AIAA Guidance, Navigation, and Control Conference
2 - 5 August 2010, Toronto, Ontario Canada

AIAA 2010-7747

# Gaussian Mixture Approximation by Another Gaussian Mixture for "Blob" Filter Re-Sampling

Mark L. Psiaki*, Jonathan R. Schoenberg#
*Cornell University, Ithaca, N.Y. 14853-7501*

and Isaac T. Miller
*Coherent Navigation, Inc., San Mateo, CA 94404*

**A new method has been developed to approximate one Gaussian mixture by another in a process that generalizes the idea of importance re-sampling in a particle filter. This algorithm is being developed as part of an effort to generalize the concept of a particle filter. In a traditional particle filter, the underlying probability density function is described by particles: Dirac delta functions with infinitesimal covariances. This paper develops an important component of a "blob" filter, which uses a Gaussian mixture of "fattened," finite-covariance blobs instead of infinitesimal particles. The goal of a blob filter is to save computational effort for a given level of probability density precision by using many fewer blobs than particles. Most of the techniques necessary for this type of filter have already been developed. The one missing component is developed in this paper: a re-sampling algorithm that bounds the covariance of each element while accurately re-producing the original probability distribution. The covariance bounds are needed in order to keep the blobs from becoming too "fat"; otherwise, Extended Kalman Filter (EKF) or Unscented Kalman Filter dynamic propagation and measurement update calculations would cause excessive truncation error for each blob. The re-sampling algorithm is described in detail, and its performance is studied using several simulated test cases. Also discussed is the usefulness of a Gaussian mixture and EKF-like techniques for nonlinear dynamic propagation and nonlinear measurement update of probability distributions.**

## I. Introduction

Difficulties can arise when solving certain nonlinear dynamic estimation problems. The default solution algorithm for such problems is the EKF, but the EKF has a known potential to diverge or to yield sub-optimal accuracy [1,2,3]. Various algorithms have been developed with the goal of improved convergence robustness or accuracy in the presence of strong nonlinearities, among them the Unscented or Sigma-Points Kalman Filter (UKF) [1,4], the Particle Filter (PF) [2], and the Backward-Smoothing Extended Kalman Filter [3].

The PF is attractive for its simplicity and its theoretical guarantee of convergence to the optimal result in the limit of very many particles. The required number of particles to achieve a reasonable result, however, can become overwhelming for state space dimensions as small as 3 or 4, as in Ref. 5.

A sensible generalization of the PF is to use Gaussian mixtures to represent probability density functions. In effect, a PF works with representations of probability density functions that are sums of Dirac delta functions. A Gaussian mixture generalizes this concept by using elements that have finite widths instead of infinitesimal widths. A sum of finite-width elements has the potential to approximate a probability density function with many fewer elements than would be needed by a PF for the same degree of accuracy, as measured based on differences of multiple moments or based on the functional norm "distance" from the true probability density. Thus, a Gaussian mixture filter has the potential to solve the curse of dimensionality that causes a PF to become impractical for state space dimensions above 2 or 3.

---

* Professor, Sibley School of Mechanical and Aerospace Engineering. Associate Fellow, AIAA.
# Graduate student, Sibley School of Mechanical and Aerospace Engineering.

Gaussian mixture filters have been studied extensively in the past, and Ref. 6 is one of the earliest known papers on this subject. The proposed "blob" filter is a modified version of the Gaussian mixture filter of Ref. 7. That filter implements a separate standard UKF dynamic propagation and measurement update for each element of its Gaussian mixture. Its approximate implementation of the full non-linear Bayesian measurement update dictates that it recalculate the weights of its mixture elements. This re-calculation increases the weights of elements whose predicted measurements best match the actual measurement at the given sample, as in the static multiple-model filter described in Ref. 8. The final filter action for a given sample is to re-approximate the mixture by drawing samples from it and then fitting a new Gaussian mixture to the samples. This action can avert degeneracy of the mixture into one element or a very few elements that have appreciable weight, and it can reduce the number of elements in cases where the multiplication of Gaussian mixtures would cause exponential growth of this number.

The Gaussian mixture filter of Ref. 7 has strengths and weaknesses. Its main strength lies in the potential accuracy of the UKF dynamic propagation and measurement update of the mixture. If each element of the Gaussian mixture has a covariance that is sufficiently small, as measured in terms of the maximum eigenvalue or some other sensible metric, then the UKF calculations will be very accurate. With sufficiently small covariances, EKF calculations also would be sufficiently accurate to yield a good estimate of the posterior probability density function. This is true because a narrow distribution implies good accuracy of the Taylor series approximations inherent in the UKF or EKF calculations. That is, the Taylor series approximations are accurate over the likely ranges of state variations of each mixture element.

The weakness of the filter of Ref. 7 lies in its re-approximation of the Gaussian mixture distribution after the measurement update. This re-approximation samples the original mixture and fits a new Gaussian mixture to the samples via Expectation Maximization (EM). This procedure achieves the worthy goal of eliminating mixture elements with low weights. Unfortunately, it does not limit the maximum covariance of any element of the re-approximation. This limitation is needed so that the next recursion of the filtering algorithm will yield good accuracy when using the approximations that are inherent in its element-by-element EKF or UKF calculations. It is not obvious how to add such a limitation to the EM-based re-sampling procedure without making it unduly complicated.

The other weakness of the mixture re-approximation is its failure to fit the old mixture as closely as possible in some functional norm sense, as in the Integral Square Difference (ISD) metric of Ref. 9. Instead, the filter of Ref. 7 uses an ad hoc transition first to particle samples of the old mixture and finally back to Gaussian mixture elements that fit these particles.

Section IV of Ref. 10 constitutes a prototype application of similar concepts to those of Ref. 7, albeit with EKFs used in place of UKFs for propagating and updating the Gaussian mixture elements and without any need for re-sampling. The goal of this application was to achieve filter convergence from large initial uncertainty in a difficult spacecraft attitude determination problem. The algorithm converged reliably from large initial errors that would have caused an EKF to fail. This reliable convergence implies that the algorithm of Ref. 7 could provide a powerful solution to difficult problems in nonlinear filtering if it were modified to use this paper's re-sampling procedure.

The present paper's contribution is an improved Gaussian mixture re-approximation algorithm that could be applied to a filter like that of Ref. 7. It has four important properties: First, it chooses elements of the new mixture so that their covariances lie below a linear matrix inequality (LMI) upper bound. This constraint is included to ensure that element-by-element EKF or UKF dynamic propagation and measurement update calculations will yield a sufficiently accurate approximation of the *a posteriori* probability density function on the next Bayesian filter step. Second, it directly chooses new mixture elements and their weights in a way that seeks to minimize the ISD between the new Gaussian mixture distribution and the original distribution. Third, it tends not to waste new elements in attempts to approximate the contributions of original elements that have low weights. Last, it tends to hold down the number of needed new elements through a combination of strategies. These strategies include a) maximization of new element covariances subject to the LMI constraint, b) selection of new element means and weights in a way that tends to limit the number of new elements needed for a given improvement to the ISD, c) termination of the generation of new elements when a conservative upper bound has been met for the ISD between the new and old mixtures, and d) combination of elements if their Gaussian sum can be approximated well by a single new element.

There exist other Gaussian mixture re-sampling schemes [9,11,12]. This paper's new algorithm differs from the existing algorithms in several important respects. The existing algorithms' primary goal is to approximate an original mixture by a new one that has fewer elements. The present algorithm retains this as a secondary goal, but its main goal is to develop a new approximate mixture whose elements all have covariances that satisfy an LMI

upper bound, which is an important property when using Gaussian mixtures to generalize nonlinear particle filtering. The new algorithm uses the ISD fit metric of Ref. 9, but in a new way: It formulates and solves quadratic programs based on the ISD metric in order to choose optimal relative weights for subsets of the new mixture's elements.

This paper's new Gaussian mixture re-approximation method will be useful for generalizing the nonlinear particle filter to create a blob filter along the lines of Ref. 7, as stated above. The key generalization is to replace particles of infinitesimal width by blobs of finite width. Therefore, another important aspect of the new re-approximation scheme is that it approaches the re-sampling scheme of a standard particle filter [2] in the limit of a very small upper bound on the covariances of the new elements. This asymptotic similarity makes the blob filter a natural generalization of the particle filter.

Asymptotic similarity to PF re-sampling is achieved by choosing the mean values of the new mixture elements through the use of a modified sampling method. This modified procedure employs a perturbation of the original distribution, and it samples the perturbed distribution using Metropolis-Hastings techniques [13]. The perturbed distribution reduces the original distribution near existing new elements, thereby reducing the probability that additional new elements will be located near existing elements. A particle filter, on the other hand, tends to sample many particles very near each other in regions of the filter's state space that have high probability densities, as characterized by the high numbers of existing particles in those regions. The new re-sampling method achieves equivalent results through an explicit increase of the weights that it assigns to new mixture elements that lie in regions of high original probability density.

Figure 1 depicts the new re-sampling algorithm in block-diagram form. It starts in the upper left-hand corner of the diagram with original Gaussian mixture distribution $p_a(\boldsymbol{x})$. Its 1st block decomposes $p_a(\boldsymbol{x})$ into sub-mixtures, and its remaining blocks fit corresponding sub-mixtures of the new re-sampled distribution to these original sub-mixtures. Initially, each sub-mixture of the new distribution is a poor fit to the corresponding sub-mixture of the original distribution because it lacks elements, and the algorithm initializes fit parameters accordingly in its 2nd block. Each new sub-mixture uses a common covariance matrix for each of its elements, and the set of sub-mixture covariance matrices is computed in the 3rd algorithm block. The 3rd block also pre-computes parameters that are used in the 7th block in order to set up an optimization problem for the relative weights within each new sub-mixture.

The algorithm's main loop is depicted by the 4th-10th blocks of Fig. 1. It adds one new element to the new Gaussian mixture per pass through these blocks. The 4th block computes modified sub-mixture weights that assign higher values to those original sub-mixtures that have a) high original weights, b) poor fits to their corresponding new sub-mixtures, or c) both. The 5th block picks which new sub-mixture to augment with a new element. The selection procedure uses importance sampling and the modified weights from the 4th block. A new sub-mixture is likely to gain a new element if its corresponding original sub-mixture has a sufficiently high original weight and if its fit to that original sub-mixture is sufficiently poor. The 6th block draws the mean value of the new mixand from a modified form of the probability density function of the corresponding original sub-mixture. This modified sub-mixture has reduced probability density near pre-existing new sub-mixture elements, thereby reducing the likelihood of close spacing between new elements. The 7th block optimizes the relative weights of the new sub-mixture to account for its new mixand. Decision block 8 rejects the new mixand if the resulting ISD fit error of the augmented sub-mixture does not decrease, in which case another new-element mean value is sampled in a return to the 6th block. Otherwise, two termination criteria are tested in the 9th and 10th blocks. Termination occurs if the overall fit error is sufficiently small or if a given upper limit on the number of new mixands has been reached. The 11th block finishes by computing the new mixands' final weights.

This paper develops and analyzes its new Gaussian mixture re-approximation algorithm in 8 main sections. Section II defines Gaussian mixtures using square-root information matrix notation, and it defines sub-mixtures as being subsets of the elements of a given mixture. Section III develops the ISD error metric between two Gaussian mixtures, derives an analytic formula for the ISD, and determines an upper bound for the relative norm error between two Gaussian mixtures. This relative upper bound is used to implement the algorithm termination test in the 9th block of Fig. 1. Section IV presents a quadratic program (QP) that chooses the weights of a new Gaussian mixture in order to minimize the ISD between it and an original Gaussian mixture. This QP algorithm is used in the 7th block of Fig. 1. Section V defines an LMI that bounds the covariances of the elements of the new Gaussian mixture. It develops an algorithm for choosing the covariance of a new element in a way that respects this limit while deviating as little as possible from the covariance of a corresponding element of the original mixture. This LMI solution algorithm is used by the 3rd block of Fig. 1. Section VI introduces a technique for decomposing the original mixture into sub-mixtures, as per Block 1 of Fig. 1. These sub-mixture groupings can help to reduce the

number of elements of the new mixture. Section VII presents the algorithm for selecting means and covariances of new mixture elements, as needed to implement the 6th block in Fig. 1. Section VIII combines the developments of Sections II-VII in order to define the new Gaussian mixture re-sampling algorithm. Section IX presents example test results that illustrate the performance and usefulness of the new algorithm. Section X summarizes this paper's developments and presents its conclusions.

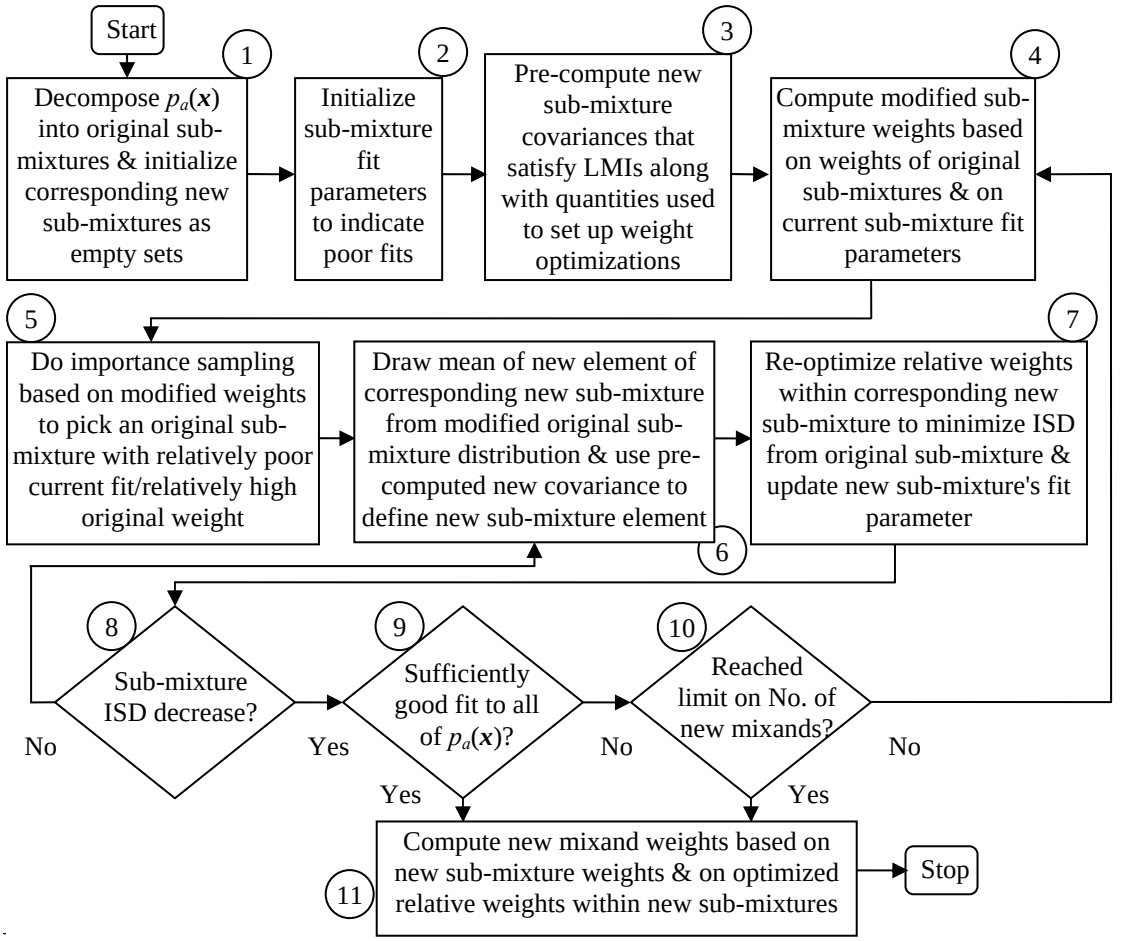


**Fig. 1. Flow chart of Gaussian mixture re-sampling algorithm.**

## II. Gaussian Mixture Probability Density Functions

### A. Original and New Gaussian Mixture Probability Density Functions

A Gaussian mixture is a weighted sum of Gaussian distributions. The $i^{\text{th}}$ element of the mixture, also called the $i^{\text{th}}$ mixand or the $i^{\text{th}}$ component, can by characterized by its square-root information matrix $R_i$ and its mean $\boldsymbol{\mu}_i$. The element probability distribution is:

$$\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_i,R_i)=\frac{|\det(R_i)|}{(2\pi)^{n/2}}e^{-0.5[R_i(\boldsymbol{x}-\boldsymbol{\mu}_i)]^{\mathrm{T}}[R_i(\boldsymbol{x}-\boldsymbol{\mu}_i)]} \;=\; \mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_i,R_i^{-1}R_i^{-\mathrm{T}}) \tag{1}$$

where $\boldsymbol{x}$ and $\boldsymbol{\mu}_i$ are $n$-dimensional vectors and $R_i$ is an $n$-by-$n$ matrix. The covariance matrix of this distribution is $P_i = R_i^{-1}R_i^{-\mathrm{T}}$, where the notation $()^{-\mathrm{T}}$ indicates the inverse of the transpose of the matrix in question. The notation $\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu},P)$ indicates the usual normal distribution in the vector $\boldsymbol{x}$ that has mean $\boldsymbol{\mu}$ and covariance matrix $P$. The new notation $\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu},R)$ indicates the same distribution in $\boldsymbol{x}$, except that its covariance is characterized by the square-root information matrix $R$ in place of the covariance matrix $P$. This non-standard parameterization of the normal

distribution will be used throughout the remainder of this paper. It has been chosen because it allows a simple LMI solution in Section V and because it is consistent with the planned square-root information filter (SRIF) implementation of the proposed "blob" filter. An SRIF implementation is desirable because it has good numerical stability.

Each element of a Gaussian mixture also has a weight, $w_i$. Each weight must be non-negative. The sum of all of the weights equals 1. If there are $N$ elements in the mixture, then

$$1 = \sum_{i=1}^{N} w_i \quad \text{and} \quad w_i \geq 0 \quad \text{for } i = 1, ..., N \tag{2}$$

Given the Gaussian component definition in Eq. (1) and weights that obey the constraints in Eq. (2), the corresponding Gaussian mixture is

$$p_{gm}(\boldsymbol{x}; w_1, \boldsymbol{\mu}_1, R_1, ..., w_N, \boldsymbol{\mu}_N, R_N) = \sum_{i=1}^{N} w_i \mathcal{N}_{sr}(\boldsymbol{x}; \boldsymbol{\mu}_i, R_i) \tag{3}$$

It is straightforward to show that this probability density function preserves the unit normalization constraint and that its mean and covariance are, respectively,

$$\boldsymbol{\mu}_{gm} = \sum_{i=1}^{N} w_i \boldsymbol{\mu}_i \quad \text{and} \quad P_{gm} = \sum_{i=1}^{N} w_i [R_i^{-1} R_i^{-\mathrm{T}} + (\boldsymbol{\mu}_i - \boldsymbol{\mu}_{gm})(\boldsymbol{\mu}_i - \boldsymbol{\mu}_{gm})^{\mathrm{T}}] \tag{4}$$

It is necessary to distinguish between two Gaussian mixture distributions in this paper. Suppose that one distribution, distribution "$a$", is characterized by the weights, mean values, and square root-information matrices $w_{ai}$, $\boldsymbol{\mu}_{ai}$, $R_{ai}$ for $i = 1, ..., N_a$. Similarly, suppose that another related distribution, distribution "$b$", is characterized by $w_{bj}$, $\boldsymbol{\mu}_{bj}$, $R_{bj}$ for $j = 1, ..., N_b$. The following short-hand notation is used to indicate these two distributions

$$p_a(\boldsymbol{x}) = p_{gm}(\boldsymbol{x}; w_{a1}, \boldsymbol{\mu}_{a1}, R_{a1}, ..., w_{aN_a}, \boldsymbol{\mu}_{aN_a}, R_{aN_a}) = \sum_{i=1}^{N_a} w_{ai} \mathcal{N}_{sr}(\boldsymbol{x}; \boldsymbol{\mu}_{ai}, R_{ai}) \tag{5a}$$

$$p_b(\boldsymbol{x}) = p_{gm}(\boldsymbol{x}; w_{b1}, \boldsymbol{\mu}_{b1}, R_{b1}, ..., w_{bN_b}, \boldsymbol{\mu}_{bN_b}, R_{aN_b}) = \sum_{j=1}^{N_b} w_{bj} \mathcal{N}_{sr}(\boldsymbol{x}; \boldsymbol{\mu}_{bj}, R_{bj}) \tag{5b}$$

The goal of this paper is to develop a method that picks the parameters of distribution "$b$", $N_b$ and $w_{bj}$, $\boldsymbol{\mu}_{bj}$, and $R_{bj}$ for $j = 1, ..., N_b$. It seeks to pick these parameters in a way that will cause $p_b(\boldsymbol{x})$ to be a good approximation of $p_a(\boldsymbol{x})$ while respecting an LMI lower bound on every $R_{bj}^{\mathrm{T}} R_{bj}$ for $j = 1, ..., N_b$. The algorithm's LMI lower bound on $R_{bj}^{\mathrm{T}} R_{bj}$ is an alternate means of enforcing an LMI upper bound on the covariance $P_{bj} = R_{bj}^{-1} R_{bj}^{-\mathrm{T}}$. The algorithm also seeks to keep the number of new elements $N_b$ from being too large. Of course, there normally is a trade-off between the size of $N_b$ and the accuracy with which $p_b(\boldsymbol{x})$ approximates $p_a(\boldsymbol{x})$.

**B. Decomposition into Sub-Mixtures**

It can be useful to break Gaussian mixtures $p_a(\boldsymbol{x})$ and $p_b(\boldsymbol{x})$ into weighted sums of sub-mixtures. This decomposition allows the original function approximation problem to be broken into a set of smaller approximation problems. It can be used to reduce the computational burden of this paper's algorithms. Note that the term "sub-mixture" is non-standard. It denotes a Gaussian mixture distribution that is formed using a re-weighted subset of the elements of an original Gaussian mixture.

Let distribution $p_a(\boldsymbol{x})$ and distribution $p_b(\boldsymbol{x})$ be broken into the following disjoint sets of sub-mixtures

$$p_{sam}(\boldsymbol{x}) = \frac{\sum_{i=i_{lom}}^{i_{him}} w_{ai} \mathcal{N}_{sr}(\boldsymbol{x}; \boldsymbol{\mu}_{ai}, R_{ai})}{w_{sam}} \qquad \text{for } m = 1, ..., M \tag{6a}$$

$$p_{sbm}(\boldsymbol{x}) = \frac{\sum_{j=j_{lom}}^{j_{him}} w_{bj} \mathcal{N}_{sr}(\boldsymbol{x}; \boldsymbol{\mu}_{bj}, R_{bj})}{w_{sbm}} \qquad \text{for } m = 1, ..., M \tag{6b}$$

where the sub-mixture cumulative weights are defined to be

$$w_{sam} = \sum_{i=i_{lom}}^{i_{him}} w_{ai} \qquad \text{for } m = 1, ..., M \tag{7a}$$

$$w_{sbm} = \sum_{j=j_{lom}}^{j_{him}} w_{bj} \qquad \text{for } m = 1, ..., M \tag{7b}$$

The start and stop indices $i_{lom}$ and $i_{him}$ define the index range of components of the original Gaussian mixture $p_a(\boldsymbol{x})$ that form its $m^{\text{th}}$ sub-mixture $p_{sam}(\boldsymbol{x})$. The indices $j_{lom}$ and $j_{him}$ work similarly to define the $m^{\text{th}}$ sub-mixture $p_{sbm}(\boldsymbol{x})$ of $p_b(\boldsymbol{x})$. These indices obey the constraints:

$$i_{lo1} = 1, \quad i_{lo(m+1)} = i_{him}+1 \text{ for } m = 1,...,(M\text{-}1), \quad i_{hiM} = N_a, \quad \text{and } i_{lom} \le i_{him} \text{ for } m = 1,...,\text{M} \tag{8a}$$

$$j_{lo1} = 1, \quad j_{lo(m+1)} = j_{him}+1 \text{ for } m = 1,...,(M\text{-}1), \quad j_{hiM} = N_b, \quad \text{and } j_{lom}\text{-}1 \le j_{him} \text{ for } m = 1,...,\text{M} \tag{8b}$$

These index constraints ensure that each original mixand appears in one and only one sub-mixture for probability density functions $p_a(\boldsymbol{x})$ and $p_b(\boldsymbol{x})$. The $m^{\text{th}}$ sub-mixture of $p_a(\boldsymbol{x})$ has $N_{am} = i_{him} - i_{lom}+1$ Gaussian components, and the $m^{\text{th}}$ sub-mixture of $p_b(\boldsymbol{x})$ has $N_{bm} = j_{him} - j_{lom}+1$ components. The constraint $j_{lom}\text{-}1 \le j_{him}$ allows for the possibility that $N_{bm} = 0$ if $j_{lom}\text{-}1 = j_{him}$. In this situation, $p_{sbm}(\boldsymbol{x})$ and $w_{sbm}$ are undefined. This situation may arise if the corresponding original sub-mixture weight $w_{sam}$ is very low, in which case the new Gaussian mixture $p_b(\boldsymbol{x})$ may not devote any elements to fitting the effects of $p_{sam}(\boldsymbol{x})$.

It is helpful to define relative weights within a given sub-mixture. They are:

$$\breve{w}_{ai} = w_{ai} / w_{sam} \qquad \text{for } i = i_{lom}, ..., i_{him} \text{ and for } m = 1, ..., M \tag{9a}$$

$$\breve{w}_{bj} = w_{bj} / w_{sbm} \qquad \text{for } j = j_{lom}, ..., j_{him} \text{ and for } m = 1, ..., M \tag{9b}$$

Equations (9a) and (9b) guarantee normalization of the weights within each sub-mixture, and they allow the sub-mixtures in Eqs. (6a) and (6b) to be expressed as true Gaussian mixtures in their own right:

$$p_{sam}(\boldsymbol{x}) = \sum_{i=i_{lom}}^{i_{him}} \breve{w}_{ai} \mathcal{N}_{sr}(\boldsymbol{x}; \boldsymbol{\mu}_{ai}, R_{ai}) \qquad \text{for } m = 1, ..., M \tag{10a}$$

$$p_{sbm}(\boldsymbol{x}) = \sum_{j=j_{lom}}^{j_{him}} \breve{w}_{bj} \mathcal{N}_{sr}(\boldsymbol{x}; \boldsymbol{\mu}_{bj}, R_{bj}) \quad \text{for } m = 1, ..., M \tag{10b}$$

It is possible to express the two original Gaussian mixtures as weighted sums of these sub-mixtures:

$$p_a(\boldsymbol{x}) = \sum_{m=1}^{M} w_{sam} p_{sam}(\boldsymbol{x}) \tag{11a}$$

$$p_b(\boldsymbol{x}) = \sum_{m=1}^{M} w_{sbm} p_{sbm}(\boldsymbol{x}) \tag{11b}$$

Equations (7a) and (7b) and the normalization and non-negativeness of the original mixture weights imply that the sub-mixture weights are also normalized and non-negative:

$$1 = \sum_{m=1}^{M} w_{sam} \qquad \text{and} \quad w_{sam} \ge 0 \quad \text{for } m = 1, ..., M \tag{12a}$$

$$1 = \sum_{m=1}^{M} w_{sbm} \qquad \text{and} \quad w_{sbm} \ge 0 \quad \text{for } m = 1, ..., M \tag{12b}$$

## III. The Integral Square Difference between Two Gaussian Mixtures as a Measure of Approximation Accuracy

### A. ISD Definition

The Integral Square Difference is a good measure of the accuracy with which $p_b(\boldsymbol{x})$ approximates $p_a(\boldsymbol{x})$. The ISD is defined to be the integral of the square of the difference between these two probability density functions [9]:

$$J_{ISD} = \int_{-\infty}^{\infty} [p_a(\boldsymbol{x}) - p_b(\boldsymbol{x})]^2 \, d\boldsymbol{x} \tag{13}$$

This quantity is non-negative, and its square root is the functional 2-norm of the difference between the probability distributions

$$\| p_a(\boldsymbol{x}) - p_b(\boldsymbol{x}) \|_2 = \sqrt{J_{ISD}} = \left\{ \int_{-\infty}^{\infty} [p_a(\boldsymbol{x}) - p_b(\boldsymbol{x})]^2 \, d\boldsymbol{x} \right\}^{0.5} \tag{14}$$

Therefore, the ISD is a good measure of the similarity between these two functions. A very small value of the ISD cost $J_{ISD}$ indicates that $p_b(\boldsymbol{x})$ is a very good approximation of $p_a(\boldsymbol{x})$. Distribution $p_b(\boldsymbol{x})$ perfectly matches $p_a(\boldsymbol{x})$ if and only if $J_{ISD} = 0$.

Some re-approximation algorithms seek to choose $p_b(\boldsymbol{x})$ in a way that explicitly constrains the mean and covariance of the new distribution to equal that of $p_a(\boldsymbol{x})$, e.g., see Ref. 11. In order to avoid an additional source of algorithmic complexity, the current approach enforces no such constraint. If $J_{ISD}$ in Eq. (13) is sufficiently small, however, then the mean and covariance of $p_b(\boldsymbol{x})$ will be very close to the corresponding $p_a(\boldsymbol{x})$ quantities, as demonstrated by example in Section IX. In addition, a small $J_{ISD}$ could cause a number of higher moments of $p_b(\boldsymbol{x})$ to be closer to the corresponding moments of $p_a(\boldsymbol{x})$ than they would be if a different re-approximation technique were used to construct $p_b(\boldsymbol{x})$ without consideration of its $J_{ISD}$.

**B. Analytic Formulas for the ISD**

Reference 9 presents analytic formulas for evaluating the integral in Eq. (13). The formulas used here are modified versions of those found in Ref. 9. The modifications account for the use of square-root information matrices in place of mixand covariance matrices. Suppose that one defines weight vectors for the two probability density functions: $\boldsymbol{w}_a = [w_{a1};w_{a2};w_{a3};...;w_{aNa}]$ and $\boldsymbol{w}_b = [w_{b1};w_{b2};w_{b3};...;w_{bNb}]$. Then the integral in Eq. (13) can be written as a quadratic form in these two vectors:

$$J_{ISD} = \boldsymbol{w}_a^{\mathrm{T}} H_{aa} \boldsymbol{w}_a - 2\boldsymbol{w}_a^{\mathrm{T}} H_{ab} \boldsymbol{w}_b + \boldsymbol{w}_b^{\mathrm{T}} H_{bb} \boldsymbol{w}_b \tag{15}$$

where $H_{aa}$, $H_{ab}$, and $H_{bb}$ are matrices with the respective dimensions $N_a$-by-$N_a$, $N_a$-by-$N_b$, and $N_b$-by-$N_b$. The matrices $H_{aa}$ and $H_{bb}$ are symmetric and at least positive semi-definite. The elements of these matrices can be evaluated using the formulas:

$$[H_{aa}]_{ik} = \int_{-\infty}^{\infty} \mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_{ai},R_{ai})\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_{ak},R_{ak})d\boldsymbol{x} \qquad \text{for } i = 1,...,N_a \text{ and } k = 1,...,N_a \tag{16a}$$

$$[H_{ab}]_{ij} = \int_{-\infty}^{\infty} \mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_{ai},R_{ai})\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_{bj},R_{bj})d\boldsymbol{x} \qquad \text{for } i = 1,...,N_a \text{ and } j = 1,...,N_b \tag{16b}$$

$$[H_{bb}]_{jl} = \int_{-\infty}^{\infty} \mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_{bj},R_{bj})\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_{bl},R_{bl})d\boldsymbol{x} \qquad \text{for } j = 1,...,N_b \text{ and } l = 1,...,N_b \tag{16c}$$

where the notation $[]_{ik}$ indicates the row-$i$/column-$k$ element of the matrix in question.

The integrals in Eqs. (16a)-(16c) can be evaluated analytically by using the normalization property of a Gaussian distribution and the fact that the product of two Gaussian distributions is itself a Gaussian distribution, although not properly normalized [9]. These integrals take the general form:

$$\int_{-\infty}^{\infty} \mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_c,R_c)\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_d,R_d)d\boldsymbol{x} = \frac{|\det(R_c)||\det(R_d)|}{(2\pi)^{n/2}|\det(\overline{R}_{cd})|} e^{-0.5[\tilde{R}_{cd}(\boldsymbol{\mu}_c-\boldsymbol{\mu}_d)]^{\mathrm{T}}[\tilde{R}_{cd}(\boldsymbol{\mu}_c-\boldsymbol{\mu}_d)]} \tag{17}$$

where the $n$-by-$n$ matrices $\overline{R}_{cd}$ and $\tilde{R}_{cd}$ are computed based upon the following orthonormal/upper-triangular (QR) factorization [14]:

$$Q\begin{bmatrix}\overline{R}_{cd}\\0\end{bmatrix} = [Q_1,Q_2]\begin{bmatrix}\overline{R}_{cd}\\0\end{bmatrix} = \begin{bmatrix}R_c\\R_d\end{bmatrix} \tag{18}$$

with $Q$ being a $2n$-by-$2n$ orthonormal matrix and $\overline{R}_{cd}$ an $n$-by-$n$ upper-triangular matrix. $Q_1$ equals the first $n$ columns of $Q$, and $Q_2$ equals the last $n$ columns. These matrices are used to compute

$$\tilde{R}_{cd} = Q_2^{\mathrm{T}}\begin{bmatrix}R_c\\0\end{bmatrix} = -Q_2^{\mathrm{T}}\begin{bmatrix}0\\R_d\end{bmatrix} \tag{19}$$

Equations (17)-(19) have been derived using a lengthy, non-intuitive sequence of matrix/vector manipulations that have been omitted for the sake of brevity. In the special case where $R_c = R_d$, it suffices to use $\overline{R}_{cd} = \sqrt{2}R_c$ and $\tilde{R}_{cd} = (1/\sqrt{2})R_c$, and in this case the Eq. (17) integral becomes

$$\int_{-\infty}^{\infty} \mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_c,R_c)\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_d,R_c)d\boldsymbol{x} = \frac{|\det(R_c)|}{2^n\pi^{n/2}} e^{-0.25[R_c(\boldsymbol{\mu}_c-\boldsymbol{\mu}_d)]^{\mathrm{T}}[R_c(\boldsymbol{\mu}_c-\boldsymbol{\mu}_d)]} \tag{20}$$

**C. An ISD-Based Metric for Relative Fit Error**

The ISD formula in Eq. (15) can be used to develop a relative measure of the accuracy with which new Gaussian mixture $p_b(\boldsymbol{x})$ approximates original Gaussian mixture $p_a(\boldsymbol{x})$. A sensible metric is the norm of the difference between $p_b(\boldsymbol{x})$ and $p_a(\boldsymbol{x})$ divided by the norm of $p_a(\boldsymbol{x})$:

$$e_{relba} = \frac{\| p_a(\boldsymbol{x}) - p_b(\boldsymbol{x}) \|_2}{\| p_a(\boldsymbol{x}) \|_2} = \frac{\sqrt{\boldsymbol{w}_a^{\mathrm{T}} H_{aa} \boldsymbol{w}_a - 2\boldsymbol{w}_a^{\mathrm{T}} H_{ab} \boldsymbol{w}_b + \boldsymbol{w}_b^{\mathrm{T}} H_{bb} \boldsymbol{w}_b}}{\sqrt{\boldsymbol{w}_a^{\mathrm{T}} H_{aa} \boldsymbol{w}_a}} \tag{21}$$

If this quantity is small relative to 1, then the difference between $p_b(\boldsymbol{x})$ and $p_a(\boldsymbol{x})$ is small relative to the magnitude of $p_a(\boldsymbol{x})$.

The metric $e_{relba}$ provides a sensible termination criterion for the procedure that generates new mixand elements of $p_b(\boldsymbol{x})$. If $e_{relba}$ is sufficiently small compared to 1, then the algorithm should terminate with the given $p_b(\boldsymbol{x})$; otherwise, it should continue generating new elements. If $e_{relba}$ is very small compared to 1, then the mean and covariance of $p_b(\boldsymbol{x})$ should closely match those of $p_a(\boldsymbol{x})$, as demonstrated by example in Section IX.

**D. Sub-Mixture ISDs and Practical Computation of a Bound on the Relative Fit Error**

The re-sampling algorithm needs a practical means for measuring the goodness of the fit of $p_b(\boldsymbol{x})$ to $p_a(\boldsymbol{x})$. It uses this metric to implement the termination test in the 9th block of Fig. 1. Computation of the relative error in Eq. (21) will be prohibitively expensive if there are large numbers of mixture elements in the original distribution, $N_a$, or in the new distribution, $N_b$. Consider the leading-term scalings of the number of operations required to compute the $H_{aa}$, $H_{ab}$, and $H_{bb}$ matrices, as per Eqs. (16a)-(19). They are, respectively, $N_a^2 n^3$, $N_a N_b n^3$, and $N_b^2 n^3$. Recall that $n$ is the dimension of the $\boldsymbol{x}$ vector. The $n^3$ factor arises from the QR factorization in Eq. (18). The other factors arise from the dimensions of the dense $H_{aa}$, $H_{ab}$, and $H_{bb}$ matrices. In many situations envisioned in this paper, it may be possible to eliminate many of the QR factorizations in Eq. (18) due to the re-use of the same $R$ matrix in multiple mixands of a given distribution. If this were the case, then the respective leading computational cost terms would scale as $N_a^2 n^2$, $N_a N_b n^2$, and $N_b^2 n^2$. The $n^2$ factors arise from the matrix-vector multiplications that are needed to compute the exponent in Eq. (17). If there were $N_a = 2000$ mixands in probability density function $p_a(\boldsymbol{x})$, $N_b = 1000$ mixands in probability density function $p_b(\boldsymbol{x})$, and $n = 5$ states, then the number of operations required to compute the $H_{aa}$, $H_{ab}$, and $H_{bb}$ matrices would be on the order of $175 \times 10^6$. These numbers imply that far too many operations would be required for computation of the Eq.-(21) metric even in the restricted case that re-uses $R$ matrices in multiple mixands.

A practical solution to this computational complexity problem is to examine how well $p_b(\boldsymbol{x})$ fits $p_a(\boldsymbol{x})$ on a term-by-term basis using the sub-mixture decomposition described in Section II.B. Consider the ISD that characterizes the fit error between the new sub-mixture $p_{sbm}(\boldsymbol{x})$ and its original sub-mixture counterpart $p_{sam}(\boldsymbol{x})$:

$$J_{ISDm} = \breve{\boldsymbol{w}}_{am}^{\mathrm{T}} H_{aam} \breve{\boldsymbol{w}}_{am} - 2\breve{\boldsymbol{w}}_{am}^{\mathrm{T}} H_{abm} \breve{\boldsymbol{w}}_{bm} + \breve{\boldsymbol{w}}_{bm}^{\mathrm{T}} H_{bbm} \breve{\boldsymbol{w}}_{bm} \tag{22}$$

where the weight vectors within the sub-mixtures are defined as $\breve{\boldsymbol{w}}_{am} = [\, \breve{w}_{ai_{lom}}; \ldots; \breve{w}_{ai_{him}} \,]$ and $\breve{\boldsymbol{w}}_{bm} = [\, \breve{w}_{bj_{lom}}; \ldots; \breve{w}_{bj_{him}} \,]$. The matrix $H_{aam}$ is a sub-matrix of the original $H_{aa}$ that appears in Eq. (15), the one along rows and columns $i_{lom}$ through $i_{him}$. Similarly, $H_{abm}$ equals the sub-matrix of $H_{ab}$ along rows $i_{lom}$ through $i_{him}$ and columns $j_{lom}$ through $j_{him}$, and $H_{bbm}$ is the $H_{bb}$ sub-matrix along rows and columns $j_{lom}$ through $j_{him}$. Consider, also, a modified version of the sub-mixture ISD. It accounts for possible differences between the total weights of the original and new sub-mixture components, differences between $w_{sam}$ and $w_{sbm}$:

$$\hat{J}_{ISDm} = \breve{\boldsymbol{w}}_{am}^{\mathrm{T}} H_{aam} \breve{\boldsymbol{w}}_{am} - 2\breve{\boldsymbol{w}}_{am}^{\mathrm{T}} H_{abm} \breve{\boldsymbol{w}}_{bm} \left( \frac{w_{sbm}}{w_{sam}} \right) + \breve{\boldsymbol{w}}_{bm}^{\mathrm{T}} H_{bbm} \breve{\boldsymbol{w}}_{bm} \left( \frac{w_{sbm}}{w_{sam}} \right)^2 \tag{23}$$

One can use the modified ISD in Eq. (23) to construct the following conservative upper bound for the relative fit-error in Eq. (21):

$$\frac{\| p_a(\boldsymbol{x}) - p_b(\boldsymbol{x}) \|_2}{\| p_a(\boldsymbol{x}) \|_2} \leq \frac{\sum_{m=1}^{M} w_{sam} \sqrt{\hat{J}_{ISDm}}}{\sqrt{\sum_{m=1}^{M} w_{sam}^2 \breve{\boldsymbol{w}}_{am}^{\mathrm{T}} H_{aam} \breve{\boldsymbol{w}}_{am}}} = \frac{\sum_{m=1}^{M} w_{sam} \left\| p_{sam}(\boldsymbol{x}) - \left( \frac{w_{sbm}}{w_{sam}} \right) p_{sbm}(\boldsymbol{x}) \right\|_2}{\sqrt{\sum_{m=1}^{M} w_{sam}^2 \| p_{sam}(\boldsymbol{x}) \|_2^2}} \tag{24}$$

The derivation of the left-hand inequality in Eq. (24) relies on the triangle inequality

$$\| p_a(\boldsymbol{x}) - p_b(\boldsymbol{x}) \|_2 \leq \sum_{m=1}^{M} \| w_{sam} p_{sam}(\boldsymbol{x}) - w_{sbm} p_{sbm}(\boldsymbol{x}) \|_2 \ , \tag{25}$$

on the inequality

$$\| p_a(\boldsymbol{x}) \|_2^2 \geq \sum_{m=1}^{M} w_{sam}^2 \| p_{sam}(\boldsymbol{x}) \|_2^2 \ , \tag{26}$$

and on straightforward manipulations of Eqs. (25) and (26). The result in Eq. (26) can be proved as follows: Replace $p_a(\boldsymbol{x})$ on the left-hand side by the Eq. (11a) sum of sub-mixtures. Next, expand the result into a double sum involving squared functional norms and functional inner products. Finally, eliminate the inner products to change the equality into an inequality because all of the inner products are non-negative due to the fact that $p_{sam}(\boldsymbol{x}) \geq 0$ for all $\boldsymbol{x}$ and all $m$.

It is possible that $p_{sbm}(\boldsymbol{x})$ will be undefined for one or more values of $m$. This will happen if no elements have been assigned in the new Gaussian mixture to approximate the corresponding original $p_{sam}(\boldsymbol{x})$ sub-mixture. The term $(w_{sbm}/w_{sam})p_{sbm}(\boldsymbol{x})$ in the numerator on the extreme right-hand side of Eq. (24) is replaced by zero for all such values of $m$.

Equation (24) implies that its right-most expression is a conservative measure of how well the new Gaussian mixture $p_b(\boldsymbol{x})$ fits the original mixture $p_a(\boldsymbol{x})$. Suppose that the new mixture yields a value for this expression that is sufficiently small relative to 1. Then the original relative fit measure, $e_{relba}$ from Eq. (21), is at least this small, and $p_b(\boldsymbol{x})$ is a good approximation of $p_a(\boldsymbol{x})$.

The importance of Eq. (24) is that its conservative right-most bound is inexpensive to compute. It only requires the computation of blocks along the diagonals of the original $H_{aa}$, $H_{ab}$, and $H_{bb}$ matrices of Eq. (15). The following expression gives the scaling law of the leading term in the number of computations needed to produce all of these matrix sub-blocks:

$$\text{Number of ops} \ \sim \ n^3 \sum_{m=1}^{M} (N_{am}^2 + N_{am} N_{bm} + N_{bm}^2) \tag{27}$$

Again, if most of the QR factorizations in Eq. (18) can be eliminated due to re-use of $R$ square-root information matrices for many mixands, then the leading $n^3$ term changes to $n^2$. Consider the same problem dimensions as discussed in the beginning of this section, i.e., $N_a$ = 2000, $N_b$ = 1000, and $n$ = 5. Assume, also, that Gaussian mixtures $p_a(\boldsymbol{x})$ and $p_b(\boldsymbol{x})$ have both been decomposed into $M$ = 500 sub-mixtures with $N_{am}$ = 4 components for each original $p_{sam}(\boldsymbol{x})$ sub-mixture and with $N_{bm}$ = 2 components for each new $p_{sbm}(\boldsymbol{x})$ sub-mixture. Then the number of operations given in Eq. (27), but with $n^3$ replaced by $n^2$, is $350 \text{x} 10^3$. This is smaller by a factor of 500 than the cost of exact computation of $e_{relba}$. Therefore, this conservative measure of the $p_b(\boldsymbol{x})$ approximation accuracy is preferred based on computational considerations.

## IV. Weight Calculation for New Gaussian Mixture Components using ISD and Quadratic Programming

### A. Quadratic Programs to Minimize the ISDs of New Sub-Mixtures

This paper's re-approximation algorithm develops a new set of Gaussian mixture elements and weights to define $p_b(\boldsymbol{x})$ in a way that seeks to closely approximate $p_a(\boldsymbol{x})$. In order to avoid too much computation, it adopts a divide-and-conquer approach in which it breaks $p_a(\boldsymbol{x})$ into the weighted sub-mixtures $p_{sam}(\boldsymbol{x})$ for $m$ = 1, ..., $M$, as defined in Eq. (10a). For each original sub-mixture $p_{sam}(\boldsymbol{x})$ that has a sufficiently large weight $w_{sam}$, the algorithm determines components and weights of a new sub-mixture $p_{sbm}(\boldsymbol{x})$ that enable it to approximate $p_{sam}(\boldsymbol{x})$ with sufficient accuracy. The procedures for picking the means and square-root information matrices of the new sub-mixture components are discussed later, in Sections V and VII.

The present section develops a method for choosing the component weights of the new sub-mixture, $\breve{\boldsymbol{w}}_{bm} = [\breve{w}_{bj_{lom}}; ...; \breve{w}_{bj_{him}}]$. This method is used in the 7th block of the algorithm flow chart in Fig. 1. This procedure starts from the assumption that the new means and square-root information matrices have already been chosen. Note that this section's weight calculation algorithm could be matched with any algorithm that selects the means and covariances of new mixands, e.g., the algorithms defined in Refs. 9, 11, and 12.

A good method of choosing $\breve{\boldsymbol{w}}_{bm}$ is to minimize the value of the fit error between $p_{sbm}(\boldsymbol{x})$ and $p_{sam}(\boldsymbol{x})$, $J_{ISDm}$ from Eq. (22). Under the assumptions of this section, the only unknown quantity on the right-hand side of Eq. (22) is the relative weight vector $\breve{\boldsymbol{w}}_{bm}$. The fit error cost in Eq. (22) has terms that are linear and quadratic in this vector. Therefore, the optimal fit is obtained by solving the following constrained quadratic program (QP):

find: $\breve{\boldsymbol{w}}_{bm}$ (28a)

to minimize: $J_m = \boldsymbol{g}_{bm}^{\mathrm{T}} \breve{\boldsymbol{w}}_{bm} + \frac{1}{2} \breve{\boldsymbol{w}}_{bm}^{\mathrm{T}} H_{bbm} \breve{\boldsymbol{w}}_{bm}$ (28b)

subject to: $1 = \boldsymbol{c}^{\mathrm{T}} \breve{\boldsymbol{w}}_{bm}$ (28c)

$0 \le \breve{\boldsymbol{w}}_{bm}$ (28d)

where $\boldsymbol{g}_{bm} = -H_{abm}^{\mathrm{T}} \breve{\boldsymbol{w}}_{am}$ and $\boldsymbol{c}$ is an $N_{bm}$-dimensional vector of ones. The cost function $J_m$ has been derived from $J_{ISDm}$ by subtracting off the first term on the right-hand side of Eq. (22), the one that does not depend on $\breve{\boldsymbol{w}}_{bm}$, and then halving the result. This has been done in order to frame the QP in a standard form [14]. The value of $\breve{\boldsymbol{w}}_{bm}$ that minimizes the cost in Eq. (28b) also minimizes $J_{ISDm}$ from Eq. (22). Equation (28c) is the scalar unit normalization equality constraint. Equation (28d) constitutes $N_{bm}$ separate scalar inequality constraints on the elements of $\breve{\boldsymbol{w}}_{bm}$. These two constraints guarantee that sub-mixture $p_{sbm}(\boldsymbol{x})$ will be a unit-normalized, non-negative probability density function.

**B. Quadratic Program Solution Strategies**

The linearly-constrained quadratic program in Eqs. (28a)-(28d) can be solved using standard active-set methods such as those described in Ref. 14. There exist standard software packages for solving such problems, e.g., the MATLAB function quadprog.m, which is part of MATLAB's optimization toolbox.

There are three reasons to develop special software for solving this QP. First, the Hessian matrix $H_{bbm}$ will be positive definite for a well chosen set of mixture components of $p_{sbm}(\boldsymbol{x})$ that have minimal overlap with each other. This fact can be exploited to speed the solution of the QP.

If, on the other hand, $H_{bbm}$ is not sufficiently positive definite, then the QP should be restarted after choosing one or more alternate new components of sub-mixture $p_{sbm}(\boldsymbol{x})$. A specially-designed QP algorithm could determine whether $H_{bbm}$ was not sufficiently positive definite. Such a determination could signal this paper's re-sampling algorithm to replace one or more components of sub-mixture $p_{sbm}(\boldsymbol{x})$.

The second reason for developing a special-purpose QP algorithm is to exploit the method by which components of $p_{sbm}(\boldsymbol{x})$ are chosen. This paper proposes choosing one new component at a time and computing a new relative weight vector $\breve{\boldsymbol{w}}_{bm}$ after adding each new component. A general-purpose QP solution would require order $N_{bm}^3$ scalar operations for the factorization of its $N_{bm}$-by-$N_{bm}$ Hessian matrix. A special-purpose algorithm could exploit the following facts: each successive QP involves the addition of only one new row and column to $H_{bbm}$, one new element to $\boldsymbol{g}_{bm}$, one new element to $\boldsymbol{c}$, and one new scalar inequality constraint in Eq. (28d). A well defined rank-1 update and re-solution based on the previous QP solution typically would require only order $N_{bm}^2$ scalar operations. Such an approach could save a considerable amount of computation.

The third reason for developing a special-purpose QP algorithm is also related to the method of choosing new mixture components. When a single new mixture component is added to $p_{sbm}(\boldsymbol{x})$, the corresponding element of the new optimal $\breve{\boldsymbol{w}}_{bm}$ may equal 0. If this happens, then the other elements of the optimal $\breve{\boldsymbol{w}}_{bm}$ all remain unchanged. This situation indicates a poor choice of the new component, one that should be rejected. A special-purpose QP solution algorithm would be able to check for this condition using only order $N_{bm}^2$ calculations.

A good strategy for developing a special-purpose QP solver is to Cholesky factorize the Hessian matrix and to use that factorization in order to define a transformed weight vector and a transformed QP. Suppose that $L_m$ is the lower-triangular Cholesky factor of $H_{bbm}$:

$$L_m L_m^{\mathrm{T}} = H_{bbm} \tag{29}$$

The transformed QP is posed in terms of the transformed weight vector $\hat{\boldsymbol{w}}_{bm} = L_m^{\mathrm{T}} \breve{\boldsymbol{w}}_{bm}$. It takes the equivalent form:

find: $\hat{\boldsymbol{w}}_{bm}$ (30a)

to minimize: $J_m = \hat{\boldsymbol{g}}_{bm}^{\mathrm{T}} \hat{\boldsymbol{w}}_{bm} + \frac{1}{2} \hat{\boldsymbol{w}}_{bm}^{\mathrm{T}} \hat{\boldsymbol{w}}_{bm}$ (30b)

subject to: $1 = \hat{\boldsymbol{c}}_m^{\mathrm{T}} \hat{\boldsymbol{w}}_{bm}$ (30c)

$0 \le L_m^{-\mathrm{T}} \hat{\boldsymbol{w}}_{bm}$ (30d)

where $\hat{\boldsymbol{g}}_{bm} = L_m^{-1} \boldsymbol{g}_{bm}$ and $\hat{\boldsymbol{c}}_m = L_m^{-1} \boldsymbol{c}$. When applied to this transformed QP, an active-set method works with a guess of the inequality constraints in Eq. (30d) that are active, i.e., that are satisfied exactly as equalities. It QR-factorizes a matrix whose rows include $\hat{\boldsymbol{c}}_m^{\mathrm{T}}$ and the active rows of $L_m^{-\mathrm{T}}$. Given this factorization, it uses simple linear algebra operations and a line search to compute the optimum under its active set assumption or to determine a

new inequality constraint to add to the active set. Once it reaches an optimum for a given assumption, it checks the Kuhn-Tucker multipliers for the active constraints to see whether any should be dropped from the active set in order to further decrease the cost.

The importance of the transformed problem lies in the fact that it can be efficiently re-solved after the addition to $p_{sbm}(\boldsymbol{x})$ of one new mixture component, and therefore, the addition of one element to $\breve{\boldsymbol{w}}_{bm}$. The corresponding additional row and column of $H_{bbm}$ can be used to compute a new $L_m^{-1}$ in order $N_{bm}^2$ scalar operations by adding one new row to the previous Cholesky factor. If the new mixand produces a new $H_{bbm}$ that is insufficiently positive definite, then this fact will become apparent in the process of computing the new row of $L_m^{-1}$, and the new mixand can be rejected. The corresponding updates of $\hat{\boldsymbol{g}}_{bm}$ and $\hat{\boldsymbol{c}}_m$ are computable in order $N_{bm}$ operations, and the corresponding rank-1 update of the active set constraint matrix can be computed in order $N_{bm}^2$ in the usual case of few active constraints. Of course, order $N_{bm}^2$ scalar operations will still be a large number of operations if $N_{bm}$ is large. Therefore, care will be taken to avoid creating more elements of the sub-mixture $p_{sbm}(\boldsymbol{x})$ than are absolutely necessary. Strategies for achieving this aim are discussed in Sections VI and VII.

## V. LMI Bounds on the Covariances of the New Mixture's Components

### A. Covariance and Square-Root Information Matrix Bounds

This section defines and solves a Linear Matrix Inequality. This solution is needed in order to compute the constrained covariances of the new mixture elements, as per the 3rd block of the main algorithm in Fig. 1. The LMI is used to enforce the following lower bound on the information matrices of the elements of the new Gaussian mixture $p_b(\boldsymbol{x})$:

$$R_{bj}^{\mathrm{T}} R_{bj} \geq R_{min}^{\mathrm{T}} R_{min} \quad \text{for all } j = 1, \ldots, N_b \tag{31}$$

where the matrix inequality is defined in the sense that the symmetric matrix on the left minus the symmetric matrix on the right equals a positive semi-definite matrix. This lower bound on the information matrix of each mixture element translates into an upper bound on each element's covariance: $P_{bj} = R_{bj}^{-1} R_{bj}^{-\mathrm{T}} \leq R_{min}^{-1} R_{min}^{-\mathrm{T}} = P_{max}$. One can prove equivalence between this covariance inequality and Eq. (31) as follows: The latter matrix inequality is equivalent to $R_{min} R_{bj}^{-1} R_{bj}^{-\mathrm{T}} R_{min}^{\mathrm{T}} \leq I$. Equation (31) is equivalent to $R_{min}^{-\mathrm{T}} R_{bj}^{\mathrm{T}} R_{bj} R_{min}^{-1} \geq I$. The left-hand sides of these last two matrix inequalities are the inverses of each other. These last two inequalities are interchangeable because the first implies that the symmetric matrix on its left-hand side has eigenvalues all less than 1, and the second implies that its left-hand-side matrix has eigenvalues all greater than 1.

If the re-sampling algorithm must be constrained to choose the elements of $p_b(\boldsymbol{x})$ to have covariances less than $P_{max}$, then it suffices to enforce the LMI in Eq. (31). This LMI provides a means of trying to ensure that element-by-element UKF or EKF operations on the mixture, as per Ref. 7, will yield a good approximation of optimal Bayesian nonlinear filtering. An example in Section IX demonstrates that this approach works as desired if the UKF or EKF approximations are accurate over a range of state variations commensurate with $P_{max}$, i.e., if $P_{max}$ is sufficiently small. Choice of the bound $P_{max}$ is problem-dependent, and no general method has yet been developed for choosing $P_{max}$ based on the degree of nonlinearity of the filtering problem's model functions.

Each $R_{bj}$ square-root information matrix will be subject to at least one additional bound beyond that of Eq. (31). In its simplest version, the algorithm of Section VII chooses the $j^{\text{th}}$ component of $p_b(\boldsymbol{x})$ with the goal of improving the accuracy with which $p_b(\boldsymbol{x})$ approximates a particular element of $p_a(\boldsymbol{x})$, call it the $i^{\text{th}}$ element. In order for the re-sampling algorithm to work well, it is necessary that the covariance of the $j^{\text{th}}$ component of $p_b(\boldsymbol{x})$ not exceed the covariance of the corresponding $i^{\text{th}}$ component of $p_a(\boldsymbol{x})$. Otherwise, the re-sampling algorithm might not be able produce a good approximation of the $i^{\text{th}}$ component of $p_a(\boldsymbol{x})$ because the new approximation's covariance can be no smaller than the smallest covariance of any of its components. The resulting additional bound on the new element's square-root information matrix becomes

$$R_{bj}^{\mathrm{T}} R_{bj} \geq R_{ai}^{\mathrm{T}} R_{ai} \tag{32}$$

One might be tempted also to impose an LMI upper bound on $R_{bj}^{\mathrm{T}} R_{bj}$. Instead of enforcing an upper bound, an optimization of $R_{bj}$ provides a means of limiting the size of $R_{bj}^{\mathrm{T}} R_{bj}$.

### B. Optimal Solution to a Pair of LMIs

The standard algorithm for choosing $R_{bj}$ seeks the smallest resulting information matrix that satisfies the two LMIs in Eqs. (31) and (32). The smallest possible information matrix results in the largest possible covariance

matrix. This is a good choice because the largest possible covariance matrix tends to enable $p_b(\boldsymbol{x})$ to approximate $p_a(\boldsymbol{x})$ accurately with the fewest possible elements.

The optimal solution procedure for this LMI starts by computing the singular value decomposition of the matrix $R_{ai}R_{min}^{-1}$:

$$U_{bj}S_{bj}V_{bj}^{\mathrm{T}} = R_{ai}R_{min}^{-1} \tag{33}$$

where $U_{bj}$ and $V_{bj}$ are orthonormal matrices and $S_{bj}$ = diag($\sigma_{bj1}$,...,$\sigma_{bjn}$) is a diagonal matrix with the $n$ positive singular values $\sigma_{bj1}$, ..., $\sigma_{bjn}$ on its diagonal.

If $\sigma_{bjk} \geq 1$, for all $k$ = 1, ..., $n$, then $R_{bj} = R_{ai}$ respects the LMIs in Eqs. (31) and (33) in an optimal manner. Otherwise, one forms the $n$-by-$n$ diagonal matrix

$$\delta S_{bjfull} = \begin{bmatrix} \sqrt{\max(1-\sigma_{bj1}^2,0)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sqrt{\max(1-\sigma_{bjn}^2,0)} \end{bmatrix} \tag{34}$$

Next, one deletes all of the zero-valued rows of $\delta S_{bjfull}$ in order to form the matrix $\delta S_{bj}$. That is, row $k$ of $\delta S_{bjfull}$ is deleted for every $k$ such that $\sigma_{bjk} \geq 1$. This latter matrix is then used to form the matrix:

$$\delta R_{bj} = \delta S_{bj}V_{bj}^{\mathrm{T}}R_{min} \tag{35}$$

Finally, one uses QR factorization in order to compute $R_{bj}$ as follows:

$$Q_{bj}\begin{bmatrix} R_{bj} \\ 0 \end{bmatrix} = \begin{bmatrix} \delta R_{bj} \\ R_{ai} \end{bmatrix} \tag{36}$$

where $Q_{bj}$ is an orthonormal matrix and $R_{bj}$ is a square, upper-triangular matrix.

One can prove that this $R_{bj}$ matrix satisfies the LMIs of Eqs. (31) and (32) if one recognizes the following implication of Eq. (36): that

$$R_{bj}{}^{\mathrm{T}}R_{bj} = R_{ai}{}^{\mathrm{T}}R_{ai} + \delta R_{bj}{}^{\mathrm{T}}\delta R_{bj} \tag{37}$$

The LMI in Eq. (32) follows directly from this relationship. One can derive Eq. (31) by multiplying this relationship on the left by $R_{min}^{-\mathrm{T}}$ and on the right by $R_{min}^{-1}$. One can then substitute in Eqs. (33) and (35) to show that $R_{min}^{-\mathrm{T}}R_{bj}^{\mathrm{T}}R_{bj}R_{min}^{-1} = V_{bj}(S_{bj}^{\mathrm{T}}S_{bj} + \delta S_{bj}^{\mathrm{T}}\delta S_{bj})V_{bj}^{\mathrm{T}} = V_{bj}(S_{bj}^{\mathrm{T}}S_{bj} + \delta S_{bjfull}^{\mathrm{T}}\delta S_{bjfull})V_{bj}^{\mathrm{T}}$. The last matrix expression in parentheses is a diagonal matrix, all of whose diagonal elements are no less than 1. Therefore, $R_{min}^{-\mathrm{T}}R_{bj}^{\mathrm{T}}R_{bj}R_{min}^{-1} \geq I$, which is equivalent to Eq. (31).

The $R_{bj}$ matrix of Eq. (36) has two significant properties. First, it is optimal in that it minimizes both of the following squared weighted-norm metrics: Trace( $R_{min}^{-\mathrm{T}}R_{bj}^{\mathrm{T}}R_{bj}R_{min}^{-1}$ ) and Trace( $R_{ai}^{-\mathrm{T}}R_{bj}^{\mathrm{T}}R_{bj}R_{ai}^{-1}$ ). Second, consider the eigenvalues of the two matrix differences ($R_{bj}{}^{\mathrm{T}}R_{bj}$ - $R_{min}^{\mathrm{T}}R_{min}$ ) and ($R_{bj}{}^{\mathrm{T}}R_{bj}$-$R_{ai}{}^{\mathrm{T}}R_{ai}$). Both sets of eigenvalues are non-negative, in accordance with the LMIs in Eqs. (31) and (32). Consider the union of the eigenvalues of these two positive semi-definite matrices, as set of 2$n$ eigenvalues. It is straight-forward to prove that $n$ or more of these eigenvalues equal zero. These properties indicate that $R_{bj}{}^{\mathrm{T}}R_{bj}$ is as close as possible, in some matrix sense, to $R_{min}^{\mathrm{T}}R_{min}$ and to $R_{ai}{}^{\mathrm{T}}R_{ai}$. Closeness to $R_{ai}{}^{\mathrm{T}}R_{ai}$ tends to reduce the number of required new mixands for a given level of probability density approximation accuracy.

Note that the LMI solution $R_{bj}$ is not unique. It can be left-multiplied by any orthonormal matrix without changing any of the properties described in this sub-section, except for upper-triangularity. This non-uniqueness presents no problems. Any $R_{bj}$ square-root information matrix with the given properties will serve for the development of the new Gaussian mixture $p_b(\boldsymbol{x})$.

**C. Differential Covariance Matrix Square Roots**

Differential covariance matrices are used to develop the modified sub-mixture distributions from which new elements' means are drawn, as in the 6th block of Fig. 1. The matrix $\delta R_{bj}$ represents the square-root of an increment to an information matrix. The corresponding covariance increment is

$$\delta P_{aibj} = P_{ai} - P_{bj} = R_{ai}^{-1}R_{ai}^{-\mathrm{T}} - R_{bj}^{-1}R_{bj}^{-\mathrm{T}} = \delta Y_{aibj}\delta Y_{aibj}^{\mathrm{T}} \tag{38}$$

The matrix increment $\delta P_{aibj}$ is positive semi-definite, and $\delta Y_{aibj}$ is its matrix square-root. The following is a valid formula for this non-unique matrix square-root

$$\delta Y_{aibj} = R_{ai}^{-1} R_{ai}^{-\mathrm{T}} \delta R_{bj}^{\mathrm{T}} R_{djai}^{-1} \tag{39}$$

where the matrix $R_{djai}$ is determined from the QR factorization

$$Q_{dj} \begin{bmatrix} R_{djai} \\ 0 \end{bmatrix} = \begin{bmatrix} R_{ai}^{-\mathrm{T}} \delta R_{bj}^{\mathrm{T}} \\ I \end{bmatrix} \tag{40}$$

with $Q_{dj}$ being an orthonormal matrix and $R_{djai}$ being a square, upper-triangular matrix. One can prove that $\delta Y_{aibj}$ from Eq. (39) satisfies Eq. (38) by squaring the $\delta Y_{aibj}$ expression in Eq. (39), as on the right-hand side of Eq. (38), and by algebraically manipulating the result to show that it equals $R_{ai}^{-1} R_{ai}^{-\mathrm{T}} - R_{bj}^{-1} R_{bj}^{-\mathrm{T}}$. This manipulation requires the formula $R_{djai}^{-1} R_{djai}^{-\mathrm{T}} = I - \delta R_{bj} R_{bj}^{-1} R_{bj}^{-\mathrm{T}} \delta R_{bj}^{\mathrm{T}}$. This formula can be proved by squaring both sides of Eq. (40) to show that $R_{djai}^{\mathrm{T}} R_{djai} = I + \delta R_{bj} R_{ai}^{-1} R_{ai}^{-\mathrm{T}} \delta R_{bj}^{\mathrm{T}}$ and by using the classic matrix inversion lemma [15] along with a substitution based on Eq. (37). One completes the proof by substituting the $R_{djai}^{-1} R_{djai}^{-\mathrm{T}}$ expression into the expression for the square of $\delta Y_{aibj}$ and by performing several associative re-groupings of matrix multiplications and two substitutions for $\delta R_{bj}^{\mathrm{T}} \delta R_{bj}$ that are based on Eq. (37).

The square-root covariance increment matrix $\delta Y_{aibj}$ has only as many columns as $\delta R_{bj}$ has rows. This number equals the number of singular values of $S_{bj}$ that satisfy $\sigma_{bjk} < 1$.

It is possible to develop a similar expression for the covariance increment

$$\delta P_{maxbj} = P_{max} - P_{bj} = R_{min}^{-1} R_{min}^{-\mathrm{T}} - R_{bj}^{-1} R_{bj}^{-\mathrm{T}} = \delta Y_{maxbj} \delta Y_{maxbj}^{\mathrm{T}} \tag{41}$$

It takes the form:

$$\delta Y_{maxbj} = R_{min}^{-1} R_{min}^{-\mathrm{T}} \delta R_{bjalt}^{\mathrm{T}} R_{djmin}^{-1} \tag{42}$$

where

$$\delta R_{bjalt} = \delta S_{bjalt} U_{bj}^{\mathrm{T}} R_{ai} \tag{43a}$$

$$\delta S_{bjalt} = \begin{bmatrix} \sqrt{\max(1-\sigma_{bj1}^{-2}, 0)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sqrt{\max(1-\sigma_{bjn}^{-2}, 0)} \end{bmatrix} \text{ but with its all-zeros rows deleted} \tag{43b}$$

$$Q_{djmin} \begin{bmatrix} R_{djmin} \\ 0 \end{bmatrix} = \begin{bmatrix} R_{min}^{-\mathrm{T}} \delta R_{bjalt}^{\mathrm{T}} \\ I \end{bmatrix} \tag{43c}$$

where Eq. (43c) represents a QR factorization that produces the orthonormal matrix $Q_{djmin}$ and the square, upper-triangular matrix $R_{djmin}$.

Note that the square-root covariance increment matrices $\delta Y_{aibj}$ and $\delta Y_{maxbj}$ are not unique. They can be right-multiplied by any orthonormal matrix without changing their respective satisfaction of Eqs. (38) and (41). Any $\delta Y_{aibj}$ and $\delta Y_{maxbj}$ matrices that satisfy these equations will serve for this paper's re-sampling algorithm.

**D. Ad Hoc Solution to 3 LMIs**

It is necessary to compute an $R_{bj}$ matrix that satisfies 3 LMIs in the situation where a sub-mixture of $p_a(\boldsymbol{x})$ has 2 elements. One is the LMI in Eq. (31), the second is the LMI in Eq. (32), and the third LMI is similar to Eq. (32):

$$R_{bj}^{\mathrm{T}} R_{bj} \geq R_{ak}^{\mathrm{T}} R_{ak} \tag{44}$$

The index $k \neq i$ is that of a second component of Gaussian mixture $p_a(\boldsymbol{x})$.

An ad hoc solution to this system of 3 LMIs can be developed by applying two successive solutions of a 2-LMI problem, as in Sub-section V.B. Suppose that the 2-LMI algorithm of that sub-section is applied, but with $R_{ak}$ replacing $R_{min}$. Suppose that the resulting solution is $R_{bjtemp}$ and that the corresponding square-roots of the covariance matrix increments from Eqs. (39) and (42) are, respectively, $\delta Y_{aibjtemp}$ and $\delta Y_{akbjtemp}$. One can then find a solution to the 3-LMI problem by re-applying the 2-LMI algorithm of Sub-section V.B, only this time using $R_{bjtemp}$ in place of $R_{ai}$. This second solution will be the final $R_{bj}$, and it will satisfy Eqs. (31), (32), and (44). Suppose that the square-root of the covariance matrix increment that is generated by Eq. (39) for this second 2-LMI solution is $\delta Y_{bjtempbj}$. Then the total square-root covariance increments are

$$\delta Y_{aibj} = [\delta Y_{aibjtemp}, \delta Y_{bjtempbj}] \tag{45a}$$

$$\delta Y_{akbj} = [\delta Y_{akbjtemp}, \delta Y_{bjtempbj}] \tag{45b}$$

so that $P_{ai} - P_{bj} = \delta Y_{aibj} \delta Y_{aibj}^{T}$ and $P_{ak} - P_{bj} = \delta Y_{akbj} \delta Y_{akbj}^{T}$. It may be possible to use QR factorization of $\delta Y_{aibj}^{T}$ or $\delta Y_{akbj}^{T}$ in order to develop alternate square-root covariance increment matrices that preserve the values of $\delta Y_{aibj} \delta Y_{aibj}^{T}$ and $\delta Y_{akbj} \delta Y_{akbj}^{T}$, but with fewer columns in $\delta Y_{aibj}$ or $\delta Y_{akbj}$.

In this 3-LMI case, nothing definite can be said about the number of zero-eigenvalues in the set of $3n$ eigenvalues that is the union of the eigenvalues of the three matrix differences ($R_{bj}^{T} R_{bj} - R_{min}^{T} R_{min}$), ($R_{bj}^{T} R_{bj} - R_{ai}^{T} R_{ai}$), and ($R_{bj}^{T} R_{bj} - R_{ak}^{T} R_{ak}$). It is possible, maybe even likely, that this ad hoc algorithm is a sub-optimal solution to the 3 LMIs in Eqs. (31), (32), and (44). That is, it may fail to minimize any sensible norm of $R_{bj}^{T} R_{bj}$ subject to the three LMI constraints. If so, then an optimized $R_{bj}$ solution might improve part of this paper's Gaussian mixture re-sampling algorithm; it might enable a sub-mixture of $p_b(\boldsymbol{x})$ to accurately approximate a 2-element sub-mixture of $p_a(\boldsymbol{x})$ using a smaller number of new elements.

## VI. Algorithm for Decomposing the Original Gaussian Mixture into Sub-Mixtures

### A. The Need for a Decomposition of $p_a(\boldsymbol{x})$ into Sub-Mixtures

This section develops a strategy for decomposing original Gaussian mixture $p_a(\boldsymbol{x})$ into the Gaussian sub-mixtures, $p_{sam}(\boldsymbol{x})$ for $m = 1, ..., M$, as described in Sub-section II.B. This decomposition is used in the 1st block of Fig. 1's main re-sampling algorithm. It is needed for several reasons. First, as noted in Sub-section III.D, it is often impractical to compute the relative fit metric defined in Eq. (21). Therefore, sub-mixtures are needed to enable use of the more practical, but conservative, metric found on the extreme right-hand side of Eq. (24). Second, this paper's method for choosing weights for new elements relies on decomposition into sub-mixtures and solution of a separate low-order QP in order to determine the relative weights of each sub-mixture. One could pose a QP for the entire weight vector $\boldsymbol{w}_b$ by optimizing $J_{ISD}$ from Eq. (15). The resulting QP, however, often would be too large and require far too much computation to solve in a reasonable amount of time. Third, the method of selecting new elements of $p_b(\boldsymbol{x})$, as defined in Section VIII, adds new elements one at a time. Its method for selecting each new element is intimately linked with the decomposition of $p_a(\boldsymbol{x})$ into sub-mixtures.

### B. The Use of One- and Two-Component Sub-Mixtures of $p_a(\boldsymbol{x})$

A simple decomposition of $p_a(\boldsymbol{x})$ uses only single-element sub-mixtures. That is, it chooses $M = N_a$ and $p_{sam}(\boldsymbol{x}) = \mathcal{N}_{sr}(\boldsymbol{x}; \boldsymbol{\mu}_{am}, R_{am})$ for $m = 1, ..., N_a$. In the notation of Sub-section II.B, this choice would imply that $i_{lom} = i_{him} = m$ and that $N_{am} = 1$ for $m = 1, ..., M$. Although simple and sometimes effective, this decomposition has one significant drawback when used in conjunction with the algorithms of Sections VII and VIII: It will never allow those algorithms to merge a pair of very similar components of original Gaussian mixture $p_a(\boldsymbol{x})$ into a single new component of Gaussian mixture $p_b(\boldsymbol{x})$.

The ability to merge components can be important, and this capability is the subject of a number of research efforts, e.g., see Refs. 9, 11 and 12. In Gaussian mixture filtering applications, the dynamic propagation and measurement update processes can tend to make different mixture components converge towards each other over time. This is especially true in cases where an initially large state uncertainty converges to a much smaller uncertainty over time, as experienced during the project that produced Ref. 10.

Therefore, this paper's algorithm also considers 2-component sub-mixtures in its decomposition of Gaussian mixture $p_a(\boldsymbol{x})$. In general, the decomposition can consist of some sub-mixtures with one component and others with two components.

It might be sensible to consider the possibility of allowing some of the sub-mixtures of $p_a(\boldsymbol{x})$ to have more than two components. The consideration of higher numbers of sub-mixture components might enhance the algorithm's ability to reduce the number of components in going from mixture $p_a(\boldsymbol{x})$ to $p_b(\boldsymbol{x})$. In the interest of simplicity, however, the possibility of using more than two components has not been considered. The two-component limit enables the re-sampling algorithm to reduce the number of mixture components, when possible, while avoiding undue complexity.

There is no limit to $N_{bm}$, the number of elements in new sub-mixture $p_{sbm}(\boldsymbol{x})$, for any $m$ in the range 1 to $M$. The matrix LMI restriction on the covariance of the new components, if coupled with a limit on $N_{bm}$, could preclude the possibility of developing a $p_{sbm}(\boldsymbol{x})$ sub-mixture that accurately approximated its $p_{sam}(\boldsymbol{x})$ counterpart. Therefore, an upper bound on $N_{bm}$ could preclude $p_b(\boldsymbol{x})$ from ever being a sufficiently accurate approximation of $p_a(\boldsymbol{x})$.

**C. An Algorithm to Select the Sub-Mixtures of $p_a(\boldsymbol{x})$**

This sub-section describes how the sub-mixture decomposition is determined for an original Gaussian mixture $p_a(\boldsymbol{x})$. The decomposition seeks to pair two components into a single sub-mixture if they are deemed likely to be mergeable and if they have high enough weights. Multiple criteria are used to make a determination about mergeability, criteria that are somewhat like those given in Refs. 9 and 11. Note that no decision to merge two components is made at the point of deciding to pair them to form a single sub-mixture $p_{sam}(\boldsymbol{x})$. A sub-mixture pairing decision merely causes two components of $p_a(\boldsymbol{x})$ to become candidates for merging. An actual decision to merge is mechanized indirectly via the operations that select new components. They occur in Blocks 4-6 of Fig. 1 and are described in Sections VII and VIII.

The sub-mixture decomposition procedure works with a candidate set of components of mixture $p_a(\boldsymbol{x})$. Suppose that this set is indicated at the start of any given stage of this procedure by the corresponding indices of its components of $p_a(\boldsymbol{x})$, the index set $\{i_{c1}, i_{c2}, i_{c3}, \ldots, i_{cK}\}$. This set has $K$ elements, with $K \le N_a$. Suppose, also, that these indices have been sorted so that they are in order of descending weights. That is, $w_{ai_{c1}} \ge w_{ai_{c2}} \ge w_{ai_{c3}} \ge \ldots \ge w_{ai_{cK}}$ .

A stage of the decomposition procedure executes by considering $K$-1 candidate pairs for forming a new sub-mixture. The first component of each candidate pair is $\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_{ai_{c1}}, R_{ai_{c1}})$, and the second component of the $k^{\text{th}}$ candidate pair is $\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_{ak}, R_{ak})$ for $k = i_{c2}, i_{c3}, i_{c4}, \ldots, i_{cK}$. The procedure selects the pair $\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_{ai_{c1}}, R_{ai_{c1}})$ and $\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_{ak}, R_{ak})$ for the first index $k$ in this sequence which yields a pair that passes all of the procedure's criteria for possible merging, criteria which are defined below. These two components are used to define a new sub-mixture $p_{sam}(\boldsymbol{x})$, their corresponding indices are dropped from the set $\{i_{c1}, i_{c2}, i_{c3}, \ldots, i_{cK}\}$, which causes $K$ to decrement by 2, and the procedure repeats starting with the next available candidates for merging. If none of these $K$-1 pairs passes all of the criteria for possible merging, then a 1-component sub-mixture is formed from $\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_{ai_{c1}}, R_{ai_{c1}})$, $i_{c1}$ is dropped from the set of candidate indices, $K$ is decremented by 1, and the procedure continues. This procedure terminates when $K = 1$ or 0. If it terminates with $K = 1$, then the remaining component is used to form a sub-mixture $p_{sam}(\boldsymbol{x})$ that has only one component.

Before entering this procedure, the initial index set $\{i_{c1}, i_{c2}, i_{c3}, \ldots, i_{cK}\}$ is formed from all components of Gaussian mixture $p_a(\boldsymbol{x})$ whose weights are above a small minimum threshold. That is, $w_{ai_{ck}} \ge w_{min}$ for all $k = 1, \ldots, K$ for the initial set. If any original mixture component does not have a weight above the threshold $w_{min}$, then it is used to form a 1-component sub-mixture $p_{sam}(\boldsymbol{x})$. This positive threshold is a tuning parameter of the algorithm. Typically it is set very low, $w_{min} < 1/N_{bmax}$, where $N_{bmax}$ is the upper limit on the number of components in the new Gaussian mixture $p_b(\boldsymbol{x})$. It is not worthwhile to try to merge elements with very small weights because they are unlikely to be sampled by the new component selection algorithm of Section VII.

The following criteria are used to determine whether a given pair of components of $p_a(\boldsymbol{x})$ should be grouped to define a two-component sub-mixture by virtue of being reasonable candidates for merging. For notational convenience, let $\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_{ai}, R_{ai})$ and $\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_{ak}, R_{ak})$ be the two candidate components during the remainder of this discussion of pairing criteria. The first two criteria test whether the two means are sufficiently close:

$$\sqrt{[R_{ai}(\boldsymbol{\mu}_{ai}-\boldsymbol{\mu}_{ak})]^{\mathrm{T}}[R_{ai}(\boldsymbol{\mu}_{ai}-\boldsymbol{\mu}_{ak})]} \le \gamma\sqrt{n} \quad \text{and} \quad \sqrt{[R_{ak}(\boldsymbol{\mu}_{ai}-\boldsymbol{\mu}_{ak})]^{\mathrm{T}}[R_{ak}(\boldsymbol{\mu}_{ai}-\boldsymbol{\mu}_{ak})]} \le \gamma\sqrt{n} \tag{46}$$

where $\gamma$ is a tuning parameter of the algorithm. Typically $\gamma$ is chosen to be less than 1 in order to ensure that the two means are close in a statistical sense. A typical tuning value is $\gamma = 0.1$. Both of these inequalities must be satisfied in order for components $i$ and $k$ to be considered for possible grouping into a single sub-mixture $p_{sam}(\boldsymbol{x})$.

If the criteria in Eq. (46) are satisfied, then a second test is applied. It checks whether there is a small ISD between the proposed two-component sub-mixture and a single new Gaussian component candidate that tries to fit the entire proposed sub-mixture. In order to carry out this test, it is necessary to define the candidate new Gaussian component. Its mean equals the mean of the proposed sub-mixture

$$\boldsymbol{\mu}_{aik} = (w_{ai}\boldsymbol{\mu}_{ai} + w_{ak}\boldsymbol{\mu}_{ak})/(w_{ai} + w_{ak}) \tag{47}$$

Its square-root information matrix is calculated based on the technique of Sub-section V.B, but with $R_{ak}$ replacing $R_{min}$ in the calculations of that sub-section. Let this candidate square-root information matrix be called $R_{aik}$. It optimally satisfies the LMIs $R_{aik}^{\mathrm{T}}R_{aik} \ge R_{ai}^{\mathrm{T}}R_{ai}$ and $R_{aik}^{\mathrm{T}}R_{aik} \ge R_{ak}^{\mathrm{T}}R_{ak}$. Thus, it is the square-root information matrix of a Gaussian distribution whose covariance is no greater than the covariance of either the $i^{\text{th}}$ or the $k^{\text{th}}$ component of $p_a(\boldsymbol{x})$. The second test is based on the ISD between the candidate merged Gaussian component

$\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_{aik},R_{aik})$ and the candidate two-component sub-mixture $p_{samc}(\boldsymbol{x}) = [w_{ai}\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_{ai},R_{ai})+w_{ak}\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_{ak},R_{ak})]/(w_{ai}+w_{ak})$. This test is passed if

$$\sqrt{\frac{\int_{-\infty}^{\infty}[\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_{aik},R_{aik})-p_{samc}(\boldsymbol{x})]^2\,d\boldsymbol{x}}{\int_{-\infty}^{\infty}p_{samc}^2(\boldsymbol{x})d\boldsymbol{x}}} \le \lambda \tag{48}$$

where $\lambda$ is another tuning parameter of this algorithm. It is normally chosen to be significantly smaller than 1. A typical value is $\lambda = 0.1$. Note that the integral in the numerator on the left-hand side Eq. (48) is an ISD. It and the integral in the denominator can be evaluated analytically by using techniques defined in Sub-section III.B.

The two criteria in Eq. (46) and the one criterion in Eq. (48) all must be satisfied in order for a given pair of components of Gaussian mixture $p_a(\boldsymbol{x})$ to be deemed mergeable. If any pair satisfies these three criteria during the search procedure defined above, then they air paired to form a single actual $p_{sam}(\boldsymbol{x})$ sub-mixture of $p_a(\boldsymbol{x})$.

Note that it is possible for a particular pair of components to satisfy these three criteria and yet not be paired into a $p_{sam}(\boldsymbol{x})$ sub-mixture. This will happen if one or the other of the pair has already been successfully paired with a different component of higher weight. Thus, this pairing algorithm prefers pairs with the highest combined weight, subject to the constraints in Eqs. (46) and (48).

## VII. Algorithm for Choosing New Sub-Mixture Components to Fit a Corresponding Old Sub-Mixture

Part of the overall Gaussian mixture re-sampling algorithm involves choosing the elements of sub-mixture $p_{sbm}(\boldsymbol{x})$ so that it will closely approximate the original sub-mixture $p_{sam}(\boldsymbol{x})$ after its relative weights have been determined by solving the QP of Section IV. Recall that the elements of $p_{sbm}(\boldsymbol{x})$ are $\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_{bj},R_{bj})$ for $j = j_{lom}, ..., j_{him}$, as per Eq. (6b). The choice of elements of $p_{sbm}(\boldsymbol{x})$ involves choosing their means and square-root information matrices, $\boldsymbol{\mu}_{bj}$ and $R_{bj}$ for $j = j_{lom}, ..., j_{him}$. This section describes how these choices are made.

### A. Choosing Square-Root Information Matrices of New Components

The square-root information matrices are pre-selected to be the same for all components of a given sub-mixture of $p_b(\boldsymbol{x})$: $R_{bj} = R_{sbm}$ for $j = j_{lom}, ..., j_{him}$. This rule is consistent with the 3rd and 6th blocks of Fig. 1's algorithm. The common square-root information matrix $R_{sbm}$ is chosen by using the LMI solution procedure in Section V. If $p_{sam}(\boldsymbol{x})$ has only one component, $\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_{ai},R_{ai})$, then $R_{sbm}$ is chosen to satisfy the LMIs in Eqs. (31) and (32), as per Sub-section V.B. If $p_{sam}(\boldsymbol{x})$ consists of two component, $\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_{ai},R_{ai})$ and $\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_{ak},R_{ak})$, then $R_{sbm}$ is chosen to satisfy the LMIs in Eqs. (31), (32), and (44) as per Sub-section V.D.

The LMI constraints on $R_{sbm}$ enable $p_{sbm}(\boldsymbol{x})$ to better approximate $p_{sam}(\boldsymbol{x})$ because each component of $p_{sbm}(\boldsymbol{x})$ has a smaller covariance than each component of $p_{sam}(\boldsymbol{x})$. Therefore, it is possible to have a set of components of $p_{sbm}(\boldsymbol{x})$ with distributed means whose net total covariance equals that of $p_{sam}(\boldsymbol{x})$. If the covariances of the components of $p_{sbm}(\boldsymbol{x})$ were larger than those of the components of $p_{sam}(\boldsymbol{x})$, then this covariance matching might be difficult or even impossible because the covariance of a weighted sum of probability density functions can be no smaller, in an LMI sense, than the smallest covariance of any of its components. Thus, the LMI constraints on $R_{sbm}$ allow latitude in choosing the new components' means $\boldsymbol{\mu}_{bj}$ for $j = j_{lom}, ..., j_{him}$ while maintaining the new sub-mixture's potential to accurately model the covariance of the corresponding original sub-mixture.

For purposes of the remainder of this section, let $\delta Y_{aim}$ be the square root of the covariance increment $P_{ai} - P_{sbm} = R_{ai}^{-1}R_{ai}^{-\mathrm{T}} - R_{sbm}^{-1}R_{sbm}^{-\mathrm{T}} = \delta Y_{aim}\delta Y_{aim}^{\mathrm{T}}$, where $i$ is the index of the first component of $p_{sam}(\boldsymbol{x})$. If $p_{sam}(\boldsymbol{x})$ has two components, then let $k$ be the index of this second component, and let $\delta Y_{akm}$ be the square root of the covariance increment $P_{ak} - P_{sbm} = R_{ak}^{-1}R_{ak}^{-\mathrm{T}} - R_{sbm}^{-1}R_{sbm}^{-\mathrm{T}} = \delta Y_{akm}\delta Y_{akm}^{\mathrm{T}}$. Methods for computing $\delta Y_{aim}$ and $\delta Y_{akm}$ are described in Sub-sections V.C and V.D, though they are called $\delta Y_{aibj}$ and $\delta Y_{akbj}$ in those sub-sections.

### B. Initial Candidate Distribution for Choosing Means of New Components

A candidate method for generating the new components' mean values is to sample the following modified form of the original sub-mixture probability:

$$p_{\mu m}(\boldsymbol{x}) = \begin{cases} \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{ai}, \delta Y_{aim} \delta Y_{aim}^{\mathrm{T}}) & \text{if } p_{sam}(\boldsymbol{x}) \text{ has 1 element} \\ \breve{w}_{ai} \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{ai}, \delta Y_{aim} \delta Y_{aim}^{\mathrm{T}}) & \\ + \breve{w}_{ak} \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{ak}, \delta Y_{akm} \delta Y_{akm}^{\mathrm{T}}) & \text{if } p_{sam}(\boldsymbol{x}) \text{ has 2 elements} \end{cases} \tag{49}$$

That is, $p_{\mu m}(\boldsymbol{x})$ is almost the same as $p_{sam}(\boldsymbol{x})$, except for the following: The respective covariance of the first component is reduced from $R_{ai}^{-1} R_{ai}^{-\mathrm{T}}$ to $\delta Y_{aim} \delta Y_{aim}^{\mathrm{T}}$, and if there is a second component, then its covariance is reduced from $R_{ak}^{-1} R_{ak}^{-\mathrm{T}}$ to $\delta Y_{akm} \delta Y_{akm}^{\mathrm{T}}$.

The usefulness of the distribution $p_{\mu m}(\boldsymbol{x})$ can be understood by considering the following scenario: Suppose that $p_{sbm}(\boldsymbol{x})$ has many elements with their means independently sampled from $p_{\mu m}(\boldsymbol{x})$, their square-root information matrices all equal to $R_{sbm}$, and their weights all equal. One can show that the resulting distribution has the same mean and covariance as $p_{sam}(\boldsymbol{x})$ in the limit of a large number of components. This result relies on the variability of the mean values $\boldsymbol{\mu}_{bj}$ for $j = j_{lom}, ..., j_{him}$ to make up for the fact that each new component's covariance, $R_{sbm}^{-1} R_{sbm}^{-\mathrm{T}}$, is less than the covariance of any of the components of $p_{sam}(\boldsymbol{x})$. In fact, the $\delta Y_{aim}$ and $\delta Y_{akm}$ matrices have been specifically designed to compensate for any such covariance differences, as per Sub-sections V.C and V.D.

**C. A Modified Distribution for Choosing Means of New Components**

The above approach for choosing means and relative weights of new components relies too much on the brute-force statistical properties of large numbers. It achieves a high probability density in a given region by locating many equally-weighted new components close together in locations of $\boldsymbol{x}$ space where $p_{sam}(\boldsymbol{x})$ is large. A better approach would be to locate fewer components in such regions, but to increase their weights. The QP solution procedure of Section IV provides the needed means of increasing weights in regions of high $p_{sam}(\boldsymbol{x})$ values. The present sub-section develops a method for spreading out the means of the new elements in order to exploit the ability of QP-based weight selection to obviate the need for many new components with closely spaced mean values. This method is used in the 6th block of the Fig. 1 algorithm. Adequate spacing of new elements' mean values helps to avert the degeneracy problems that can occur in typical particle filters.

Suppose that the means $\boldsymbol{\mu}_{bj}$ for $j = j_{lom}, ..., (j_{him}\text{-}1)$ have been chosen for $p_{sbm}(\boldsymbol{x})$. There is no loss of generality in assuming that means have already been chosen for all but the last component of $p_{sbm}(\boldsymbol{x})$. The main algorithm of Section VIII adds a single component to a given $p_{sbm}(\boldsymbol{x})$ at any given stage of its procedure. The procedure always assumes that the newly added component may be the last. Under this assumption, a better approach for choosing the next mean of a new component is to sample it from the following modified probability density function:

$$p_{\mu hm}(\boldsymbol{x}) = C \left\{ \prod_{j=j_{lom}}^{j_{him}-1} \left[ 1 - e^{-0.5 (R_{sbm}\{\boldsymbol{x} - \boldsymbol{\mu}_{bj}\})^{\mathrm{T}} (R_{sbm}\{\boldsymbol{x} - \boldsymbol{\mu}_{bj}\})} \right] \right\} p_{\mu m}(\boldsymbol{x}) = C \pi_m(\boldsymbol{x}) p_{\mu m}(\boldsymbol{x}) \tag{50}$$

where $C$ is a normalization constant. The factor $\pi_m(\boldsymbol{x})$ is a scalar multiplicative factor involving terms of the form $1\text{-}e^{(*)}$. It takes on values in the range $0 < \pi_m(\boldsymbol{x}) < 1$. It takes on values very near 1 in regions of $\boldsymbol{x}$ space that are remote from the existing component mean values $\boldsymbol{\mu}_{bj}$ for $j = j_{lom}, ..., (j_{him}\text{-}1)$. In regions near the existing means, on the other hand, this scaling factor is very near 0. Thus, the new mean $\boldsymbol{\mu}_{bjhim}$ is nominally sampled from $p_{\mu m}(\boldsymbol{x})$, unless the resulting value would be too near to one of the existing mean values, $\boldsymbol{\mu}_{bj}$ for $j = j_{lom}, ..., (j_{him}\text{-}1)$.

An example $p_{\mu hm}(\boldsymbol{x})$ distribution is shown in Fig. 2 for a 1-dimensional $\boldsymbol{x}$ space. The black dash-dotted curve is $p_{\mu hm}(\boldsymbol{x})/C$, and the grey curve is $p_{\mu m}(\boldsymbol{x})$. The means of new mixands that have already been selected, $\boldsymbol{\mu}_{bj}$ for $j = j_{lom}, ..., (j_{him}\text{-}1)$, are 3.5, 6, and 7.5 in this case. $p_{\mu hm}(\boldsymbol{x})/C$ equals zero at these values, and it nearly equals $p_{\mu m}(\boldsymbol{x})$ remote from these values. Thus, the next mean, $\boldsymbol{\mu}_{bjhim}$, is unlikely to be near the values 3.5, 6, and 7.5. This is a good property because the pre-existing components are already able to approximate $p_{\mu m}(\boldsymbol{x})$ accurately near these values.

**D. How to Sample a New Mean from the Modified Distribution**

It is necessary to develop a special algorithm in order to properly sample from the probability density function in Eq. (50). This distribution is ideally suited to use a form of Metropolis-Hastings (M-H) sampling [13] because it is the product of the following three factors: the easily-sampled distribution $p_{\mu m}(\boldsymbol{x})$, the constant $C$, and the uniformly bounded function $\pi_m(\boldsymbol{x})$. The M-H algorithm for sampling $p_{\mu hm}(\boldsymbol{x})$ consists of an algorithm for sampling $p_{\mu m}(\boldsymbol{x})$ coupled with an algorithm for accepting or rejecting the sample based on evaluation of $\pi_m(\boldsymbol{x})$ at the current candidate sample and, typically, at a number of alternate candidate samples.

The algorithm for sampling the distribution $p_{\mu m}(\boldsymbol{x})$ of Eq. (49) is straightforward. If it has only one Gaussian component, then a Gaussian random vector is sampled from a distribution with a mean of zero and a covariance

equal to the identity matrix. Its dimension equals the number of columns of the square-root covariance matrix $\delta Y_{aim}$. This sampled vector is multiplied by $\delta Y_{aim}$, and the result is added to $\boldsymbol{\mu}_{ai}$ in order to produce a sample of $p_{\mu m}(\boldsymbol{x})$.

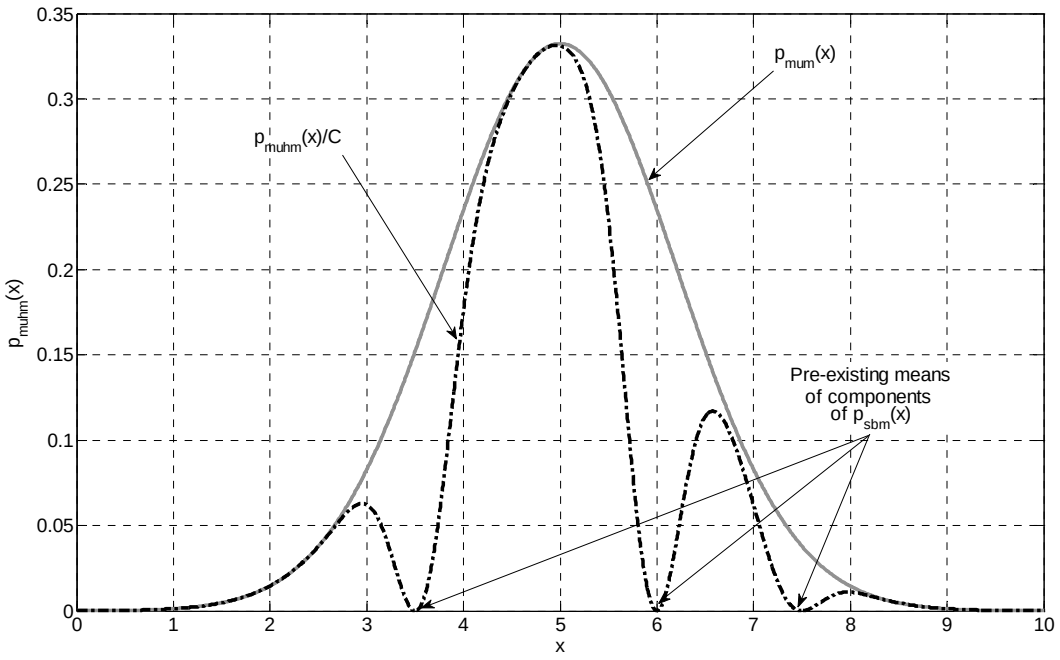


**Fig. 2. The original and modified sampling distribution functions for the mean values of components of a new sub-mixture, a 1-dimensional example.**

If $p_{\mu m}(\boldsymbol{x})$ has two components, then its sampling procedure must start with an importance sampling step. A scalar is sampled from the uniform distribution between 0 and 1 $\mathcal{U}[0,1]$. If that scalar is no greater than the relative weight $\breve{w}_{ai}$, then the desired sample of $p_{\mu m}(\boldsymbol{x})$ is drawn from the Gaussian distribution $\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_{ai},\delta Y_{aim}\delta Y_{aim}^{\mathrm{T}})$, as described in the preceding paragraph. If the uniform sample exceeds $\breve{w}_{ai}$, on the other hand, then the desired sample of $p_{\mu m}(\boldsymbol{x})$ is drawn from the alternate Gaussian distribution $\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_{ak},\delta Y_{akm}\delta Y_{akm}^{\mathrm{T}})$. If $p_{\mu m}(\boldsymbol{x})$ were allowed to have more than 2 Gaussian components, then this method would be modified to make the importance sampling step decide between the multiple Gaussian components based on their relative weights, as per standard procedures that are given in Ref. 2 and elsewhere.

Given an ability to sample from $p_{\mu m}(\boldsymbol{x})$, a mixture of the accept/reject method and an M-H method are used to sample from $p_{\mu hm}(\boldsymbol{x})$. Pseudo-code for this sampling algorithm is

Initialize $l = 0$, a counter of the number of M-H accept/reject cycles.
Draw $\alpha_l$ from $\mathcal{U}[0,1]$ and draw $\boldsymbol{x}_l$ from $p_{\mu m}(\boldsymbol{x})$.
If $\pi_m(\boldsymbol{x}_l) \geq \alpha_l$, then stop and accept $\boldsymbol{x}_l$.
While $l < l_{max}$
    $l = l + 1$.
    Draw $\alpha_l$ from $\mathcal{U}[0,1]$ and draw $\boldsymbol{x}_l$ from $p_{\mu m}(\boldsymbol{x})$.
    If $\pi_m(\boldsymbol{x}_l) \geq \alpha_l$, then stop and accept $\boldsymbol{x}_l$.
    Draw $\beta_l$ from $\mathcal{U}[0,1]$
    If $\beta_l \pi_m(\boldsymbol{x}_{l-1}) > \pi_m(\boldsymbol{x}_l)$
        $\boldsymbol{x}_l = \boldsymbol{x}_{l-1}$
    end
end
Let $\boldsymbol{\mu}_{bj_{him}} = \boldsymbol{x}_l$.

The M-H iteration limit $l_{max}$ must be chosen large enough to ensure that the final sample is from a distribution that nearly approximates $p_{\mu hm}(\boldsymbol{x})$ [13]. This limit must not be too large; otherwise, it wastes computational resources. Computational experience suggests that a good value might be $l_{max} = 30$.

One acceptance test for the sample takes the form: if $\pi_m(\boldsymbol{x}_l) \geq$ a sample from $\mathcal{U}[0,1]$, then stop and accept $\boldsymbol{x}_l$. This is a classic accept/reject test. It would suffice to generate the $p_{\mu hm}(\boldsymbol{x})$, but the accept/reject algorithm can be very inefficient.

The final if/end block implements the M-H part of the sampling method. It automatically keeps $\boldsymbol{x}_l$ as a better candidate sample from $p_{\mu hm}(\boldsymbol{x})$ if $\pi_m(\boldsymbol{x}_{l-1}) \leq \pi_m(\boldsymbol{x}_l)$ because $\beta_l$ sampled from $\mathcal{U}[0,1]$ will always obey $\beta_l \leq 1$. Whether or not it keeps $\boldsymbol{x}_l$ when $\pi_m(\boldsymbol{x}_{l-1}) > \pi_m(\boldsymbol{x}_l)$ depends on how small $\pi_m(\boldsymbol{x}_l)$ is relative to $\pi_m(\boldsymbol{x}_{l-1})$ and on how small the sampled value $\beta_l$ is, consistent with the M-H method. The accept/reject operations and the M-H operations both increase the likelihood of choosing a given $\boldsymbol{x}_l$ if its $\pi_m(\boldsymbol{x}_l)$ is near 1. This is exactly what is needed in order modify the distribution $p_{\mu m}(\boldsymbol{x})$ from Eq. (49) in order to yield samples from the $p_{\mu hm}(\boldsymbol{x})$ distribution of Eq. (50). Note that the algorithm never needs to know the scaling constant $C$.

**E. Additional QP-Based Accept/Reject Criteria**

Two additional accept/reject tests are applied to the candidate component mean value $\boldsymbol{\mu}_{bj_{him}}$. These criteria implement the decision in the 8th block of Fig. 1's main algorithm. They involve the new QP that gets generated, as per Eqs. (28a)-(28d) of Section IV. The new mean and the corresponding new component of $p_{sbm}(\boldsymbol{x})$ give rise to a new last component of the relative-weight QP solution vector $\breve{\boldsymbol{w}}_{bm}$, a new last row and column of the Hessian matrix $H_{bbm}$, and new last elements of the gradient vector $\boldsymbol{g}_{bm}$ and of the normalization constraint vector $\boldsymbol{c}$.

As discussed in Section IV, the new QP may have two deficiencies. One is that $H_{bbm}$ may not be sufficiently positive definite. The second is that the optimal value of the new component of $\breve{\boldsymbol{w}}_{bm}$ may be zero. Both of these situations indicate that the new Gaussian mixture component does not enable the augmented $p_{sbm}(\boldsymbol{x})$ sub-mixture to approximate the original sub-mixture $p_{sam}(\boldsymbol{x})$ with significantly better accuracy. In this case, the candidate mean $\boldsymbol{\mu}_{bj_{him}}$ is rejected, and the sampling algorithm of the previous sub-section is re-executed with the goal of generating a better candidate.

These rejection criteria tend to correlate with $\boldsymbol{\mu}_{bj_{him}}$ being too close to the mean of an existing element of $p_{sbm}(\boldsymbol{x})$, i.e., to one of the means $\boldsymbol{\mu}_{bj}$ for $j = j_{lom}, \ldots, (j_{him}-1)$. One would expect the sampling algorithm of the previous sub-section to avert this situation in most cases. Computational experience indicates that this situation is not completely precluded. Therefore, these two accept/reject criteria are worth including in the algorithm. Note, the first of these criteria maintains the uniqueness of the global solution of the QP in Eqs. (28a)-(28d), and it ensures that one can use a Cholesky-factorization-based solution procedure for this QP, as outlined in Section IV.

**F. Selection of Mean of First Sub-Mixture Component**

The algorithm of the previous two sub-sections presumes that at least one component of $p_{sbm}(\boldsymbol{x})$ has already been selected. That is, it assumes that $j_{him} - j_{lom} = 1$. Therefore, an auxiliary algorithm is needed in order to select the first mean value of the first component of $p_{sbm}(\boldsymbol{x})$. One reasonable choice is to sample $\boldsymbol{\mu}_{bj_{lom}}$ from the $p_{\mu m}(\boldsymbol{x})$ distribution of Eq. (49).

An alternative approach is to choose the initial mean $\boldsymbol{\mu}_{bj_{lom}}$ to equal the mean of $p_{sam}(\boldsymbol{x})$, which also equals the mean of $p_{\mu m}(\boldsymbol{x})$. An adaptation of the mean formula in Eq. (4) can be used to compute this mean value. The adaptation substitutes relative weights $\breve{w}_{ai}$ for absolute weights $w_{ai}$, and it sums only from $i_{lom}$ to $i_{him}$. Note: the merging criteria of Sub-section VI.C ensure that this mean will not lie in a region of low probability.

The use of the alternative scheme may be advisable for many situations. It is particularly important if the differential covariance $\delta Y_{aim}\delta Y_{aim}^{\mathrm{T}}$ is small and $p_{sam}(\boldsymbol{x})$ has only one component, or if $p_{sam}(\boldsymbol{x})$ has two components with each component's differential covariance being small and with both components being very much alike. In this case, a single component of $p_{sbm}(\boldsymbol{x})$ may be able to approximate $p_{sam}(\boldsymbol{x})$ to a very high degree of accuracy. The use of the mean of $p_{sam}(\boldsymbol{x})$ for $\boldsymbol{\mu}_{bj_{lom}}$ increases the likelihood of achieving such a good approximation.

There is no need to apply the additional accept/reject criteria of the previous sub-section to the first component of $p_{sbm}(\boldsymbol{x})$. The scalar Hessian $H_{bbm}$ is guaranteed to be positive definite in this special case, and the new element of $\breve{\boldsymbol{w}}_{bm}$ is guaranteed to be non-zero. In fact, it will be the only element. It must equal 1 because of the unit normalization constraint in Eq. (28c).

## VIII. Main Algorithm for Generating the New Gaussian Mixture $p_b(\boldsymbol{x})$

This section uses the definitions and results of Sections II-VII and several additional definitions in order to develop the main algorithm for generating the new Gaussian mixture $p_b(\boldsymbol{x})$ from the original Gaussian mixture $p_a(\boldsymbol{x})$. This algorithm starts by decomposing $p_a(\boldsymbol{x})$ into a set of $M$ 1- and 2-component sub-mixtures, as described in Section VI. Next, it recursively selects individual component sub-mixtures of $p_a(\boldsymbol{x})$ as candidates for improved approximation. This selection process is similar to importance re-sampling in a traditional particle filter [2], except that it uses a modified set of weights that reflect both the original sub-mixtures' weights and the residual fit errors between each original sub-mixture and its new counterpart. For the sub-mixture $p_{sam}(\boldsymbol{x})$ that is selected by this modified importance re-sampling procedure, the algorithm adds one new component to $p_{sbm}(\boldsymbol{x})$, as per Section VII, and it uses a QP to re-calculate the relative weights of the modified $p_{sbm}(\boldsymbol{x})$ in a way that minimizes its fit error, as per Section IV. This process repeats itself, adding one new component to $p_b(\boldsymbol{x})$ per recursion, until it has achieved a sufficiently good fit of $p_b(\boldsymbol{x})$ to $p_a(\boldsymbol{x})$, or until it reaches a pre-allotted number of Gaussian mixture components of $p_b(\boldsymbol{x})$.

### A. Modified Importance Re-Sampling

The modified re-sampling procedure uses the following relative fit parameter between the old and new sub-mixtures $p_{sam}(\boldsymbol{x})$ and $p_{sbm}(\boldsymbol{x})$:

$$f_m = \min\left[1, \sqrt{\frac{\breve{\boldsymbol{w}}_{am}^{\mathrm{T}} H_{aam} \breve{\boldsymbol{w}}_{am} - 2\breve{\boldsymbol{w}}_{am}^{\mathrm{T}} H_{abm} (\breve{\boldsymbol{w}}_{bm})_{opt} + (\breve{\boldsymbol{w}}_{bm})_{opt}^{\mathrm{T}} H_{bbm} (\breve{\boldsymbol{w}}_{bm})_{opt}}{\breve{\boldsymbol{w}}_{am}^{\mathrm{T}} H_{aam} \breve{\boldsymbol{w}}_{am}}}\right] \tag{51}$$

where $(\breve{\boldsymbol{w}}_{bm})_{opt}$ is the optimal solution of the QP in Eqs. (28a)-(28d) . Recall from the analysis of Section III that the right-hand argument of the min[ , ] function in Eq. (51) equals $\|p_{sbm}(\boldsymbol{x})-p_{sam}(\boldsymbol{x})\|_2/\|p_{sam}(\boldsymbol{x})\|_2$. The value of $f_m$ is initialized to 1 when there are not yet any components of new sub-mixture $p_{sbm}(\boldsymbol{x})$. The min[ , ] function in Eq. (51) restricts $f_m$ to be no greater than 1. As will become evident from what follows, this restriction avoids the possibility of amplifying the weight of an original sub-mixture due to a very poor fit between it and the corresponding new sub-mixture. These fit parameters are initialized in the 2nd block of the main algorithm of Fig. 1, and they are updated in the 7th block.

The modified sub-mixture weights used in the modified re-sampling procedure are

$$\tilde{w}_{sam} = \frac{w_{sam} f_m}{\sum_{l=1}^{M} w_{sal} f_l} \quad \text{for } m = 1, \ldots, M \tag{52}$$

These weights are computed in the 4th block of the Fig.-1 algorithm. They indicate both the importance of a given sub-mixture and the current inaccuracy of its fit by the corresponding sub-mixture of $p_b(\boldsymbol{x})$. The weight will be small if the original weight is small or if the current fit parameter is small, smallness indicating a good fit. In either case, the algorithm probably will not add a new component to the corresponding sub-mixture of $p_b(\boldsymbol{x})$. This will be a reasonable decision because larger modified weights of other sub-mixtures would indicate that those other sub-mixtures have relatively higher importance, relatively poorer fit accuracy for their corresponding $p_b(\boldsymbol{x})$ component, or both. The algorithm, therefore, would do well to improve the fit of one of those other sub-mixtures by giving it a new component.

The following modified importance re-sampling algorithm chooses which new sub-mixture to augment: First, sample $\eta$ from $\mathcal{U}[0,1]$. Next, seek the lowest index $m$ such that

$$\left\{\begin{matrix} 0 & \text{if } m = 1 \\ \sum_{l=1}^{m-1} \tilde{w}_{sal} & \text{if } m > 1 \end{matrix}\right\} \le \eta < \sum_{l=1}^{m} \tilde{w}_{sal} \tag{53}$$

This index chooses sub-mixture $p_{sbm}(\boldsymbol{x})$ as being the one for which a new component will be selected and for which new relative weights will be calculated in order to better fit the original sub-mixture $p_{sam}(\boldsymbol{x})$. This modified importance re-sampling algorithm is implemented in the 5th block of Fig. 1.

### B. Weights of New Sub-Mixtures

No information has yet been given about how the weight $w_{sbm}$ of the new sub-mixture $p_{sbm}(\boldsymbol{x})$ will be calculated. The chosen ad hoc method seeks to keep $w_{sbm}$ close to $w_{sam}$, the weight of the corresponding original sub-mixture. This technique is reasonable given that the components and relative weights of $p_{sbm}(\boldsymbol{x})$ are chosen in a

way that tries to fit $p_{sam}(\boldsymbol{x})$ accurately. An optimal method, on the other hand, might rely on definition and solution of some sort of QP. As noted in Section III, however, the size of such a QP might be prohibitively large.

The ad hoc algorithm tries to choose the new weights $w_{sbm} = w_{sam}$ for $m = 1, ..., M$, but it cannot use this choice if any $p_{sbm}(\boldsymbol{x})$ sub-mixture has no components. An empty sub-mixture can occur if the corresponding original weight $w_{sam}$ is very small. In this case, the $w_{sam}$ weights are used to define un-normalized $w_{sbm}$ weights for the non-empty $p_{sbm}(\boldsymbol{x})$ sub-mixtures, un-normalized weights of zero are assigned to the empty $p_{sbm}(\boldsymbol{x})$ sub-mixtures, and the final weights are determined via re-normalization.

The calculation procedure for the new weights relies on the mask weights

$$z_m = \begin{cases} 0 & \text{if } N_{bm} = 0 \\ 1 & \text{if } N_{bm} > 0 \end{cases} \tag{54}$$

Recall from the text after Eq. (8b) that $N_{bm}$ is the number of Gaussian components in $p_{sbm}(\boldsymbol{x})$. Given these values, the chosen weights of the new sub-mixtures are

$$w_{sbm} = \frac{w_{sam} z_m}{\sum_{l=1}^{M} w_{sal} z_l} \qquad \text{for } m = 1, ..., M \tag{55}$$

**C. Main Re-Sampling Algorithm**

Given the foregoing definitions, it is possible to define this paper's main Gaussian mixture re-sampling algorithm. It takes the following form, as defined using pseudo-code:

Decompose $p_a(\boldsymbol{x})$ into sub-mixtures $p_{sam}(\boldsymbol{x})$ with weights $w_{sam}$ and relative weight vectors $\breve{\boldsymbol{w}}_{am}$ for $m = 1, ..., M$. Do this using the decomposition definition of Sub-section II.B, the relative weight vector definition after Eq. (22) in Sub-section III.D, and the decomposition algorithm of Section VI.

Initialize the fit parameters and the mask weights to, respectively, $f_m = 1$ and $z_m = 0$ for $m = 1, ..., M$ as per Sub-sections VIII.A and VIII.B.

Compute the new square-root information matrices $R_{sbm}$ and the corresponding $\delta Y_{aim}$ and $\delta Y_{akm}$ square-root covariance increment matrices for $m = 1, ..., M$ as per Sub-section VII.A.

Pre-compute miscellaneous quantities that will be used in calculating the $H_{aam}$, $H_{abm}$, and $H_{bbm}$ matrices for $m = 1, ..., M$, as per Section III.B. These include the factor to the left of $e$ on the right-hand side of Eq. (17) and that equation's $\tilde{R}_{cd}$ matrix for all possible pairings of Gaussian components between $p_{sam}(\boldsymbol{x})$ and $p_{sbm}(\boldsymbol{x})$.

Initialize $N_b = 0$ and $j_{lom} = 1$, $j_{him} = 0$, $N_{bm} = 0$, and $\breve{\boldsymbol{w}}_{bm}$ = the empty vector for $m = 1, ..., M$. This initialization indicates that all of the $p_{sbm}(\boldsymbol{x})$ sub-mixtures start with no elements.

Initialize $E_{rel} = \infty$.

While ($N_b < N_{bmax}$) and $E_{rel} > E_{relmax}$

- Use Eq. (52) to re-compute the modified weights $\tilde{w}_{sam}$ for $m = 1, ..., M$
- Do modified importance re-sampling to select sub-mixture $p_{sbm}(\boldsymbol{x})$ as the one to which a new component will be added, as per Eq. (53) in Sub-section VIII.A.
- If $m < M$, then re-index the existing components of $p_b(\boldsymbol{x})$ in order to open up the quantities associated with index $(j_{him}+1) = j_{lo(m+1)}$ for the new component. That is, move $\boldsymbol{\mu}_{bj}$ into $\boldsymbol{\mu}_{b(j+1)}$ and $R_{bj}$ into $R_{b(j+1)}$ for $j = j_{lo(m+1)}, \ldots, N_b$. Afterwards, increment by 1 the quantities $j_{lol}$ and $j_{hil}$ for $l = (m+1), \ldots, M$.
- Increment by 1 the quantities $N_b$, $j_{him}$, and $N_{bm}$.
- Let $R_{bj_{him}} = R_{sbm}$.
- If $N_{bm} > 1$, then use the sampling and accept/reject methods of Sub-sections VII.D and VII.E in order to determine $\boldsymbol{\mu}_{bj_{him}}$; otherwise, determine $\boldsymbol{\mu}_{bj_{him}}$ as per Sub-section VII.F.
- Set up and solve the QP in Eqs. (28a)-(28d) in order to determine $\breve{\boldsymbol{w}}_{bm}$. Use the previous problem matrices, factorizations, and solution in order to perform low-rank updates to efficiently re-pose and re-solve the QP as discussed in Section IV.

Set $z_m = 1$ and use the new definition of $p_{sbm}(\boldsymbol{x})$ along with the new optimal relative weight vector $\breve{\boldsymbol{w}}_{bm}$ in order to update $f_m$ as per Eq. (51).
Update $w_{sbl}$ for $l = 1, \ldots, M$ as per Eq. (55).
Compute the updated $E_{rel}$ as the right-most term in Eq. (24).
end
For $l = 1{:}M$
For $j = j_{lol}{:}j_{hil}$
$w_{bj} = w_{sbl}[\breve{\boldsymbol{w}}_{bl}]_{(j-j_{lol}+1)}$
end
end

This algorithm is structured into 3 main sections. Everything before the main "while" loop constitutes an initialization and corresponds to the 1st-3rd blocks of Fig. 1. The main "while" loop implements blocks 4-10 of Fig. 1, adding one new component to $p_b(\boldsymbol{x})$ for each pass through its operations. This component is the new last element of sub-mixture $p_{sbm}(\boldsymbol{x})$. The main "while" loop terminates when $p_b(\boldsymbol{x})$ has $N_{bmax}$ Gaussian components or when $E_{rel}$, the conservative bound on the relative norm error between $p_a(\boldsymbol{x})$ and $p_b(\boldsymbol{x})$, falls to a value no greater than the upper limit $E_{relmax}$. This upper limit is typically set to a small number relative to 1, for example, something in the range 0.01 to 0.0001. The final pair of nested "for" loops computes the final absolute weights of the components of $p_b(\boldsymbol{x})$, as called for in the 11th block of Fig. 1. Each component's final weight equals the product of its relative weight within its sub-mixture and the absolute weight of that sub-mixture.

**D. Discussion of Algorithm**

The main algorithm has been designed with the goal of approximating elements of $p_a(\boldsymbol{x})$ with as small a number of $p_b(\boldsymbol{x})$ elements as possible. Suppose that an original mixture element has a sufficiently small covariance such that it does not violate the covariance upper limit on the new mixture elements, the limit defined in Eq. (31) using square-root information matrices. If the old mixture element's weight is large enough, then the new mixture will retain the old element exactly, and it will yield a perfect approximation of this component of the original mixture using only one component of the new mixture. This new component may have a modified weight if such a modification might allow it to also approximate another one of the original elements, one that is very similar to it. Thus, this algorithm will tend to be frugal about creating components of the new distribution if the components of its original distribution have sufficiently small covariances.

There are three computationally expensive parts of the algorithm. One is the pre-computation of new square-root information matrices, which occurs in the 3rd block of Fig. 1. The expensive parts of the corresponding LMI solutions are the singular-value decomposition in Eq. (33), the matrix algebra in Eqs. (33) and (35), and the QR factorization in Eq. (36). These calculations require order $Mn^3$ operations. Block 3 also contains the second expensive part, the pre-computation of quantities that are used to set up the QP problems for the relative weights. The expensive parts of this calculation are the QR factorization in Eq. (18) and the matrix algebra in Eq. (19). These calculations also require order $Mn^3$ operations. The third expensive component is the setting up and solving of the QP problems that determine the relative weights in the 7th block of Fig. 1. The expensive parts of the QP calculations are the computation of the off-diagonal elements of the $H_{bbm}$ Hessian matrices, as per Eqs. (16c) and (17), and the Cholesky factorizations of the Hessian matrices, as per Eq. (29). The cumulative count of these QP operations will be on the order of $\sum_{m=1}^{M}(n^2 N_{bm}^2 + N_{bm}^3)$ if the QP solutions do not require many changes of their active constraint sets. If the average $N_{bm}^2$ exceeds $n$ or if the average $N_{bm}^3$ exceeds $n^3$, then the QP operations in Block 7 constitute the dominant computational cost. Otherwise, the set-up operations in Block 3 dominate the cost.

This analysis does not consider the computational costs associated with the sub-mixture decomposition calculations in the 1st block of Fig. 1. The second test of candidate sub-mixture pairs, the one implemented in Eq. (48), requires many of the same calculations as are required for Block 3 of Fig. 1. The preceding analysis assumes that any such results from the execution of Block 1 are saved until the execution of Block 3 so that they can be used to reduce the number of computations that must be carried out in Block 3.

The algorithm in the previous sub-section reverts to the standard importance re-sampling of a particle filter [2] (one that does not include regularization), in the limit of vary small covariances of the Gaussian components. This limit corresponds to a very large $R_{min}$ minimum square-root information matrix for the components of the new mixture. In this case, the original and new Gaussian mixture components will be Dirac delta functions. The mean-value selection procedure of Sub-section VII.D collapses to a selection of means of the new components from the

set of means of the old components because the $\delta Y_{aim}$ and $\delta Y_{akm}$ matrices are very small in this case. Similarly, the modified re-sampling associated with Eq. (53) collapses to standard importance re-sampling because the $f_m$ values from Eq. (51) always equal 1, which causes the re-sampling weights in Eq. (52) to equal the original weights. The QP solution in Eqs. (28a)-(28d) will tend to yield nearly equal relative weights in this case. The expected number of elements of each new sub-mixture, when combined with the near equality of the relative weights from the QP, tends to produce equal weights in the main algorithm's final pair of nested "for" loops. All of these properties yield a filter that functions like a standard PF.

## IX. Examples of Algorithm Performance

The algorithm described in Sections II-VIII has been implemented in MATLAB and tested on several problems. Consider the $n$ = 2-dimensional example whose original Gaussian mixture $p_a(\boldsymbol{x})$ is depicted in the top plot of Fig. 3. The figure's 3-D view of the distribution lies along a direction that is almost parallel to the $x_1$-$x_2$ plane, thereby projecting the $x_1$-$x_2$ plane almost onto a horizontal line in the plot. This mixture has 5 elements. Their standard deviations range from 0.18 to 1.70 for the first element of $\boldsymbol{x}$ and from 0.47 to 1.39 for the second element. The maximum $p_b(\boldsymbol{x})$ component standard deviations, as imposed by $R_{min}$, are 1.04 for the first element of $\boldsymbol{x}$ and 0.95 for the second element.

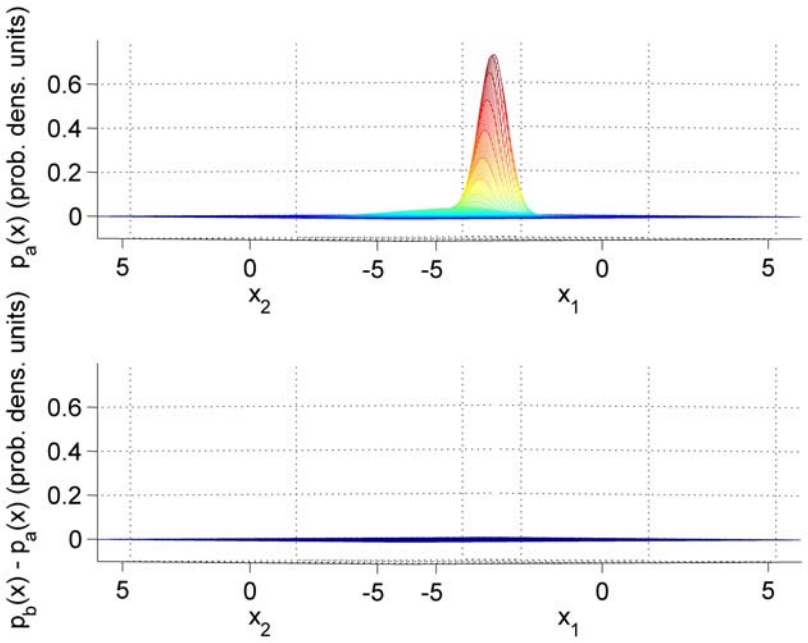


**Fig. 3. Original $N_a$= 5 element 2-D Gaussian mixture $p_a(\boldsymbol{x})$ -- top plot, and error of $N_b$ = 40 element Gaussian mixture $p_b(\boldsymbol{x})$ -- bottom plot.**

The fit of $p_b(\boldsymbol{x})$ was carried out using the following tuning parameters: $N_{bmax}$ = 100 components and $E_{relmax}$ = 0.01 to control termination criteria as in Section VIII; $w_{min}$ = 0.001, $\gamma$= 0.1, and $\lambda$ = 0.1 to control decomposition of $p_a(\boldsymbol{x})$ into sub-mixtures as in Section VI; $l_{max}$ = 30 iterations of the M-H sampling procedure as in Section VII.

The actual fit of $p_b(\boldsymbol{x})$ to $p_a(\boldsymbol{x})$ is very good, as shown on the fit error plot, the bottom plot of Fig. 3. The achieved conservative measure of the relative error is $E_{rel}$ = 0.0084, and the actual relative error norm, as determined numerically, is $\|p_b(\boldsymbol{x})-p_a(\boldsymbol{x})\|_2/\|p_a(\boldsymbol{x})\|_2$ = 0.0056, thus demonstrating the conservatism of the computed $E_{rel}$ as defined by the extreme right-hand term in Eq. (24). This good fit has been achieved using only $N_b$ = 40 components in the new distribution. Their standard deviations range from 0.18 to 0.82 for the first element of $\boldsymbol{x}$ and from 0.47 to 0.75 for the second element. Thus, they are narrower than the widest elements of $p_a(\boldsymbol{x})$ due to the covariance restrictions imposed by the $R_{min}$ LMI in Eq. (31).

The weights of the Gaussian components of $p_b(\boldsymbol{x})$ range from 0.0002 to 0.2912. The mean weight is $1/N_b$ = 0.025, and the standard deviation of the weights is 0.0451. Unlike particle filtering re-sampling, this procedure does

not produce equal-weighted components. This feature enables the new method to fit $p_b(\boldsymbol{x})$ to $p_a(\boldsymbol{x})$ with much greater accuracy for a given number of components.

Another useful metric of this algorithm's performance is the accuracy with which it approximates the mean and covariance of the original distribution as computed using Eq. (4). Reasonable metrics of the relative accuracies of the mean and covariance are

$$\Delta\mu = \sqrt{(\boldsymbol{\mu}_{gmb} - \boldsymbol{\mu}_{gma})^{\mathrm{T}} P_{gma}^{-1} (\boldsymbol{\mu}_{gmb} - \boldsymbol{\mu}_{gma})} \quad \text{and} \quad \Delta P = \|P_{gmb} - P_{gma}\| / \|P_{gma}\| \tag{56}$$

where the $()_{gma}$ subscripts indicate the mean and covariance of original Gaussian mixture $p_a(\boldsymbol{x})$, and the $()_{gmb}$ subscripts denote the new approximate mixture $p_b(\boldsymbol{x})$. The matrix norms in the $\Delta P$ expression are induced matrix 2-norms. These performance metrics evaluate to $\Delta\mu = 0.0042$ and $\Delta P = 0.0102$ for this example. Thus, both the mean and the covariance of $p_b(\boldsymbol{x})$ closely match those of $p_a(\boldsymbol{x})$ in a relative sense, despite the fact that there is no explicit matching constraint in the algorithm that constructs $p_b(\boldsymbol{x})$.

As a point of comparison, 40 particles have been sampled independently from $p_a(\boldsymbol{x})$, and they have been used to approximate the mean and covariance. The relative mean error for these 40 equal-weight Dirac delta functions is $\Delta\mu = 0.2748$, and the relative covariance error is $\Delta P = 0.2068$. Thus, the mean estimate from the 40 particles is 66 times less accurate than the mean estimate from the 40-component re-sampled Gaussian mixture, and the particles' covariance estimate is 20 times less accurate. The non-infinitesimal widths of the $p_b(\boldsymbol{x})$ Gaussian mixture components enable them to do a much better job of approximating $p_a(\boldsymbol{x})$.

Of course, MATLAB requires much less time to sample 40 particles from $p_a(\boldsymbol{x})$ than it requires to construct $p_b(\boldsymbol{x})$ using this paper's algorithm. A better comparison uses 8000 particles, which require about the same amount of processor time to sample as is required to construct $p_b(\boldsymbol{x})$. The average $\Delta\mu$ and $\Delta P$ fit metrics decrease significantly when using 8000 particles, but they are still larger than those achieved by the 40-component Gaussian mixture $p_b(\boldsymbol{x})$. The 8000 particles' average $\Delta\mu$ fit metric is larger by a factor of 3.7 than that of $p_b(\boldsymbol{x})$, and their average $\Delta P$ metric is 2.1 times larger. Thus, the new algorithm offers a clear advantage over particle methods for this example problem.

This paper's algorithm could have fit $p_b(\boldsymbol{x})$ to $p_a(\boldsymbol{x})$ very accurately with many fewer mixands if the LMI bound in Eq. (31) had been less restrictive. The re-sampling algorithms in Refs. 9, 11, and 12 could have done the same. With a very loose LMI bound, the algorithm could achieve a perfect fit with just 3 elements, the original elements. The next example, however, demonstrates the usefulness of adding extra elements, even many extra elements, in order to satisfy an LMI that enforces a restrictive bound on the new elements' covariances.

The required number of components of $p_b(\boldsymbol{x})$ grows significantly if the maximum covariance of each component is reduced, that is, if $R_{min}$ increases in the LMI of Eq. (31). This is illustrated by the example in Fig. 4. The original $p_a(\boldsymbol{x})$ has 3-components and is plotted in solid blue along the horizontal axis. The approximate $p_b(\boldsymbol{x})$ has 100 components and is plotted as the dash-dotted red curve along the same axes. The 3 components of $p_a(\boldsymbol{x})$ are plotted as dashed green curves, and the 100 components of $p_b(\boldsymbol{x})$ are plotted as dotted blue-grey curves. The standard deviations of the components of $p_a(\boldsymbol{x})$ are 0.42, 0.75, and 2.06. The components of $p_b(\boldsymbol{x})$ all have the same standard deviation: 0.20, the upper limit imposed by $R_{min}$. As can be seen from Fig. 4, $p_b(\boldsymbol{x})$ approximates $p_a(\boldsymbol{x})$ very well. The relative fit error is $\|p_b(\boldsymbol{x}) - p_a(\boldsymbol{x})\|_2 / \|p_a(\boldsymbol{x})\|_2 = 0.0034$. The cost of achieving this good fit is the need to use 100 components to construct $p_b(\boldsymbol{x})$.

Figure 4 illustrates an important point about why this Gaussian mixture re-sampling method has been developed. It plots an example nonlinear function $\boldsymbol{f}(\boldsymbol{x})$ as the dash-dotted black curve. It also shows the exact propagation of the probability density function $p_a(\boldsymbol{x})$ through $\boldsymbol{f}(\boldsymbol{x})$ to produce the corresponding probability density function for $\boldsymbol{f}$: $p_f(\boldsymbol{f}) = p_a[\boldsymbol{x}(\boldsymbol{f})] / |\partial \boldsymbol{f} / \partial \boldsymbol{x}|$, where $\boldsymbol{x}(\boldsymbol{f})$ represents the function inverse of $\boldsymbol{f}(\boldsymbol{x})$. This probability density is plotted as the solid blue distribution that is shown along the left-hand vertical axis (after being moved to have its zero value line up at the horizontal position $\boldsymbol{x} = -12$ and after being scaled down by a factor of 3 in order to fit well within the figure's horizontal range). $p_f(\boldsymbol{f})$ is plotted along the vertical $\boldsymbol{f}$ axis because $\boldsymbol{f}$ is its independent variable. Also plotted on that axis are two approximations of $p_f(\boldsymbol{f})$. The dashed green curve is the $p_f(\boldsymbol{f})$ that results from performing EKF-type propagations through $\boldsymbol{f}(\boldsymbol{x})$ of the 3 components of $p_a(\boldsymbol{x})$. The dash-dotted red curve is similar, except that it applies the EKF-type propagations to the 100 components of $p_b(\boldsymbol{x})$. It is obvious from this plot that the latter approximation is much closer to the truth. It even reproduces the bi-modal peaks of the true distribution. Thus, there can be significant benefit in terms of nonlinear filtering accuracy if one re-approximates $p_a(\boldsymbol{x})$ by a Gaussian mixture $p_b(\boldsymbol{x})$ with bounded covariances on each of its components.

A different calculation is required in order to illustrate the benefits of using a re-sampled Gaussian mixture with bounded component covariances when performing the measurement update of a nonlinear filter. Suppose that $p_a(\boldsymbol{x})$

of Fig. 4 is the *a priori* probability distribution for $\boldsymbol{x}$, and suppose that $\boldsymbol{f}(\boldsymbol{x})$ of Fig. 4 is a nonlinear measurement function rather than a nonlinear dynamic propagation function. Suppose that the measurement model takes the form:

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{\nu} \tag{57}$$

where $\boldsymbol{y}$ is the observed measurement vector and $\boldsymbol{\nu}$ is a Gaussian measurement noise vector with a mean of zero and a covariance of $P_{\nu\nu}$. Then Bayes' rule dictates that the *a posteriori* probability distribution of $\boldsymbol{x}$ is

$$p_{posterior}(\boldsymbol{x}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{x}) p_a(\boldsymbol{x})}{\int_{-\infty}^{\infty} p(\boldsymbol{y} \mid \boldsymbol{x}) p_a(\boldsymbol{x}) d\boldsymbol{x}} = C e^{-0.5[\boldsymbol{y}-f(\boldsymbol{x})]^{\mathrm{T}} P_{\nu\nu}^{-1}[\boldsymbol{y}-f(\boldsymbol{x})]} p_a(\boldsymbol{x}) \tag{58}$$

where $C$ is a normalization constant. This posterior distribution can be approximated as a Gaussian sum by using EKF or UKF calculations to do individual updates for each of the Gaussian components followed by re-weighting of the components. The re-weighting is based on chi-squared statistics of the components' normalized innovations, as in Ref. 7.

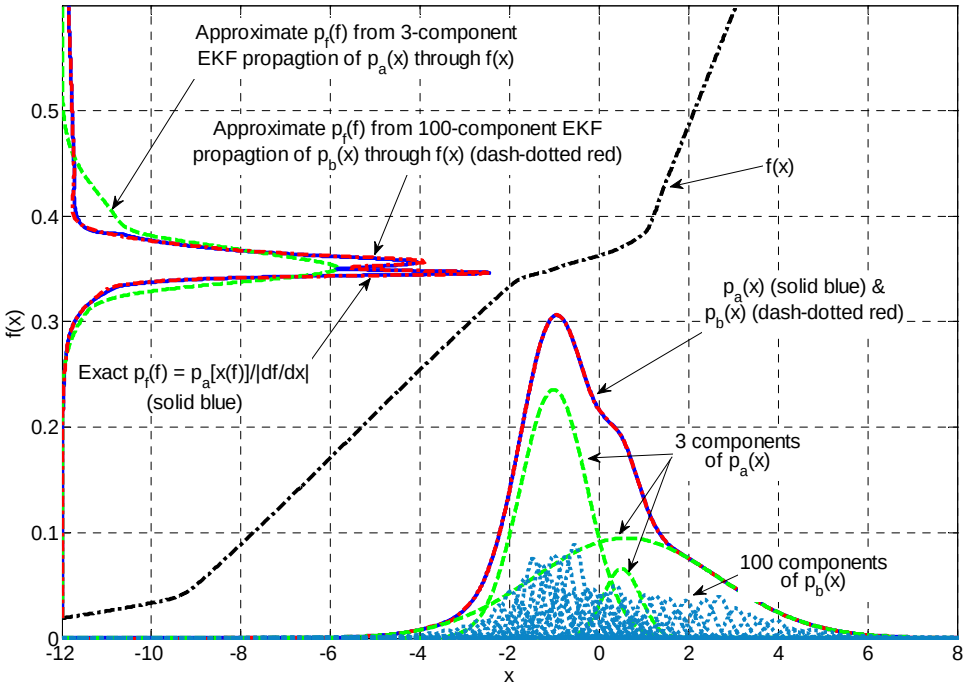


**Fig. 4. A 3-component original Gaussian mixture, a 100-component approximation, and their propagation through a nonlinear function.**

Figure 5 presents three *a posteriori* probability density functions for this example. The example's measurement error covariance is $P_{\nu\nu} = (0.1)^2$. A truth-model simulation generated $\boldsymbol{x}_{true} = -2.0965$ and $\boldsymbol{y} = 0.2996$. The solid blue curve is the true *a posteriori* probability density, the dash-dotted red curve is based on 100-element multiple-model EKF calculations involving the approximate *a priori* distribution $p_b(\boldsymbol{x})$, and the dashed green curve is based on 3-element multiple-model EKF calculations involving the true *a priori* distribution $p_a(\boldsymbol{x})$. The dash-dotted red curve is obviously a much better approximation of the solid blue curve than is the dashed green curve. This improvement further illustrates the advantages for nonlinear Kalman filtering of this paper's technique of re-sampling a Gaussian mixture in order to limit the covariance of its individual elements. In this example, the technique is used to intentionally over-sample the original 3-element Gaussian mixture in order to produce accurate transformations of probability densities through nonlinear functions.

The components of $p_b(\boldsymbol{x})$ act like basis functions in the re-approximation of $p_a(\boldsymbol{x})$. The main idea of this technique is to provide a means of dynamically updating the basis functions for the nonlinear Kalman filter's probability distribution. This update seeks to maintain the accuracy with which the basis functions approximate the underlying distribution. At the same time, it seeks to limit the covariances of the basis functions in order to maintain the accuracy of the approximate EKF or UKF calculations that will be used to dynamically propagate them and to update them when new measurement data become available.

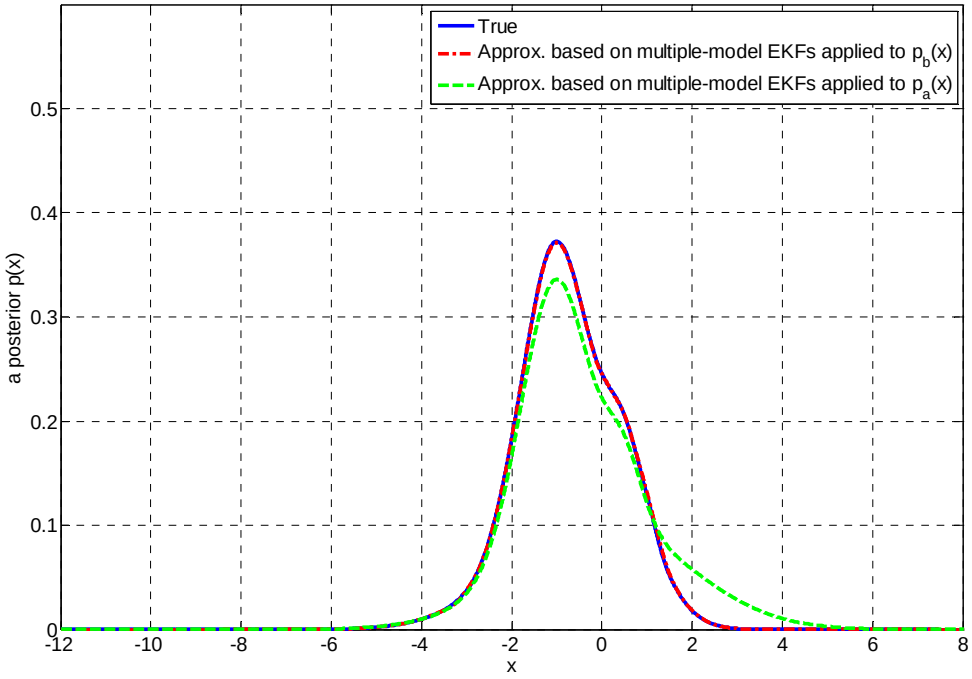


**Fig. 5. True and approximate a posteriori probability distributions after a nonlinear measurement update.**

## X. Summary and Conclusions

A new Gaussian mixture re-approximation/re-sampling algorithm has been developed. It has three goals. First, it seeks to create a new mixture that is a close approximation of the original mixture. Second, it limits the covariances of the elements of its new mixture so that each one will propagate accurately through typical EKF or UKF nonlinear filter calculations, provided that the covariances have been limited to a sufficient degree for a given problem model. The algorithm's third goal is to limit the number of components of the re-sampled mixture. It employs two complementary strategies for achieving this goal. One is to use optimal weight calculations rather than large numbers of components in order to accurately approximate the original distribution in regions of high probability density. The other is to merge components of the original mixture when possible.

The re-sampling algorithm's form represents a natural generalization of particle filtering techniques. Covariance matrices of the new mixture components are determined by solving systems of linear matrix inequalities that set lower bounds on the corresponding information matrices. Mean values of new mixture components are sampled from sub-mixtures that have decreased covariances. These decreased covariances compensate for the fact that the total covariance consists of contributions from the variability of the new means and from the new covariances. The weights of new elements are determined by quadratic programs that optimally fit sub-mixtures of new components to sub-mixtures of old components.

The re-sampling algorithm has been tested on two example problems. The results show that good approximations can be achieved with reasonable numbers of narrowed components of a new Gaussian mixture. The re-sampled mixture is not constrained to preserve the mean or covariance of the original mixture. Nevertheless, the algorithm's concern for accurate approximation of the original probability density function tends to result in better reproduction of the original mean and covariance than is achieved by the same number of particles in a standard PF.

It is reasonable to inflate the number of particles in a comparison PF until the PF processing time equals the amount required by the new algorithm. Even in this case, however, the PF fails to achieve mean and covariance accuracies as good as those of the new algorithm.

Another example calculation demonstrates good EKF/multiple-model propagation and measurement-update results in the presence of significant nonlinearities. The approximate EKF/multiple-model probability density functions closely match the true density functions, as determined numerically, if the approximate EKF calculations are applied to a re-sampled mixture that has sufficiently narrow components.

## References


[1]Julier, S., Uhlmann, J., and Durrant-Whyte, H.F., "A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators," *IEEE Transactions on Automatic Control*, Vol. AC-45, No. 3, 2000, pp. 477-482.

[2]Arulampalam, M.S., Maskell, S., Gordon, N., and Clapp, T., "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking," *IEEE Transactions on Signal Processing*, Vol. 50, No. 2, Feb. 2002, pp. 174-188.

[3]Psiaki, M.L., "Backward-Smoothing Extended Kalman Filter," *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 5, Sept.-Oct. 2005, pp. 885-894.

[4]Wan, E.A., and van der Merwe, R., "The Unscented Kalman Filter," *Kalman Filtering and Neural Networks*, S. Haykin, ed., J. Wiley & Sons, (New York, 2001), pp. 221-280.

[5]Psiaki, M.L., "Estimation Using Quaternion Probability Densities on the Unit Hypersphere", *Journal of the Astronautical Sciences*, Vol. 54, Nos. 3-4, July-Dec. 2006, pp. 415-431.

[6]Sorenson, H.W., and Alspach, D.L., "Recursive Bayesian Estimation Using Gaussian Sums," *Automatica*, Vol. 7, No. 4, 1971, pp. 465-479.

[7]van der Merwe, R., and Wan, E., "Gaussian Mixture Sigma-Point Particle Filters for Sequential Probabilistic Inference in Dynamic State-Space Models," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, (Hong Kong), IEEE, Apr. 2003. Available at http://www.cse.ogi.edu/~rudmerwe/pubs/index.html.

[8]Bar-Shalom, Y., Li, X.-R., and Kirubarajan, T., *Estimation with Applications to Tracking and Navigation*, J. Wiley & Sons, (New York, 2001), pp. 441-443.

[9]Williams, J.L., and Maybeck, P.S., "Cost-Function-Based Gaussian Mixture Reduction for Target Tracking," *Proceedings of the Sixth International Conference on Information Fusion*, Cairns, Queensland, Australia, 2003. Available online at http://ieee-aess.org/isif/sites/default/files/proceedings/fusion03CD/regular/r214.pdf.

[10]Psiaki, M.L., "Global Magnetometer-Based Spacecraft Attitude and Rate Estimation," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 2, March-April 2004, pp. 240-250.

[11]Salmond, D.J., "Mixture Reduction Algorithms for Uncertain Tracking," Technical Report 88004, Farnborough, UK: Royal Aerospace Establishment, January 1988.

[12]Runnalls, A.R., "Kullback-Leibler Approach to Gaussian Mixture Reduction," *IEEE Trans. on Aerospace and Electronic Systems*, Vol. 43, No. 3, July 2007, pp. 989-999.

[13]Chib, S., and Greenberg, E., "Understanding the Metropolis-Hastings Algorithm," *The American Statistician*, Vol. 49, No. 4, Nov. 1995, pp. 327-335.

[14]Gill, P.E., Murray, W., and Wright, M.H., *Practical Optimization*, Academic Press, (New York, 1981), pp. 37-40, 164, 167-180.

[15]Kailath, T., *Linear Systems*, Prentice-Hall, (Englewood Cliffs, N.J., 1980), p. 656.