On fusion of probability density functions using tensor train decomposition

Jiří Ajgl, Ondřej Straka
European Centre of Excellence – New Technologies for Information Society and
Department of Cybernetics, Faculty of Applied Sciences, University of West Bohemia,
Pilsen, Czech Republic.
Email: jiriajgl@kky.zcu.cz, straka30@kky.zcu.cz

***Abstract*—Non-linear filters consider probability density functions in various non-parametric representations. They often suffer from the curse of dimensionality. Computation of weights over a grid of points becomes infeasible even for low dimensions. Filters processing data produced in different sensor nodes provide their own probability densities. Combination of such densities is desired. A favourite paradigm is to construct a fused density as a weighted arithmetic or geometric mean of the individual densities. This paper prospects the fusion for tensor train representation of densities produced by point-mass filters. In this representation, the weights are neither evaluated for a whole grid nor fully stored in the memory of the filters. Aspects of tensor-train-based fusion are discussed, such as computation of auxiliary characteristics and experience with numerical examples.**

***Index Terms*—probability density fusion, point-mass filter, tensor train decomposition**

I. Introduction

Combination of processed data is a traditional problem in diverse disciplines [1], [2]. No matter whether the data originate from sensors, financial statistics or expert judgements, the combination is generally beneficial. Concepts like accuracy, ambiguity and uncertainty are frequently introduced. Many frameworks have been studied in the literature [3], but the most widely-known is the probabilistic "one". The quotes in "one" have been used, since the axiomatic calculus can be used in disparate ways. The fundamental division recognises outcomes of repeated experiments and rationally-supported beliefs.

Fusion of probabilistic beliefs can be founded on axioms [4], [5] like symmetry or zero preservation property, i.e. that the results are independent of the order of the individual beliefs and that the fused belief reports zero probability for events with zero probability attributed by each individual density function. Another approach stems from the information-geometry. Densities are perceived as points in an abstract space and the objective is to minimise some distance criterion. Weighted sum of Kullback–Leibler divergences (with either order of the densities) is a canonical example [6], [7]. Weighted geometric and arithmetic means of the probability densities are the arguments of the minima for the respective orders; theoretical properties of the fusion rules and other aspects have been discussed e.g. in [8]–[10]. Other examples include Wasserstein or Siegel distance [11], [12]. If the densities are not restricted to special parametric families like Gaussian densities, difficulties with density representation emerge.

Densities can be approximated by Gaussian mixtures. Unfortunately, application of the weighted geometric mean of Gaussian mixtures is not a closed operation; although the power of a Gaussian density and the product of two Gaussian densities are proportional to another Gaussian density, the geometric mean does not generate Gaussian mixture. Sigma-point based approximations have been studied in [13]–[15].

Particle filters [16] represent densities by random samples with corresponding weights. The samples are drawn from a sampling density and the weights are proportional to the ratio of the represented density and the sampling density evaluated in the samples. These densities correspond to state trajectories of dynamical systems and are therefore inappropriate for the fusion, which is interested in single-state densities [17], [18]. The key step is the design of the sampling density for the fused samples [19]. Extensions to a semi-parametric representation have been discussed in [20], [21].

Point-mass filters [22], [23] replace random samples by deterministic grids. Since the densities correspond to the state at a single time, they are better suited for the fusion. The problem of fused grid nevertheless persists. Moreover, the filters suffer from the curse of dimensionality, i.e. their computational and storage requirements scale exponentially with the dimension of the state. An alternative to the semi-parametric approach has been proposed recently [24] and is based on tensor train decomposition.

In the area of machine learning, tensors are multidimensional arrays of numerical values [25], [26]. Matrices, i.e. two-dimensional tensors, can be factorised by the SVD decomposition. Truncation of small singular values leads to a low-rank matrix approximation, which diminishes the storage requirements. Several generalisations for higher dimensions are available. The canonical polyadic decomposition has been employed in [27], [28]. Series of small canonical factors can be used to express tensors in the tensor train form [29] or in the tensor ring form [30], [31]. There are many algorithms to construct the decompositions. This paper considers the cross interpolation algorithms for tensor trains [32]–[36].

The goal of the paper is to prospect the area of tensor-train-based fusion. Storage requirements of the point-mass filter are the main motivation.

Section II formulates the tensor-train-based fusion. Sections III–V show the design, experiments and conclusion.

II. Problem Formulation

Section II-A introduces the fusion problem on a general level. Section II-B shows the point-mass representation of densities. Section II-C presents the tensor train form.

A. Weighted Geometric Mean of Densities

Let $\mathbf{x}$ be the state of a system. Dynamical systems are considered, but the time index is omitted here for brevity. Let two filters process their own measurements and produce two conditional densities. The fusion is a heuristic step that ignores the conditioning, treats the densities as two points in an abstract space of densities and produces a fused density that is later declared as conditioned by both sets of measurements. Denote the individual density functions as p and q and the fused density as f. The weighted geometric mean of densities is given by

$$f(\mathbf{x}) \propto \big(p(\mathbf{x})\big)^{\omega} \big(q(\mathbf{x})\big)^{1-\omega}, \tag{1}$$

where the weight ω fulfils $0 \leq \omega \leq 1$ and can be predetermined by the user or tuned according to an optimality criterion [6], [9]. The proportionality symbol $\propto$ denotes that the geometric mean itself is divided by the integral over $\mathbf{x}$, such that $f(\mathbf{x})$ integrates to one.

B. Point-Mass Densities

The point-mass filter approximates the continuous probability density function p by a piece-wise constant density represented by a grid of N points $\mathbf{x}^{(i)}$, their non-overlapping neighbourhoods $\Delta^{(i)}$ and weights $\mathbf{P}^{(i)}$ according to [22] as

$$p(\mathbf{x}) \approx \sum_{i=1}^{N} \mathbf{P}^{(i)} \mathbf{1}(\mathbf{x}; \Delta^{(i)}), \tag{2}$$

where $\mathbf{1}(\mathbf{x}; \Delta^{(i)})$ is the indicator function that is equal to one for $\mathbf{x} \in \Delta^{(i)}$ and zero otherwise. The values of the weights $\mathbf{P}^{(i)}$ are normalised, such that $p(\mathbf{x})$ integrates to one.

Let the state $\mathbf{x}$ be d-dimensional, $\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_d]^{\mathrm{T}}$, and the points $\mathbf{x}^{(i)}$ be given by an orthogonal grid. The weights $\mathbf{P}^{(i)}$ are organised in a d-dimensional array with $N_1, N_2, \ldots, N_d$ elements in the respective dimensions, such that it holds $N = N_1 \cdot N_2 \cdot \ldots \cdot N_d$. Each index i can thus be replaced by a tuple of indices $(i_1, i_2, \ldots, i_d)$ with the corresponding ranges.

C. Tensor Train Representation

The elements $\mathbf{P}(i_1, i_2, \ldots, i_d)$ of a d-dimensional $N_1 \times N_2 \times \ldots \times N_d$ array $\mathbf{P}$ are expressed as products of d matrices [29],

$$\mathbf{P}(i_1, i_2, \ldots, i_d) = \mathbf{P}_1(i_1)\mathbf{P}_2(i_2)\ldots\mathbf{P}_d(i_d), \tag{3}$$

where $\mathbf{P}_j(i_j)$ are $r_{j-1} \times r_j$ matrices with $r_0 = 1$ and $r_d = 1$. The matrices $\mathbf{P}_j(i_j)$ can be organised in three-dimensional $r_{j-1} \times N_j \times r_j$ arrays $\mathbf{P}_j$, which are called cores, whereas the values r_j are called ranks. For high-enough ranks r_j, the tensor $\mathbf{P}$ can be represented exactly. Nevertheless, approximations by tensor trains with low ranks r_j are often adequate. This is analogous to truncating the SVD decomposition of matrices in order to obtain a low-rank approximation and can attenuate the curse of dimensionality. Many algorithms for tensor-train approximation have been implemented in [37].

III. Design of Fusion

Section III-A presents auxiliary computations of moments and marginal densities for an axis-aligned grid. Section III-B discusses the fusion.

A. Computations for an Axis-Aligned Grid

The grid of points $\mathbf{x}^{(i)}$ can be given by $N_1 \times N_2 \times \ldots \times N_d$ arrays $\mathbf{X}^j$ for each element of the state, $j = 1, \ldots, d$. For an economical storage, the orthogonal grid should be axis-aligned; in Matlab, such arrays can be obtained by the `ndgrid` function. The tensor train representation of $\mathbf{X}^j$ assumes the factorisation $\mathbf{X}^j(i_1, i_2, \ldots, i_d) = \mathbf{X}^j_1(i_1)\mathbf{X}^j_2(i_2)\ldots\mathbf{X}^j_d(i_d)$. Each $1 \times N_j \times 1$ array $\mathbf{X}^j_j$ is thus given by the corresponding j-th marginal grid. For $k \neq j$, the $1 \times N_k \times 1$ arrays $\mathbf{X}^j_k$ are arrays of all ones, $\mathbf{X}^j_k = \mathbf{1}^{1 \times N_k \times 1}$.

The mean vector μ associated with the density $p(\mathbf{x})$ can now be approximated. It is assumed that the grid is fine-enough, i.e. that the function $g(\mathbf{x}) = \mathbf{x}$ can be approximated as $g(\mathbf{x}) \approx \mathbf{x}^{(i)}$, if it holds $\mathbf{x} \in \Delta^{(i)}$ for some i. This leads to $\int_{\mathbb{R}} \mathbf{x} p(\mathbf{x})\, d\mathbf{x} \approx \sum_{i=1}^{N} \mathbf{x}^{(i)} \mathbf{P}^{(i)} |\Delta^{(i)}|$, where $|\Delta^{(i)}|$ is the volume of the neighbourhood $\Delta^{(i)}$. For the orthogonal grid, the volumes can be stored in a tensor train $\mathbf{D}$ with ranks r_j equal to one and the cores derived from the grid steps. Auxiliary tensors $\mathbf{M}^j$ can be introduced as the element-wise (Hadamard) products of $\mathbf{X}^j$ and $\mathbf{D}$, $\mathbf{M}^j = \mathbf{X}^j \circ \mathbf{D}$. Since all ranks of these two tensor trains are equal to one, the product can be computed by the element-wise product of the corresponding cores, $\mathbf{M}^j_k = \mathbf{X}^j_k \circ \mathbf{D}_k$. Then, the components μ_j of the mean vector μ can be approximated by the dot product of the tensors $\mathbf{M}^j$ and $\mathbf{P}$, $\mu_j \approx \mathbf{M}^j \cdot \mathbf{P}$. The dot and Hadamard products have been described in [29] and are available in the TT-toolbox [37].

The covariance matrix Σ can be approximated analogously. Auxiliary tensor trains $\mathbf{C}^{jl}$, $j \neq l$, have cores $\mathbf{C}^{jl}_k$ given by $\mathbf{C}^{jl}_k = \mathbf{1}^{1 \times N_k \times 1}$ for $k \neq j$, $k \neq l$ and by the shifted marginal grids otherwise, $\mathbf{C}^{jl}_j(i_j) = \mathbf{X}^j_j(i_j) - \mu_j$, $\mathbf{C}^{jl}_l(i_l) = \mathbf{X}^l_l(i_l) - \mu_l$. Tensor trains $\mathbf{C}^{jj}$ have cores given by $\mathbf{C}^{jj}_k = \mathbf{1}^{1 \times N_k \times 1}$ for $k \neq j$ and by $\mathbf{C}^{jj}_j(i_j) = \big(\mathbf{X}^j_j(i_j) - \mu_j\big)^2$. The Hadamard products with $\mathbf{D}$ and the subsequent dot products with $\mathbf{P}$ lead to the approximation of the components of the covariance matrix Σ.

Marginal densities can be computed by summing over a subset of indices i_k in (3) and multiplying by the lengths of the neighbourhoods in the corresponding marginal grids, which have been stored in the rank-one cores $\mathbf{D}_{i_k}$. Matrices $\mathbf{K}_j$ of sizes $r_{j-1} \times r_j$ are introduced as $\mathbf{K}_j = \sum_{i_j=1}^{N_j} \mathbf{P}_j(i_j)\mathbf{D}_j(i_j)$. These matrices are then used to compute new cores of the tensor trains of the marginal densities. For example, the marginal density of $\mathbf{x}_1$ and $\mathbf{x}_2$ is represented by $N_1 \times N_2$ matrix with the elements $\mathbf{P}(i_1, i_2) = \mathbf{P}_1(i_1)\mathbf{P}_2(i_2)\mathbf{K}_3 \ldots \mathbf{K}_d$. The core $\mathbf{P}_1$ remains the same as in the joint density, whereas the new core $\mathbf{P}_2$ is now composed from the matrix products $\mathbf{P}_2(i_2)\mathbf{K}_3 \ldots \mathbf{K}_d$. Note also that for neighbouring indices, it is possible to use the tensor dot product from the TT-toolbox [37] for chunks of tensor trains.

It should be noted that the same operations appear many times in the above computations. For example, the core $\mathbf{K}_3$ can be reused in the computation of the marginal density of $\mathbf{x}_4$. Similarly, the computations of μ_1, μ_2, Σ_{11}, Σ_{22} and Σ_{12} contain the computations of $\mathbf{K}_3, \ldots, \mathbf{K}_d$ in the products $\mathbf{M}^j \cdot \mathbf{P}$ and $(\mathbf{C}^{jl} \circ \mathbf{D}) \cdot \mathbf{P}$. Furthermore, see that a correctly normalised density requires $\mathbf{P} \cdot \mathbf{D} = 1$, i.e. $\mathbf{K}_1 \ldots \mathbf{K}_d = 1$.

B. Fusion of Tensor Train Densities

The weighted geometric mean of densities is straightforward on the general level (1), but its numerical implementation for point-mass densities becomes painful. The problems are analogous to those known from particle filtering [19].

First, the grid points $\mathbf{x}^{(i)}$ and the neighbourhoods $\Delta^{(i)}$ have to be same for both $p(\mathbf{x})$ and $q(\mathbf{x})$. That means that at least one density has to be extrapolated to another grid. Since the fused density $f(\mathbf{x})$ need not be concentrated in the areas of high concentration of neither of the individual densities, the preferred choice is to anticipate the moments corresponding to $f(\mathbf{x})$, design the common grid and extrapolate both densities before the fused density is evaluated. The moments can be anticipated by the fusion of moment-matched Gaussian densities.

The key idea of the point-mass filters is to use orthogonal grids that are rotated with respect to the coordinate axes and scaled [22]. This will bring a small complication in the computation of the moments and hamper the computation of marginal densities for the tensor trains. Nevertheless, the main problem of orthogonal grids is that they become inefficient in high dimensions; see that the ratio of the volume of a unit ball and the volume of the hypercube circumscribing this ball decreases rapidly to zero. That is, many points at the corners of the grid will correspond to negligible density values. An efficient representation of the arrays $\mathbf{P}$ by tensor trains will not alleviate this fundamental problem much.

For simplicity, this study assumes that the grids are axis-aligned as in [24] and that both densities $p(\mathbf{x})$ and $q(\mathbf{x})$ are already related to a common grid and represented by tensor trains $\mathbf{P}$ and $\mathbf{Q}$. Further, the fusion weight ω is fixed, i.e. no optimisation of ω is needed. The tensor train $\mathbf{F}$ is sought.

Tensor trains are constructed as low-rank approximations. Non-negativity of $\mathbf{P}(i_1, \ldots, i_d)$ is therefore not guaranteed. Since the weighted geometric mean is defined for non-negative numbers, it is proposed to trim the negative values. The cross methods [32]–[36] can be used to construct the cores $\mathbf{F}_j$ of the tensor train $\mathbf{F}$ without creating the full tensor in a memory. Unfortunately, a general ω power with $0 < \omega < 1$ is not a function that could be dealt with cheaply. A computationally demanding construction will be adopted. The values for one set of indices $i_1, \ldots, i_d$ are given by a function

$$\begin{aligned}&\mathbf{G}(i_1, \ldots, i_d) = \\ &= \big(\max\{0, \mathbf{P}(i_1, \ldots, i_d)\}\big)^{\omega} \big(\max\{0, \mathbf{Q}(i_1, \ldots, i_d)\}\big)^{1-\omega}\end{aligned} \tag{4}$$

that is used as an input to a function implemented in the TT-toolbox [37], e.g. `dmrg_cross` (density matrix renormalization group) [32], `amen_cross` (alternating minimal energy) [33]–[35] or `greedy2_cross` (two-site greedy cross interpolation) [36]. The obtained tensor train $\mathbf{G}$ approximates the non-normalised weighted geometric mean. The normalising constant is obtained by the integral over the domain of $\mathbf{x}$, which is computed as the dot product $\mathbf{G} \cdot \mathbf{D}$. The division of $\mathbf{G}$ by this constant can be implemented as dividing one core by the constant. The sought tensor $\mathbf{F}$ is thus given by $\mathbf{F} = \mathbf{G}/(\mathbf{G} \cdot \mathbf{D})$.

The approximative nature of tensor trains emerges also in the case of the fused tensor train $\mathbf{F}$. Although non-negative values are used during the construction (4), the resulting numerical tensor trains $\mathbf{G}$ and $\mathbf{F}$ are approximate, i.e. they are not guaranteed to provide non-negative values. This will be reminded in the next section.

IV. Experiments

Section IV-A introduces a four-dimensional example used in numerical experiments. Section IV-B reports the observations regarding a general level of application of tensor trains in estimation problems. Section IV-C continues with fusion problems. Memory requirements are shown in Section IV-D.

A. Example Settings

Let the state $\mathbf{x}$ be four dimensional, $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4]^{\mathrm{T}}$, and normally distributed, $\pi(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mu, \Sigma)$. Let the mean vector μ and the covariance matrix Σ be given by

$$\mu = \begin{bmatrix} 50 \\ 50 \\ -200 \\ -200 \end{bmatrix}, \qquad \Sigma = \begin{bmatrix} 0.01 & 0 & 1.5 & 0 \\ 0 & 0.01 & 0 & 1.5 \\ 1.5 & 0 & 400 & 0 \\ 0 & 1.5 & 0 & 400 \end{bmatrix}. \tag{5}$$

For a fixed measurement $\mathbf{z}$ and a known position $\mathbf{s}$ of a radar, $\mathbf{s} = [\mathbf{s}_1, \mathbf{s}_2]^{\mathrm{T}}$, let the likelihood function of the state $\mathbf{x}$ be given by $\ell(\mathbf{x}; \mathbf{z}, \mathbf{s}) = \mathcal{N}\big(\mathbf{z}; h(\mathbf{x}, \mathbf{s}), \mathbf{R}\big)$, where the non-linear function $h(\mathbf{x}, \mathbf{s})$ is defined as

$$h(\mathbf{x}, \mathbf{s}) = \begin{bmatrix} \sqrt{(\mathbf{x}_1 - \mathbf{s}_1)^2 + (\mathbf{x}_2 - \mathbf{s}_2)^2} \\ \operatorname{atan2}(\mathbf{x}_2 - \mathbf{s}_2, \mathbf{x}_1 - \mathbf{s}_1) \end{bmatrix} \tag{6}$$

with $\operatorname{atan2}$ being the four-quadrant inverse tangent function. Consider two radars with different positions $\mathbf{s}_p$ and $\mathbf{s}_q$ but the same covariance matrix $\mathbf{R}$ and let their values be given by

$$\mathbf{s}_p = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \mathbf{s}_q = \begin{bmatrix} 40 \\ 0 \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} 0.0025 & 0 \\ 0 & 0.0004 \end{bmatrix}. \tag{7}$$

Ignoring the conditioning by measurements, the densities $p(\mathbf{x})$ and $q(\mathbf{x})$ are given by

$$p(\mathbf{x}) \propto \ell(\mathbf{x}; \mathbf{z}_p, \mathbf{s}_p)\pi(\mathbf{x}), \qquad q(\mathbf{x}) \propto \ell(\mathbf{x}; \mathbf{z}_q, \mathbf{s}_q)\pi(\mathbf{x}), \tag{8}$$

where the following values are used, $\mathbf{z}_p = [70.7, 0.832]^{\mathrm{T}}$, $\mathbf{z}_q = [50.9, 1.37]^{\mathrm{T}}$.

The marginal grids are designed as 4-σ intervals with 81 equidistantly spaced points, i.e. the cores $\mathbf{X}_j^j$ are given by $\mathbf{X}_j^j = \mu_j + \sqrt{\Sigma_{jj}}[-4, -3.9, \ldots, 4]$ (and have size $1 \times 81 \times 1$). The neighbourhoods of the four-dimensional points $\mathbf{x}^{(i)}$ have a constant volume $|\Delta^{(i)}|$, which is given by $(\Sigma_{11}\Sigma_{22}\Sigma_{33}\Sigma_{44})^{\frac{1}{2}}0.1^4$, i.e. $|\Delta^{(i)}| = 0.004$. The cores $\mathbf{D}_j$ have the size $1 \times 81 \times 1$ and contain the values $(\Sigma_{jj})^{\frac{1}{2}}0.1$.

B. Individual Tensor Trains

First, the tensor train $\mathbf{\Pi}$ representing the prior density π is computed. Various functions from the TT-toolbox [37] have been tested. All of them are stochastic, which complicates the evaluation. The `greedy2_cross` method is unreliable in the inspected case (5); many times, the provided marginal densities have been negative, sliced or distorted. The reason why the case (5) is difficult to deal with is that the components $\mathbf{x}_1$ and $\mathbf{x}_2$ are independent, but $\mathbf{x}_1$ and $\mathbf{x}_3$ are dependent. Similarly, the components $\mathbf{x}_4$ and $\mathbf{x}_3$ are independent, but $\mathbf{x}_4$ and $\mathbf{x}_2$ are dependent. Also, the components $\mathbf{x}_1$, $\mathbf{x}_2$ are conditionally independent for given $\mathbf{x}_3$, $\mathbf{x}_4$. Neither the rank r_1 nor the rank r_3 is equal to one. Even for two-dimensional Gaussian densities, the ranks needed to approximate the joint density with a given precision increase rapidly with an increasing absolute value of the correlation coefficient. Higher correlation implies that the Gaussian density is concentrated in a smaller area of the axis-aligned grid. That is, finer grids are needed. The `amen_cross` and `dmrg_cross` methods are slower, but they have performed better, although they are not immune to the high-correlation problem.

Next, tensor trains $\mathbf{L}^p$ and $\mathbf{L}^q$ representing the likelihood functions have been computed. Since the likelihoods do not depend on $\mathbf{x}_3$ and $\mathbf{x}_4$, the ranks r_2 and r_3 are equal to one. A two-dimensional array actually need to be approximated by the likelihood tensor train and hence, the `greedy2_cross` method is appropriate here. According to (8), auxiliary tensors $\mathbf{B}^p$ and $\mathbf{B}^q$ are obtained by Hadamard products, $\mathbf{B}^p = \mathbf{L}^p \circ \mathbf{\Pi}$, $\mathbf{B}^q = \mathbf{L}^q \circ \mathbf{\Pi}$. The matrices $\mathbf{B}_j(i_j)$ forming the cores are given by taking the Kronecker product of the matrices $\mathbf{L}_j(i_j)$ and $\mathbf{\Pi}_j(i_j)$ see [29] and therefore, rounding to the precision of the likelihood tensor trains is applied to keep the ranks low. Finally, the tensors $\mathbf{B}$ are divided by the sums of their elements and by the constant volume of the neighbourhoods $\Delta^{(i)}$, which can be achieved by dividing one core by the divisors. The tensors $\mathbf{P}$ and $\mathbf{Q}$ fulfilling $\mathbf{P} \propto \mathbf{L}^p \circ \mathbf{\Pi}$ and $\mathbf{Q} \propto \mathbf{L}^q \circ \mathbf{\Pi}$ are thus obtained in the tensor train representation. See that all necessary functions (.*|`times`, `round`, `sum`, /|`mrdivide`) are implemented in the TT-toolbox [37].

The tensor train representation is known to be sensitive to the order of the state components. In this example, the order $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_4$ (x-position, y-position, x-velocity, y-velocity) is disadvantageous from the perspective of the prior tensor train $\mathbf{\Pi}$. The order $\mathbf{x}_1$, $\mathbf{x}_3$, $\mathbf{x}_2$, $\mathbf{x}_4$ would be preferred in $\mathbf{\Pi}$, but it would be detrimental in $\mathbf{L}$. The order $\mathbf{x}_3$, $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_4$ would satisfy both $\mathbf{\Pi}$ and $\mathbf{L}$ in this four-dimensional static example, but the addition of dynamics or of two z-components would not admit a convenient ordering.

C. Fused Tensor Trains

The fusion weight ω has been chosen as $\omega = \frac{1}{2}$ for simplicity. The non-normalised tensor train $\mathbf{G}$ has been computed by the `greedy2_cross` method. This was the slowest part of the computations due to the absence of vector-wise access to the elements of the $\mathbf{P}$ and $\mathbf{Q}$ tensors. A more efficient implementation would be needed. The fused tensor train $\mathbf{F}$ has been obtained by normalisation. Thus, the moments and marginal densities can be compared next. For a given mean vector μ and covariance matrix Σ, the n-σ ellipsoids are given by $\mathcal{E}(\mu, \Sigma) = \{\mathbf{x} | (\mathbf{x} - \mu)^{\mathrm{T}} \Sigma^{-1} (\mathbf{x} - \mu) = n^2\}$. Ellipses are two-dimensional ellipsoids.

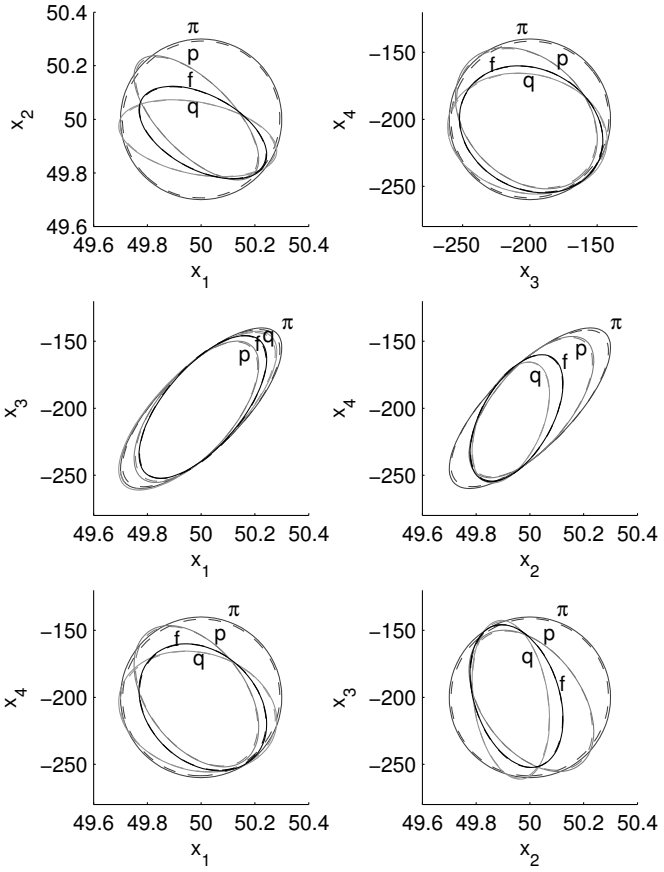

Fig. 1. Comparison of 3-σ ellipses obtained from an EKF filter (solid ellipses) and the tensor train representation (dashed ellipses) for all combinations of the state components. Prior density π (dark grey), local densities p and q (light grey) and the fused density f (black).

Fig. 1 compares the 3-σ ellipses of the π, p, q and f densities. The limits of the axes correspond to the limits of the respective grids. In order to check appropriateness of the tensors, Extended Kalman filter has been used to provide analytical approximations of the moments. It can be observed that the tensor-train-based ellipses (solid) and the analytical ones (dashed) are close together. It has to be confessed that the figures show a carefully selected situation. Other situations would be hard to deal with by any grid-based filter. First, different measured values $\mathbf{z}_p$, $\mathbf{z}_q$ can lead to significantly shifted ellipses. In extreme cases, the theoretical ellipses for p, q and f can leave the grid. Second, higher values of elements Σ_{13} and Σ_{24} in (5) produce ellipsoids with smaller volumes, while the ellipsoids are not axis-aligned. The tensor train decompositions lead to high ranks in such cases and the methods need not converge. Since the algorithms are stochastic, they can stuck in a local minima and stop before the chosen precision is achieved.

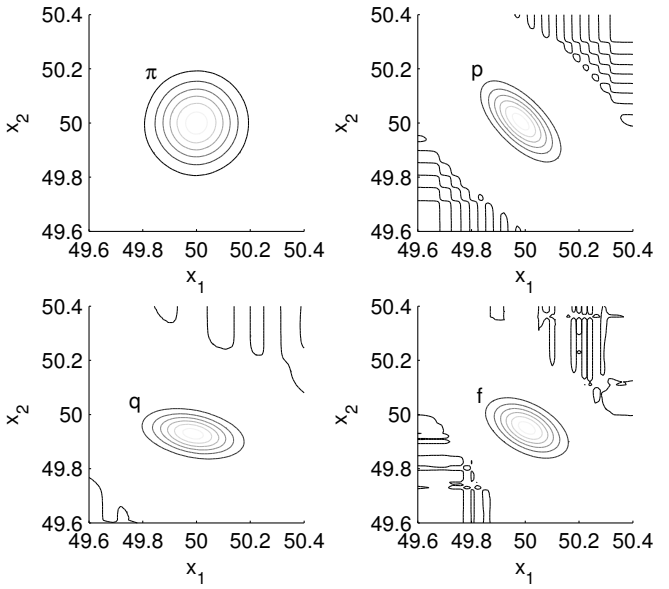

Fig. 2. Contour lines of $\mathbf{x}_1$-$\mathbf{x}_2$ marginal densities corresponding to the prior density π, local densities p, q and the fused density f.

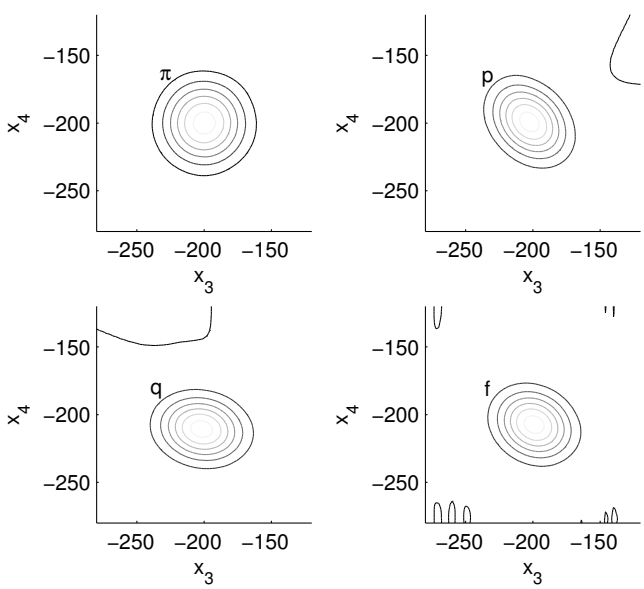

Fig. 3. Contour lines of $\mathbf{x}_3$-$\mathbf{x}_4$ marginal densities corresponding to the prior density π, local densities p, q and the fused density f.

Figs. 2–4 show selected marginal densities of π, p, q and f. Namely, six contour lines are shown for positive levels and if it is present, the contour line for zero is shown as well. It can be seen that in the corners of the grids, the tensor train approximation produces negative values. In the considered run of the stochastic algorithms, the minima for the respective marginal densities were `2.2e-09`, `-1.6e-08`, `-7.1e-09` and `-5.4e-3` in Fig. 2, whereas they were `3.5e-14`, `-1.7e-09`, `-2.0e-08` and `-1.2e-09` in Fig. 3 and `-7.2e-05`, `-8.2e-05`, `-1.1e-4` and `-4.6e-05` in Fig. 4. The minima for the joint densities have not been inspected, since full tensors were nowhere constructed. The negative values can lie far from the mean vector, see Fig. 3, or relatively close to it, see Fig. 4. Since the fused density f approximates the geometric mean of p and q, the contour lines for the zero level of f can be very twisted.

D. Memory Requirements

The full 4-dimensional arrays with $N_j = 81$ points in each dimension would occupy 43 046 721 data cells. In the experiment shown in Figs. 1–4, the cross algorithms for $\mathbf{\Pi}$, $\mathbf{L}^p$ and $\mathbf{L}^q$ stopped at the ranks shown in Table I. The `amen_cross` used to compute $\mathbf{\Pi}$ sometimes stopped prematurely in other runs, e.g. with ranks $\{1, 7, 8, 7, 1\}$ and visibly poor approximations of the density. For $\Sigma_{13} = \Sigma_{24} = 1.5$, the `greedy2_cross` failed repeatedly, even after hundreds of sweeps. Another critical point is the computation of the non-normalised tensor trains $\mathbf{B}^p$, $\mathbf{B}^q$, which require much more memory than the others. The rounding with the precision parameter `1e-8` leads to $\mathbf{P}$ and $\mathbf{Q}$ with memory requirement comparable to that of $\mathbf{\Pi}$. The toolbox implementation of the `greedy2_cross` used for the fusion is slow; the computation was terminated after 20 sweeps, i.e. at the ranks 21.

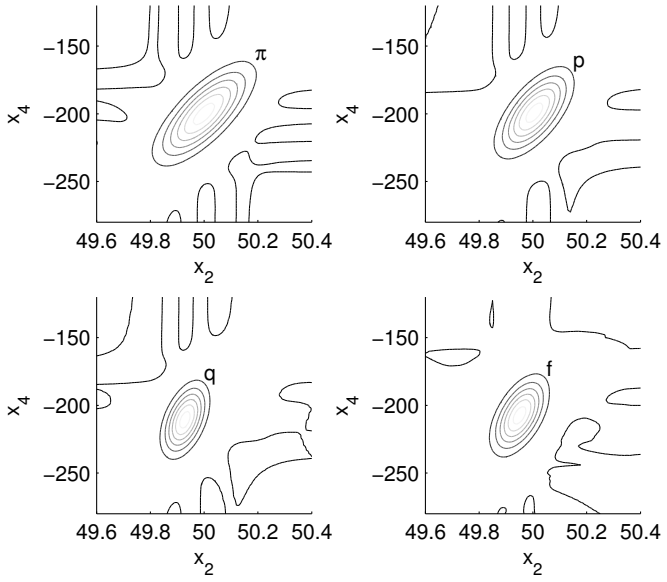

Fig. 4. Contour lines of $\mathbf{x}_2$-$\mathbf{x}_4$ marginal densities corresponding to the prior density π, local densities p, q and the fused density f.

TABLE I
RANKS OF CORES IN THE CONSIDERED RUN AND THE NUMBER OF DATA CELLS IN THE MEMORY. FULL TENSORS OCCUPY 43 046 721 DATA CELLS.

tensor train	r_0	r_1	r_2	r_3	r_4	memory
$\mathbf{\Pi}$	1	16	26	16	1	69 984
$\mathbf{L}^p$	1	28	1	1	1	4 698
$\mathbf{L}^q$	1	14	1	1	1	2 430
$\mathbf{B}^p$	1	448	26	16	1	1 014 768
$\mathbf{B}^q$	1	224	26	16	1	524 880
$\mathbf{P}$	1	22	26	14	1	78 732
$\mathbf{Q}$	1	18	26	14	1	69 984
$\mathbf{F}$	1	21	21	21	1	74 844

V. Summary

Tensor decomposition has been applied in point-mass filters to approximate probability densities over orthogonal grids of points. Fusion of two densities by a weighted geometric mean has been considered. Basic computations have been discussed first, the fusion framework has been designed next and experiments have been conducted last. Numerical aspects have been pointed out. Namely, the performance of the cross methods for a four dimensional system has been evaluated with respect to convergence, precision and memory requirements. It can be concluded that the curse of dimensionality can be alleviated by the tensor train representation, but it cannot be broken in estimation problems.

Acknowledgment

This work was supported by the Czech Science Foundation, project no. P103/22-11101S.

References

[1] R. T. Clemen, "Combining forecasts: A review and annotated bibliography," *International Journal of Forecasting*, vol. 5, no. 4, pp. 559–583, 1989.

[2] K. F. Wallis, "Combining forecasts – forty years later," *Applied Financial Economics*, vol. 21, no. 1–2, pp. 33–41, 2011.

[3] D. Dubois, W. Liu, J. Ma, and H. Prade, "The basic principles of uncertain information fusion. an organised review of merging rules in different representation frameworks," *Information Fusion*, vol. 32, no. Part A, pp. 12–39, November 2016.

[4] C. Genest and J. V. Zidek, "Combining probability distributions: A critique and an annotated bibliography," *Statistical Science*, vol. 1, no. 1, pp. 114–148, 1986.

[5] G. Koliander, Y. El-Laham, P. M. Djurić, and F. Hlawatsch, "Fusion of probability density functions," *Proceedings of the IEEE*, vol. 110, no. 4, pp. 404–453, April 2022.

[6] R. Kulhavý and F. J. Kraus, "On duality of regularized exponential and linear forgetting," *Automatica*, vol. 32, no. 10, pp. 1403–1415, October 1996.

[7] J. Dokoupil and P. Václavek, "Design of variable exponential forgetting for estimation of the statistics of the normal distribution," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Las Vegas, Nevada, USA, December 2016, pp. 1179–1184.

[8] J. Ajgl and M. Šimandl, "On conservativeness of posterior density fusion," in *Proceedings of the 16th International Conference on Information Fusion*, Istanbul, Turkey, July 2013.

[9] ——, "Design of a robust fusion of probability densities," in *2015 American Control Conference*, Chicago, Illinois, USA, July 2015.

[10] J. Ajgl and O. Straka, "Approximate fusion of probability density functions using Gaussian copulas," in *Proceedings of the 26th International Conference on Information Fusion*, Charleston, South Carolina, USA, June 2023.

[11] A. N. Bishop, "Information fusion via the Wasserstein barycenter in the space of probability measures: Direct fusion of empirical measures and Gaussian fusion with unknown correlation," in *Proceedings of the 17th International Conference on Information Fusion*, Salamanca, Spain, July 2014.

[12] M. Tang, Y. Rong, J. Zhou, and X. R. Li, "Information geometric approach to multisensor estimation fusion," *IEEE Transactions on Signal Processing*, vol. 67, no. 2, pp. 279–292, January 2019.

[13] M. Günay, U. Orguner, and M. Demirekler, "Approximate Chernoff fusion of Gaussian mixtures using sigma-points," in *Proceedings of the 17th International Conference on Information Fusion*, Salamanca, Spain, July 2014.

[14] J. Ajgl, M. Šimandl, and J. Duník, "Approximation of powers of Gaussian mixtures," in *Proceedings of the 18th International Conference on Information Fusion*, Washington, DC, USA, July 2015.

[15] M. Günay, U. Orguner, and M. Demirekler, "Chernoff fusion of Gaussian mixtures for distributed maneuvering target tracking," in *Proceedings of the 18th International Conference on Information Fusion*, Washington, DC, USA, July 2015.

[16] S. Godsill, "Particle filtering: The first 25 years and beyond," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, May 2019, pp. 7760–7764.

[17] M. Klaas, N. de Freitas, and A. Doucet, "Toward practical N^2 Monte Carlo: the marginal particle filter," in *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, Arlington, Virginia, USA, 2005, pp. 308–315.

[18] O. Straka and J. Duník, "Efficient implementation of marginal particle filter by functional density decomposition," in *25th International Conference on Information Fusion*, Linköping, Sweden, July 2022.

[19] J. Ajgl and M. Šimandl, "Particle based probability density fusion with differential Shannon entropy criterion," in *Proceedings of the 14th International Conference on Information Fusion*, Chicago, Illinois, USA, July 2011.

[20] F. Lindsten, T. B. Schön, and L. Svensson, "A non-degenerate Rao-Blackwellised particle filter for estimating static parameters in dynamical models," in *IFAC Proceedings Volumes, 16th IFAC Symposium on System Identification*, vol. 45, no. 16, Brussels, Belgium, July 2012, pp. 1149–1154.

[21] J. Ajgl and M. Šimandl, "Marginal marginalised particle filter," in *2013 American Control Conference*, Washington, DC, USA, June 2013.

[22] M. Šimandl, J. Královec, and T. Söderström, "Advanced point-mass method for nonlinear state estimation," *Automatica*, vol. 42, no. 7, pp. 1135–1145, July 2006.

[23] J. Duník, M. Soták, M. Veselý, O. Straka, and W. Hawkinson, "Design of Rao-Blackwellized point-mass filter with application in terrain aided navigation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 1, pp. 251–272, February 2019.

[24] J. Matoušek, M. Brandner, J. Duník, and I. Punčochář, "Tensor train discrete grid-based filters: Breaking the curse of dimensionality," in *20th IFAC Symposium on System Identification*, Boston, Massachusetts, USA, July 2024.

[25] L. Grasedyck, D. Kressner, and C. Tobler, "A literature survey of low-rank tensor approximation techniques," *GAMM-Mitteilungen*, vol. 36, no. 1, pp. 53–78, August 2013.

[26] S. Rabanser, O. Shchur, and S. Günnemann, "Introduction to tensor decompositions and their applications in machine learning," *ArXiv*, no. abs/1711.10781, 2017.

[27] Y. Sun and M. Kumar, "Nonlinear Bayesian filtering based on Fokker-Planck equation and tensor decomposition," in *18th International Conference on Information Fusion*, Washington, DC, USA, July 2015.

[28] F. Govaers, B. Demissie, A. Khan, M. Ulmke, and W. Koch, "Tensor decomposition-based multitarget tracking in cluttered environments," *Journal of advances in information fusion*, vol. 14, no. 1, pp. 86–97, June 2019.

[29] I. Oseledets, "Tensor-train decomposition," *SIAM Journal of Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.

[30] Q. Zhao, G. Zhou, S. Xie, L. Zhang, and A. Cichocki, "Tensor ring decomposition," *ArXiv*, no. abs/1606.05535, June 2016.

[31] O. Mickelin and S. Karaman, "On algorithms for and computing with the tensor ring decomposition," *Numerical Linear Algebra with Applications*, vol. 27, no. 3, p. e2289, May 2020.

[32] D. Savostyanov and I. Oseledets, "Fast adaptive interpolation of multi-dimensional arrays in tensor train format," in *The 2011 International Workshop on Multidimensional (nD) Systems*, Poitiers, France, September 2011.

[33] S. V. Dolgov and D. V. Savostyanov, "Alternating minimal energy methods for linear systems in higher dimensions," *SIAM Journal on Scientific Computing*, vol. 36, no. 5, pp. A2248–A2271, 2014.

[34] ——, "Alternating minimal energy methods for linear systems in higher dimensions. part i: Spd systems," *ArXiv*, no. abs/1301.6068, January 2014.

[35] ——, "Alternating minimal energy methods for linear systems in higher dimensions. part ii: Faster algorithm and application to nonsymmetric systems," *ArXiv*, no. abs/1304.1222, April 2014.

[36] D. V. Savostyanov, "Quasioptimality of maximum-volume cross interpolation of tensors," *Linear Algebra and its Applications*, vol. 458, pp. 217–244, 2014.

[37] I. V. Oseledets. (2024) TT-Toolbox. [Online]. Available: https://github.com/oseledets/TT-Toolbox