 3.)

Theorem 8 (Fourier Transform on Factor Graph): Let $\tilde{f}'(\omega_1, \dots, \omega_n)$ be the global function of a dual factor graph. Then, the Fourier transform of

$$\int_{x_{k+1}} \dots \int_{x_n} f(x_1, \dots, x_n) dx_{k+1} \dots dx_n \quad (206)$$

is

$$\int_{\omega_{k+1}} \dots \int_{\omega_n} \tilde{f}'(\omega_1, \dots, \omega_n) d\omega_{k+1} \dots d\omega_n \quad (207)$$

(up to a scale factor). \square

Note that the factor graph is allowed to have cycles. Note also that \tilde{f}' is not the Fourier transform of f ; only the half-edge marginals (206) and (207) are a Fourier pair (up to a scale factor).

Example 1: Let

$$f(x_1, x_2, x_3) = g_1(x_1, x_3) g_2(x_2, x_3) \quad (208)$$

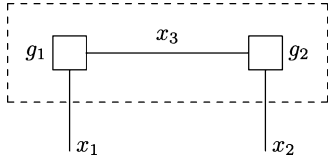


Fig. 22. Factor graph of (208).

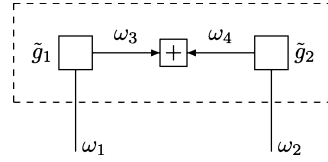


Fig. 24. Factor graph of (210), which is equivalent to Fig. 23.

the factor graph of which is shown in Fig. 22. Then

$$\begin{aligned} \tilde{f}'(\omega_1, \omega_2, \omega_3) &= \tilde{g}_1(\omega_1, \omega_3)\tilde{g}_2(\omega_2, -\omega_3) \end{aligned} \quad (209)$$

$$= \int_{\omega_4} \tilde{g}_1(\omega_1, \omega_3)\tilde{g}_2(\omega_2, \omega_4)\delta(\omega_3 + \omega_4)d\omega_4 \quad (210)$$

as shown in Figs. 23 and 24. Theorem 8 now claims that the Fourier transform of

$$f(x_1, x_2) \triangleq \int_{x_3} f(x_1, x_2, x_3)dx_3 \quad (211)$$

is

$$\tilde{f}(\omega_1, \omega_2) \propto \int_{\omega_3} \tilde{f}'(\omega_1, \omega_2, \omega_3)d\omega_3 \quad (212)$$

the “closed-box” global function in Figs. 23 and 24. \square

An outline of the proof of Theorem 8 goes as follows. We first note that the Fourier transform of

$$f(x_1, x_2, x_5) \triangleq g_1(x_1, x_5)g_2(x_2, x_5) \quad (213)$$

is the convolution

$$\begin{aligned} \tilde{f}(\omega_1, \omega_2, \omega_5) &= \frac{1}{2\pi} \int_{\omega_3} \int_{\omega_4} \tilde{g}_1(\omega_1, \omega_3)\tilde{g}_2(\omega_2, \omega_4) \\ &\quad \cdot \delta(\omega_3 + \omega_4 - \omega_5)d\omega_4d\omega_5 \end{aligned} \quad (214)$$

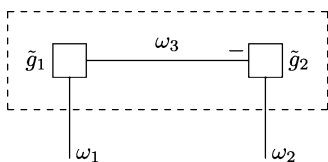


Fig. 23. Factor graph of (209), a dual factor graph of Fig. 22.

(see Figs. 25 and 26), which may be verified by direct calculation of the inverse Fourier transform of (214). Inserting $\omega_5 = 0$ changes Fig. 26 into Fig. 24 and Fig. 25 into Fig. 22 [the latter by (203)]. We have thus established that the Fourier transform of Fig. 22 is indeed Fig. 24. But the proof of this simple example is easily generalized to general factor graphs by treating all full edges simultaneously like the edge x_3 in Fig. 22 and using the corresponding generalization of (214).

We will use the following generalization of the Dirac delta. If V is a subspace of \mathbb{R}^n , we define δ_V by $\delta_V(x) = 0$ for $x \notin V$ and

$$\int_x f(x)\delta_V(x)dx \triangleq \int_V f(x)dx \quad (215)$$

for any integrable function f . For $\dim(V) = 0$, we define $\delta_V(x) = \delta(x)$, the Dirac delta.

Theorem 9 (Fourier Transform of Linear Constraints): Let V be a subspace of \mathbb{R}^n and let V^\perp be its orthogonal complement. Then

$$\mathcal{F}(\delta_V(x)) \propto \delta_{V^\perp}(\omega). \quad (216)$$

\square

Proof: The main idea is to write δ_V as a limit of a Gaussian distribution. Let k be the dimension of V . Let A_0 be an $n \times k$ matrix whose columns form an orthonormal basis of V , and let A_1 be an $n \times (n - k)$ matrix whose columns form an orthonormal basis of V^\perp . Note that $\|A_0^T x\|$ is the norm of the projection of x into V and $\|A_1^T x\|$ is the

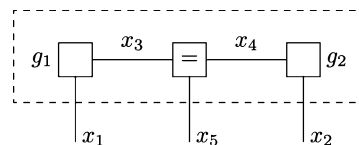
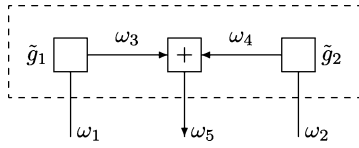


Fig. 25. Factor graph of (213).


Fig. 26. Fourier transform of Fig. 25 (up to scale factor).

norm of the projection of x into V^\perp . Let $A = (A_0, A_1)$, which is a nonsingular $n \times n$ matrix. Then

$$\delta_V(x) \propto \lim_{\beta \rightarrow \infty} e^{-\frac{\beta}{2} \|A_1^T x\|^2} \quad (217)$$

$$= \lim_{\beta \rightarrow \infty} e^{-\frac{1}{2\beta} \|A_0^T x\|^2 - \frac{\beta}{2} \|A_1^T x\|^2} \quad (218)$$

$$= \lim_{\beta \rightarrow \infty} e^{-\frac{1}{2\beta} x^T A_0 A_0^T x - \frac{\beta}{2} x^T A_1 A_1^T x} \quad (219)$$

$$= \lim_{\beta \rightarrow \infty} e^{-\frac{1}{2} x^T A D A^T x} \quad (220)$$

with

$$D \triangleq \text{diag}(\underbrace{1/\beta, \dots, 1/\beta}_{k \text{ times}}, \underbrace{\beta, \dots, \beta}_{n-k \text{ times}}). \quad (221)$$

Analogously

$$\delta_{V^\perp}(x) \propto \lim_{\beta \rightarrow \infty} e^{-\frac{1}{2} x^T A D^{-1} A^T x} \quad (222)$$

$$= \lim_{\beta \rightarrow \infty} e^{-\frac{1}{2} x^T (A D A^T)^{-1} x} \quad (223)$$

where the last step follows from noting that $A^T A = I$ and thus $A^{-1} = A^T$. The theorem then follows from noting that (220) and (223) are a Fourier pair (up to a scale factor), as is obvious from (205). ■

Example 2 (Equality Constraint): Let

$$V = \{(x_1, x_2, x_3)^T \in \mathbb{R}^3 : x_1 = x_2 = x_3\}. \quad (224)$$

Then

$$V^\perp = \{(x_1, x_2, x_3)^T \in \mathbb{R}^3 : x_1 + x_2 + x_3 = 0\}. \quad (225)$$

It follows that the Fourier transform of

$$\delta_V(x) = \delta(x_1 - x_2)\delta(x_2 - x_3) \quad (226)$$

is the sum constraint

$$\delta_{V^\perp}(\omega) = \delta(\omega_1 + \omega_2 + \omega_3) \quad (227)$$

(up to a scale factor). □

With the Fourier transform of this example, applying Theorem 8 to Fig. 25 yields Fig. 26.

Example 3 (Matrix Multiplication): Let A be a real matrix (without any conditions on rank or aspect ratio). Let

$$V = \{(x_1^T, x_2^T)^T : x_1 = A x_2\} \quad (228)$$

where x_1 and x_2 are column vectors of suitable dimensions. Then

$$V^\perp = \{(x_1^T, x_2^T)^T : x_2 = -A^T x_1\}. \quad (229)$$

It follows that the Fourier transform of

$$\delta_V(x) = \delta(x_1 - A x_2) \quad (230)$$

is

$$\delta_{V^\perp}(\omega) = \delta(\omega_2 + A^T \omega_1) \quad (231)$$

(up to a scale factor). □

It follows from Examples 2 and 3 that the two factor graphs in each of the Tables 2 and 4–6 are Fourier pairs (up to a scale factor).

Acknowledgment

The material of this paper has grown over many years into its present shape. The first author wishes to particularly acknowledge the enjoyable collaborations with N. Wiberg and with P. O. Vontobel. Both the first and the last author are immensely grateful to G. D. Forney, Jr., for his continued encouragement and feedback. S. Korl and H.-A. Loeliger are indebted to A. Lindgren for many helpful discussions. L. Ping wishes to acknowledge helpful discussions with Q. Guo.

REFERENCES

- [1] B. J. Frey, F. R. Kschischang, H.-A. Loeliger, and N. Wiberg, "Factor graphs and algorithms," in *Proc. 35th Allerton Conf. Communications, Control, and Computing*, Monticello, IL, Sep. 29–Oct. 1, 1997, pp. 666–680.
- [2] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.
- [3] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Proc. Mag.*, pp. 28–41, Jan. 2004.
- [4] M. I. Jordan, "Graphical models," *Statistical Sci.*, vol. 19, no. 1, pp. 140–155, 2004.
- [5] M. Jordan, Ed., *Learning in Graphical Models*. New York: Kluwer, 1998.
- [6] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter, *Probabilistic Networks and Expert Systems*. New York: Springer, 1999.
- [7] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2006.
- [8] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, 2nd ed. San Mateo, CA: Morgan Kaufmann, 1988.
- [9] R. Kindermann and J. L. Snell, *Markov Random Fields and Their Applications*. Providence, RI: Amer. Math. Soc., 1980.
- [10] S. M. Aji and R. J. McEliece, "The generalized distributive law," *IEEE Trans. Inform. Theory*, vol. 46, no. 2, pp. 325–343, Mar. 2000.
- [11] S. M. Aji and R. J. McEliece, "The generalized distributive law and free energy minimization," in *Proc. 39th Allerton Conf. Communications, Control, and Computing*, Monticello, IL, Oct. 2001, pp. 672–681.
- [12] Y. Weiss and W. T. Freeman, "On the optimality of the max-product belief propagation algorithm in arbitrary graphs," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 736–744, Feb. 2001.
- [13] P. Rusevchientong and B. Van Roy, "An analysis of belief propagation on the turbo decoding graph with Gaussian densities," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 745–765, Feb. 2001.
- [14] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Generalized belief propagation," *Advances Neural Information Processing Systems (NIPS)*, vol. 13, pp. 689–695, Dec. 2000.
- [15] J. Feldman, D. Karger, and M. J. Wainwright, "LP decoding," in *Proc. 41st Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Oct. 1–3, 2003.
- [16] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Constructing free-energy approximations and generalized belief propagation algorithms," *IEEE Trans. Inform. Theory*, vol. 51, no. 7, pp. 2282–2312, Jul. 2005.
- [17] V. Kolmogorov and M. J. Wainwright, "On the optimality of tree-reweighted max-product message-passing," in *Proc. 21st Conf. Uncertainty in Artificial Intelligence*, Edinburgh, Scotland, Jul. 2005.
- [18] T. P. Minka, "A family of algorithms for approximate Bayesian inference," Ph.D. dissertation, Massachusetts Inst. Technology (MIT), Cambridge, 2001.
- [19] T. P. Minka, "Expectation propagation for approximate Bayesian inference," in *Proc. 17th Annu. Conf. Uncertainty in Artificial Intelligence (UAI-01)*, 2001, vol. 17, pp. 362–369.
- [20] J. Winn and C. Bishop, "Variational message passing," *J. Machine Learning Res.*, vol. 6, pp. 661–694, 2005.
- [21] M. Chertkov and V. Y. Chernyak, "Loop series for discrete statistical models on graphs," *J. Statistical Mechanics: Theory and Experiment*, Jun. 2006.
- [22] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation 440, Univ. Linköping, Linköping, Sweden, 1996, Linköping Studies in Science and Technology.
- [23] A. P. Worthen and W. E. Stark, "Unified design of iterative receivers using factor graphs," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 843–849, Feb. 2001.
- [24] J. Boutros and G. Caire, "Iterative multiuser joint decoding: Unified framework and asymptotic analysis," *IEEE Trans. Inform. Theory*, vol. 48, no. 7, pp. 1772–1793, Jul. 2002.
- [25] E. Biglieri, A. Nordin, and G. Taricco, "Iterative receivers for coded MIMO signaling," *Wirel. Commun. Mob. Comput.*, vol. 4, no. 7, pp. 697–710, Nov. 2004.
- [26] G. Colavolpe and G. Germe, "On the application of factor graphs and the sum-product algorithm to ISI channels," *IEEE Trans. Commun.*, vol. 53, no. 5, pp. 818–825, May 2005.
- [27] G. Colavolpe, A. Barbieri, and G. Caire, "Algorithms for iterative decoding in the presence of strong phase noise," *IEEE J. Select. Areas Commun.*, vol. 23, no. 9, pp. 1748–1757, Sep. 2005.
- [28] H. Niu, M. Shen, J. A. Ritcey, and H. Liu, "A factor graph approach to iterative channel estimation and LDPC decoding over fading channels," *IEEE Trans. Wireless Commun.*, vol. 4, no. 4, pp. 1345–1350, Jul. 2005.
- [29] A. W. Eckford, "The factor graph EM algorithm: Applications for LDPC codes," in *Proc. 6th Workshop Signal Proc. Advances in Wireless Communications*, Jun. 5–8, 2005, pp. 910–914.
- [30] Q. Guo, L. Ping, and H.-A. Loeliger, "Turbo equalization based on factor graphs," in *Proc. IEEE Int. Symp. Information Theory*, Adelaide, Australia, Sep. 4–9, 2005, pp. 2021–2025.
- [31] F. Simoons and M. Moeneclaey, "Code-aided estimation and detection on time-varying correlated MIMO channels: A factor graph approach," *EURASIP J. Applied Signal Processing*, vol. 2006, no. 1, pp. 1–11, 2006.
- [32] R. Drost and A. C. Singer, "Factor graph algorithms for equalization," *IEEE Trans. Signal Processing*, to appear.
- [33] H.-A. Loeliger, "Least squares and Kalman filtering on Forney graphs," in *Codes, Graphs, and Systems*, R. E. Blahut and R. Koetter, Eds. New York: Kluwer, 2002, pp. 113–135, (Festschrift in Honor of David Forney on the Occasion of his 60th Birthday).
- [34] H.-A. Loeliger, J. Dauwels, V. M. Koch, and S. Kori, "Signal processing with factor graphs: Examples," in *Proc. First Int. Symp. Control, Communications and Signal Processing*, Hammamet, Tunisia, Mar. 21–24, 2004, pp. 571–574.
- [35] S. Kori, H.-A. Loeliger, and A. G. Lindgren, "AR model parameter estimation: From factor graphs to algorithms," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Montreal, Canada, May 17–21, 2004, vol. V, pp. 509–512.
- [36] J. Dauwels and H.-A. Loeliger, "Phase estimation by message passing," in *Proc. IEEE Int. Conf. Communications*, Paris, France, Jun. 20–24, 2004, pp. 523–527.
- [37] V. M. Koch and H.-A. Loeliger, "EMG signal decomposition by loopy belief propagation," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Philadelphia, PA, Mar. 18–23, 2005, pp. 397–400.
- [38] S. Kori, "A factor graph approach to signal modelling, system identification and filtering," Ph.D. dissertation 16170, ETH Zurich, Switzerland, 2005.
- [39] J. Dauwels, "On graphical models for communications and machine learning: Algorithms, bounds, and analog implementation," Ph.D. dissertation 16365, ETH Zurich, Switzerland, 2005.
- [40] Y. Mao, F. R. Kschischang, B. Li, and S. Pasupathy, "A factor graph approach to link loss monitoring in wireless sensor networks," *IEEE J. Select. Areas Commun.*, vol. 23, no. 4, pp. 820–829, Apr. 2005.
- [41] B. J. Frey, N. Mohammad, Q. D. Morris, W. Zhang, M. D. Robinson, S. Mnaimneh, R. Chang, Q. Pan, E. Sat, J. Rossant, B. G. Bruneau, J. E. Aubin, B. J. Blencowe, and T. R. Hughes, "Genome-wide analysis of mouse transcripts using exon microarrays and factor graphs," *Nature Genetics*, vol. 37, no. 9, Sep. 2005.
- [42] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, pp. 972–976, Feb. 16, 2007.
- [43] G. D. Forney, Jr., "Codes on graphs: Normal realizations," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 520–548, Feb. 2001.
- [44] N. Wiberg, H.-A. Loeliger, and R. Kötter, "Codes and iterative decoding on general graphs," *Eur. Trans. Telecommun.*, vol. 6, pp. 513–525, Sep./Oct. 1995.
- [45] Y. Mao and F. R. Kschischang, "On factor graphs and the Fourier transform," *IEEE Trans. Inform. Theory*, vol. 51, no. 5, pp. 1635–1649, May 2005.
- [46] S. Roweis and Z. Ghahramani, "A unifying review of linear Gaussian models," *Neural Computation*, vol. 11, no. 2, pp. 305–345, Feb. 1999.
- [47] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 1996.
- [48] X. Wang and H. V. Poor, "Iterative (turbo) soft interference cancellation and decoding for coded CDMA," *IEEE Trans. Commun.*, vol. 47, pp. 1046–1061, Jul. 1999.
- [49] J. Hu, H.-A. Loeliger, J. Dauwels, and F. Kschischang, "A general computation rule for lossy summaries/messages with examples from equalization," in *Proc. 44th Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Sep. 27–29, 2006.
- [50] H.-A. Loeliger, "Some remarks on factor graphs," in *Proc. 3rd Int. Symp. Turbo Codes Related Topics*, Brest, France, Sep. 1–5, 2003, pp. 111–115.
- [51] J. Dauwels, S. Kori, and H.-A. Loeliger, "Steepest descent on factor graphs," in *Proc. IEEE Information Theory Workshop*, Rotorua, New Zealand, Aug. 28–Sep. 1, 2005, pp. 42–46.
- [52] P. M. Djuric, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Miguez, "Particle filtering," *IEEE Signal Proc. Mag.*, vol. 20, pp. 19–38, Sep. 2003.
- [53] A. Doucet, J. F. G. de Freitas, and N. J. Gordon, Eds., *Sequential Monte Carlo Methods in Practice*. New York: Springer-Verlag, 2001.
- [54] M. Briers, A. Doucet, and S. S. Singh, "Sequential auxiliary particle belief propagation," in *Proc. 7th Int. Conf. Information Fusion*, 2005.
- [55] F. Hamze and N. de Freitas, "Hot coupling: A particle approach to inference and normalization on pairwise undirected graphs," in *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Schölkopf,

- and J. Platt, Eds. Cambridge: MIT Press, 2006, pp. 491–498.
- [56] J. Dauwels, S. Kori, and H.-A. Loeliger, “Particle methods as message passing,” in *Proc. 2006 IEEE Int. Symp. Information Theory*, Seattle, WA, Jul. 9–14, 2006, pp. 2052–2056.
- [57] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” *J. Roy. Statistical Soc., ser. B*, vol. 39, pp. 1–38, 1977.
- [58] P. Stoica and Y. Selén, “Cyclic minimizers, majorization techniques, and the expectation-maximization algorithm: A refresher,” *IEEE Signal Proc. Mag.*, pp. 112–114, Jan. 2004.
- [59] S. Lauritzen, “The EM algorithm for graphical association models with missing data,” *Computational Statistics and Data Analysis*, vol. 19, pp. 191–201, 1995.
- [60] J. Dauwels, S. Kori, and H.-A. Loeliger, “Expectation maximization as message passing,” in *Proc. 2005 IEEE Int. Symp. Information Theory*, Adelaide, Australia, Sep. 4–9, 2005, pp. 583–586.
- [61] J. Dauwels, A. W. Eckford, S. Kori, and H.-A. Loeliger, *Expectation maximization in factor graphs*, in preparation.
- [62] P. O. Vontobel, “Kalman filters, factor graphs, and electrical networks,” ISI-ITET, ETH Zurich, Internal Rep. INT/200202, Apr. 2002.
- [63] P. O. Vontobel and H.-A. Loeliger, “On factor graphs and electrical networks,” in *Mathematical Systems Theory in Biology, Communication, Computation, and Finance*, J. Rosenthal and D. S. Gilliam, Eds. New York: Springer Verlag, 2003, pp. 469–492.

ABOUT THE AUTHORS

Hans-Andrea Loeliger (Fellow, IEEE) received the diploma in electrical engineering in 1985 and Ph.D. degree in 1992, both from ETH Zurich, Switzerland.

From 1992 to 1995, he was with Linköping University, Sweden. From 1995 to 2000, he was with Endora Tech AG, Basel, Switzerland, of which he is a Cofounder. Since 2000, he has been a Professor at ETH Zurich. His research interests lie in the broad areas of signal processing, information theory, communications, and electronics.



Li Ping (Senior Member, IEEE) received the Ph.D. degree from Glasgow University, U.K., in 1990.

He lectured at the Department of Electronic Engineering, Melbourne University, from 1990 to 1992, and worked as a member of research staff at Telecom Australia Research Laboratories from 1993 to 1995. He has been with the Department of Electronic Engineering, City University of Hong Kong, since January 1996, where he is now a Chair Professor. His research interests are mixed analog/digital circuits, communications systems, and coding theory.



Justin Dauwels (Member, IEEE) received the diploma (M.Sc. degree) in engineering physics from the University of Gent, Belgium, in 2000, and the Ph.D. degree in electrical engineering from ETH Zurich, Switzerland, in 2005.

He is currently a Research Scientist at the Amari Research Unit of the RIKEN Brain Science Institute, Japan. His broader research interests include signal processing, machine learning, digital communications, applied information theory, and computational neuroscience.

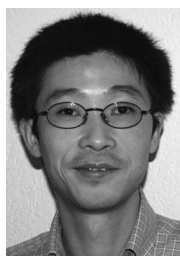


Dr. Ping was awarded a British Telecom-Royal Society Fellowship in 1986, the IEE J. J. Thomson premium in 1993, and a Croucher Senior Research Fellowship in 2005.

Dr. Dauwels is a recipient of the ICASSP2006 Student Paper Award, a JSPS Fellowship in 2006, and the Henri-Benedictus Fellowship of the BAEF and the King Baudouin Foundation in 2007.

Junli Hu (Member, IEEE) received the B.Sc. degree from Shanghai Jiantong University, in 1992, and the diploma from ETH Zurich, Switzerland, in 1998, both in electrical engineering. Since 2004, he has been working toward the Ph.D. degree at the Signal and Information Processing Laboratory, ETH Zurich.

Since 2001 he has been with Wavecom Elektronik AG, Zurich. His current research interest is equalization and estimation using graphical models.



Frank R. Kschischang (Fellow, IEEE) received the B.A.Sc. (Honors) degree from the University of British Columbia, Vancouver, BC, Canada, in 1985, and the M.A.Sc. and Ph.D. degrees from the University of Toronto, Toronto, ON, Canada, in 1988 and 1991, respectively, all in electrical engineering.

During 1997 to 1998, he spent a sabbatical year as a Visiting Scientist at the Massachusetts Institute of Technology, Cambridge, and in 2005 he was a Guest Professor at ETH Zurich, Switzerland. Currently, he is a Professor and Canada Research Chair in the Department of Electrical and Computer Engineering, University of Toronto, where he has been a faculty member since 1991. His research interests include coding techniques, primarily on soft-decision decoding algorithms, trellis structure of codes, codes defined on graphs, and iterative decoders and in the application of coding techniques to wireline, wireless, and optical communication systems.



Sascha Kori (Member, IEEE) was born in Vienna, Austria, in 1974. He received the M.Sc. degree in electrical engineering from the Graz University of Technology, Graz, Austria, in 2000, and the Ph.D. degree in electrical engineering from ETH Zurich, Switzerland, in 2005.

During his studies he developed several new message-passing techniques on factor graphs, for example the use of particle and gradient techniques on factor graphs and the derivation of EM-type algorithms by local message updates on factor graphs. In 2005, he joined the applied research department of Phonak AG, Stäfa, Switzerland, where he is engaged in research on signal processing and machine learning methods for innovative hearing solutions.

