Interpolation of Probability Densities on a Grid in Tensor Decomposition Representation

Daniel Frisch and **Uwe D. Hanebeck**
Intelligent Sensor-Actuator-Systems Laboratory (ISAS)
Institute for Anthropomatics and Robotics
Karlsruhe Institute of Technology (KIT), Germany
{daniel.frisch, uwe.hanebeck}@{kit.edu, ieee.org}

***Abstract*—We propose a new method to interpolate probability densities that are represented on a regular or on a rectilinear grid through canonical polyadic decomposition (CPD). We introduce a finer grid and compute the respective density values, or more precisely, the loading vectors of the tensor decomposition on the high-resolution grid, such that i) the density values at the pre-existing grid locations remain unchanged, i.e., we perform pure interpolation, and ii) no unnecessary information is introduced. The latter is achieved by minimizing the Fisher Information Number (FIN) of the interpolated density. The FIN is a roughness measure or regularizer for densities. Internally, we use a square root density representation to be able to compute the FIN in CPD format and to enforce nonnegativity of the density values. Computationally, the proposed method yields an optimization problem of a smooth cost function with one normalization constraint that can be solved, e.g., with interior-point algorithms, for the desired high-resolution loading matrix entries. We demonstrate the performance of the proposed method through visual examples in 1D and 2D.**

I. Introduction

A. Context and Motivation

Low-rank tensor decompositions, such as the canonical decomposition (CANDECOMP) or parallel factor decomposition (PARAFAC), are a powerful tool to represent high-dimensional data in a compact way, thus circumventing the curse of dimensionality [1], [2], [3]. They have been successfully applied in various fields, such as signal processing and machine learning [4], Bayesian filtering, e.g., for multi-target tracking [5], and prediction with stochastic dynamical systems [6], [7], also on nonlinear manifolds [8]. In these applications, it is often desirable to increase the resolution of the underlying grid, e.g., to consolidate the discretization points in the area that is currently of interest. Especially in the context of Bayesian state estimation and probabilistic inference, the relevant support of the density changes over time [9].

B. Density Estimation

The density interpolation problem could also be framed as a density estimation problem. For density estimation, one usually has unweighted data points that are unevenly distributed; however, a density can also be described via unevenly distributed weighted samples [10] or with uniformly distributed but weighted samples. Here, the density values at the pre-existing grid locations represent uniformly distributed but weighted data points, and the density values at the new grid locations are the parameters to be estimated. This could be solved with Gaussian mixture estimation from weighted samples [11], [12], [13], which would however not generalize well to higher dimensions, as there would be too many grid points to iterate over in the expectation maximization procedure — imagine 10 dimensions with 100 discretization points each, yielding 100 quintillion grid points. Instead, tensorized variants [14] of the classical non-parametric kernel density estimator (KDE) approaches [15] could be applied, if the kernel is in canonical polyadic decomposition (CPD) format, e.g., a Gaussian kernel with diagonal covariance matrix. The problem with this is the limitation to and choice of one single bandwidth parameter, which is crucial for the performance of the method and is a long-standing problem in the field of density estimation [16]. Therefore, in the following, we view the problem as directly interpolating a coarsely defined probability density function.

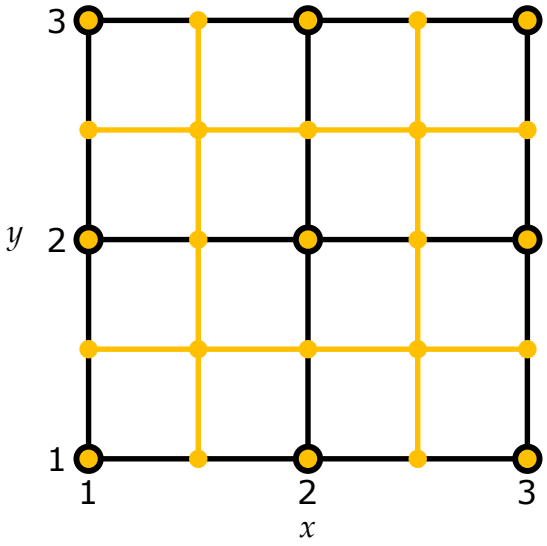
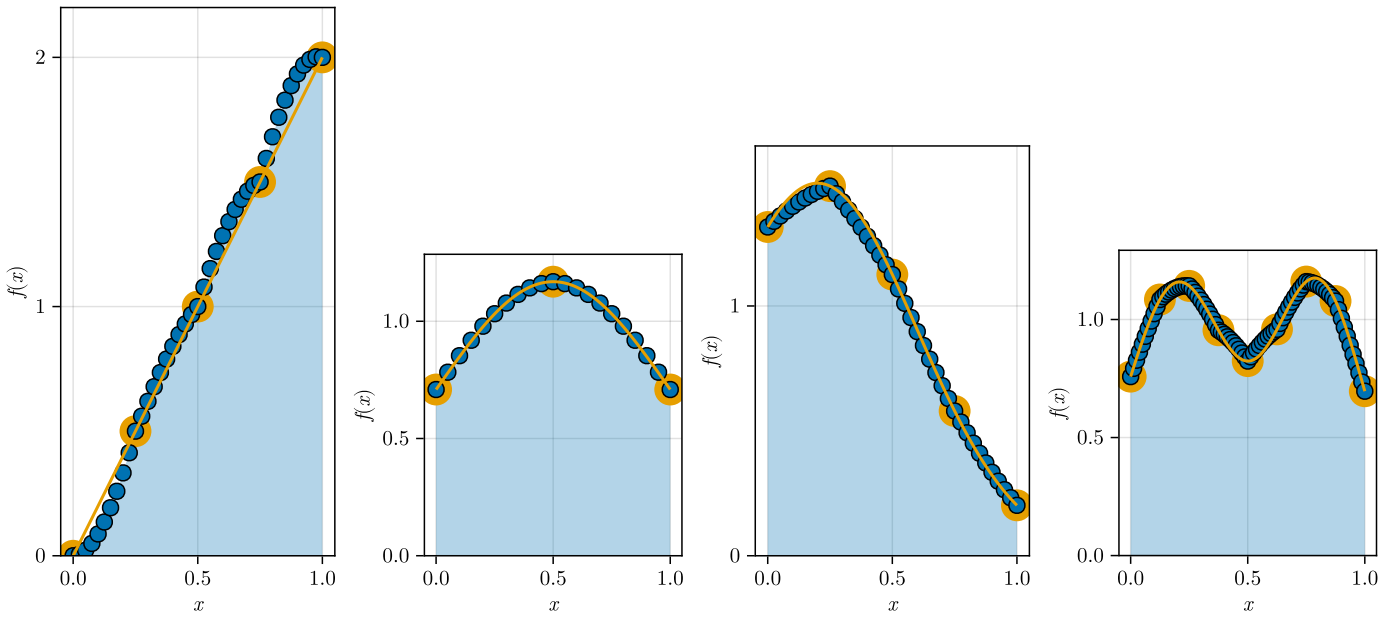


Fig. 1: Grid Refinement: Original grid points $\underline{x}^{\mathrm{g}}[\underline{n}]$ (black) and refined grid points $\underline{x}^{\widetilde{\mathrm{g}}}[\underline{\widetilde{n}}]$ (orange), with $N_1 = N_2 = 3$ and $\widetilde{N}_1 = \widetilde{N}_2 = 5$. The grid locations are given by $\underline{x}_1^{\mathrm{g}} = \underline{x}_2^{\mathrm{g}} = [1, 2, 3]^\top$ and $\underline{x}_1^{\widetilde{\mathrm{g}}} = \underline{x}_2^{\widetilde{\mathrm{g}}} = [1, 1.5, 2, 2.5, 3]^\top$.



(a) Rising, 100 ms (b) Gauss, $\sigma = 0.5$, 140 ms (c) Gauss, $\sigma = 0.4$, 80 ms (d) GM, 100 ms

Fig. 2: 1D examples of the proposed density interpolation method. There are $N_1 \in \{3, 5, 9\}$ input grid points $x_1^g[n_1] \in [0, 1]$ with given and fixed density values (yellow dots) obtained from linear (a), truncated Gaussian (b,c), and truncated Gaussian mixture (d) densities (yellow lines). For $\widetilde{N}_1 = (N_1 - 1) \cdot 10 + 1$ output grid points, we computed the interpolated density (blue dots). Note that in 1D, the tensor decomposition is not needed, as the density can be directly represented as a vector of values on the grid points, therefore we have $L = 1$. Computation times are given in the captions.

C. Fisher Information Number (FIN)

Whenever some data points are interpolated, to make it well-defined and unique, one has to define some constraints, like continuity of the function or its derivatives, some cost function that should be minimized, or a smooth kernel function to convolve the data with. For interpolating densities, we probably do not want the typical nearest-neighbor, piecewise linear, or spline interpolation [17], as they do not take into account the properties of densities, such as nonnegativity and information content. Instead, we want to introduce as little information as possible. Therefore, the information content would be an ideal cost function. Here, we propose to minimize the Fisher information Number (FIN) of the interpolated density. It is a roughness measure and a regularizer for densities [18] and has been used, e.g., for approximating densities with Gaussian mixtures [19], [20] and for spline-based density estimation [21]. See Fig. 2 for a trivial 1D example, where few points (yellow dots) are interpolated to many points (blue dots).

II. Notation

A. Rectilinear Grid

Rectilinear grids are axis-aligned and with arbitrary spacing along the individual dimensions [22]. We begin defining the grid point locations $\underline{x}^{\mathrm{g}} \colon \times_{s=1}^{S} \{1, 2, \ldots, N_s\} \to \mathbb{R}^S$

$$\underline{x}^{\mathrm{g}}[\underline{n}] = \underline{x}^{\mathrm{g}}[n_1, \ldots, n_S] = \begin{bmatrix} x_1^{\mathrm{g}}[n_1] \\ \vdots \\ x_S^{\mathrm{g}}[n_S] \end{bmatrix} \quad , \tag{1}$$

where $\underline{n}$ is a multi-index containing one index n_s, where $n_s = 1, \ldots, N_s$, for each dimension $s = 1, \ldots, S$, with N_s being the number of grid points along each dimension. Thus, the overall number of grid points is

$$N_{\mathrm{grid}} = \prod_{s=1}^{S} N_s \quad . \tag{2}$$

All grid point locations are determined by the vectors

$$\underline{x}_s^{\mathrm{g}} = \left[x_s^{\mathrm{g}}[1], \ldots, x_s^{\mathrm{g}}[N_s]\right]^\top \in \mathbb{R}^{N_s} \tag{3}$$

containing the grid point locations along each dimension s.

B. Probability Density

We represent a probability density $f(\underline{x})$ as density values on the grid points $f[\underline{n}]$. Furthermore, we choose a square root representation $r[\underline{n}]$, where

$$f[\underline{n}] = r^2[\underline{n}]. \tag{4}$$

This helps enforce the nonnegativity constraint and facilitates the tensorized computation of the FIN, as shown later. To circumvent the curse of dimensionality, we employ a rank-L tensor decomposition

$$r[\underline{n}] = \sum_{l=1}^{L} \prod_{s=1}^{S} \mathbf{\Phi}_s[n_s, l] \quad , \tag{5}$$

where $\mathbf{\Phi}_s \in \mathbb{R}^{N_s \times L}$ is the loading matrix for dimension s, and its columns are the loading vectors. The density $f[\underline{n}]$ can then be represented in terms of the loading matrices via

$$f[\underline{n}] = \left(\sum_{l=1}^{L} \prod_{s=1}^{S} \mathbf{\Phi}_s[n_s, l] \right)^2 \tag{6}$$

$$= \sum_{l_1=1}^{L} \sum_{l_2=1}^{L} \prod_{s=1}^{S} \mathbf{\Phi}_s[n_s, l_1] \cdot \mathbf{\Phi}_s[n_s, l_2] \ , \tag{7}$$

thus $f[\underline{n}]$ is a rank-L^2 tensor.

III. Fisher Information Number (FIN)

The so-called linear score or informant is defined as [18, Eq. 13]

$$s(\underline{x}) = \frac{\partial}{\partial \underline{x}} \log(f(\underline{x})) = \frac{\nabla f(\underline{x})}{f(\underline{x})} \ . \tag{8}$$

The FIN is defined as [19, Eq. 10], [18, Eq. 1],

$$I_F(f) = \int_{\{\underline{x} \in \mathbb{R}^S \,:\, f(\underline{x}) > 0\}} \frac{\|\nabla f(\underline{x})\|_2^2}{f(\underline{x})} \, \mathrm{d}\underline{x} \ . \tag{9}$$

It can be used as a roughness measure for densities [19]. Using the square root representation $r(\underline{x}) = \sqrt{f(\underline{x})}$, according to [19, Eq. 19], this becomes

$$I_F(f) = I_F^R(r) \tag{10}$$

$$= 4 \int_{\mathbb{R}^S} \|\nabla r(\underline{x})\|_2^2 \, \mathrm{d}\underline{x} \tag{11}$$

$$= 4 \int_{\mathbb{R}^S} \sum_{s=1}^{S} \left(\frac{\partial r(\underline{x})}{\partial x_s} \right)^2 \mathrm{d}\underline{x} \tag{12}$$

$$= 4 \sum_{s=1}^{S} \int_{\mathbb{R}^S} \left(\frac{\partial r(\underline{x})}{\partial x_s} \right)^2 \mathrm{d}\underline{x} \ . \tag{13}$$

With our discretized density (5) that is defined on the regular grid (1), and using forward differences, we obtain

$$I_F^R(r) = 4 \sum_{s=1}^{S} \sum_{\underline{n}=\underline{1}}^{\underline{N}-\underline{e}_s} \left(\frac{r[\underline{n} + \underline{e}_s] - r[\underline{n}]}{x_s^{\mathrm{g}}[n_s + 1] - x_s^{\mathrm{g}}[n_s]} \right)^2 V[\underline{n}] \ . \tag{14}$$

Thereby, $\underline{e}_s$ is a unit vector pointing in the direction of dimension s and $\sum_{\underline{n}=\underline{1}}^{\underline{N}}$ is an S-fold multi-sum that iterates over all N_{grid} grid entries (with exponential complexity in S). $V[\underline{n}]$ is the volume element associated with the root density value $r[\underline{n}]$ at location $\underline{x}^{\mathrm{g}}[\underline{n}]$. The precise location of this volume element depends on the type of Riemann sum we use for integration, e.g., for the left Riemann sum simply

$$V[\underline{n}] = \prod_{s=1}^{S} (x_s^{\mathrm{g}}[n_s + 1] - x_s^{\mathrm{g}}[n_s]) \ . \tag{15}$$

We must now avoid the multi-sum in order to circumvent the curse of dimensionality and be able to apply the method in arbitrary dimensions. The derivation is given in Appendix A, and the result is presented below.

For an equidistant grid with spacing Δx_s^{g} along each dimension s, and collecting factors that do not depend on the loading matrices to constant c, the tensorized form of the FIN can be written as

$$I_F^R(\mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_S) = c \cdot \sum_{l_1=1}^{L} \sum_{l_2=1}^{L} \sum_{s=1}^{S} \frac{1}{(\Delta x_s^{\mathrm{g}})^2} \, \text{↲} \tag{16}$$

$$\left(\sum_{n_s=1}^{N_s-1} (\mathbf{\Phi}_s[n_s + 1, l_1] - \mathbf{\Phi}_s[n_s, l_1]) \, \text{↲} \right.$$

$$\left. \cdot \, (\mathbf{\Phi}_s[n_s + 1, l_2] - \mathbf{\Phi}_s[n_s, l_2]) \right) \, \text{↲}$$

$$\cdot \prod_{\substack{s'=1 \\ s' \neq s}}^{S} \left(\sum_{n_{s'}=1}^{N_{s'}} \mathbf{\Phi}_{s'}[n_{s'}, l_1] \cdot \mathbf{\Phi}_{s'}[n_{s'}, l_2] \right).$$

In this form, the complexity is no longer exponential but merely quadratic in S.

The gradient of this, which is helpful for optimization, is stated in Appendix B.

IV. High-Resolution Grid Representation

Now we want to increase the resolution of the existing grid (1). We do this by adding intermediate grid points while keeping the old grid points in place as part of the new grid. This does not limit the generality of the method, since unwanted grid points can simply be removed in the end.

We define a finer regular grid

$$\underline{x}^{\widetilde{\mathrm{g}}}[\underline{n}] = \underline{x}^{\widetilde{\mathrm{g}}}[\widetilde{n}_1, \ldots, \widetilde{n}_S] = \begin{bmatrix} x_1^{\widetilde{\mathrm{g}}}[\widetilde{n}_1] \\ \vdots \\ x_S^{\widetilde{\mathrm{g}}}[\widetilde{n}_S] \end{bmatrix} \ , \tag{17}$$

with new indices $\widetilde{n}_s = 1, \ldots, \widetilde{N}_s$, where $\widetilde{N}_s$ is the new number of grid points along dimension s, with $\widetilde{N}_s \geq N_s$. As mentioned before, without loss of generality we assume that the new grid contains all the old grid points, i.e., for each old grid point index n_s there exists a new grid point index $\widetilde{n}_s$ such that

$$x_s^{\mathrm{g}}[n_s] = x_s^{\widetilde{\mathrm{g}}}[\widetilde{n}_s] \ . \tag{18}$$

Correspondingly, we want to expand the existing loading matrices $\mathbf{\Phi}_s \in \mathbb{R}^{N_s \times L_{\mathrm{in}}}$ with entries $\mathbf{\Phi}_s[n_s, l]$ to new loading matrices $\widetilde{\mathbf{\Phi}}_s \in \mathbb{R}^{\widetilde{N}_s \times L_{\mathrm{out}}}$ with entries $\widetilde{\mathbf{\Phi}}_s[\widetilde{n}_s, l]$ and $L_{\mathrm{out}} \geq L_{\mathrm{in}}$. Again, some of the entries in $\widetilde{\mathbf{\Phi}}_s$ correspond to the entries in $\mathbf{\Phi}_s$, and each entry in $\mathbf{\Phi}_s$ corresponds to one entry in $\widetilde{\mathbf{\Phi}}_s$. The new, interpolated density values on the new grid are then given by

$$\widetilde{f}[\widetilde{\underline{n}}] = \left(\sum_{l=1}^{L_{\mathrm{out}}} \prod_{s=1}^{S} \widetilde{\mathbf{\Phi}}_s[\widetilde{n}_s, l] \right)^2 \ . \tag{19}$$

V. Key Idea

Our goal is to compute the entries of the new loading matrices $\widetilde{\mathbf{\Phi}}_s$ such that (G1) the density values $f[\underline{n}]$ at the pre-existing grid locations remain unchanged and that (G2) we add the least possible amount of information to the density in that process.

We can achieve (G1) exactly. Note that the old density values depend only on the old loading vectors, even in the finer grid setup. Take, for example, the center grid point in Fig. 1 with $n_1 = n_2 = 2$ and $\widetilde{n}_1 = \widetilde{n}_2 = 3$. Its density value is

$$f[2,2] = \sum_{l=1}^{L_{\text{in}}} \mathbf{\Phi}_1[2,l] \cdot \mathbf{\Phi}_2[2,l] \ , \tag{20}$$

$$\widetilde{f}[3,3] = \sum_{l=1}^{L_{\text{out}}} \widetilde{\mathbf{\Phi}}_1[3,l] \cdot \widetilde{\mathbf{\Phi}}_2[3,l] \ . \tag{21}$$

One way of ensuring $f[2,2] = \widetilde{f}[3,3]$ is having $\mathbf{\Phi}_1[2,l] = \widetilde{\mathbf{\Phi}}_1[3,l]$ and $\mathbf{\Phi}_2[2,l] = \widetilde{\mathbf{\Phi}}_2[3,l]$ for all $l = 1,\dots,L_{\text{in}}$ and $\widetilde{\mathbf{\Phi}}_1[3,l] = \widetilde{\mathbf{\Phi}}_2[3,l] = 0$ for $l > L_{\text{in}}$. This reasoning holds for all other old grid points as well. Thus, we can ensure (G1) simply by fixing those new loading matrix entries that correspond to the old loading matrix entries (same grid point locations) to the old values, and if $L_{\text{out}} > L_{\text{in}}$, the remaining entries to zero. In that case, the density remains unchanged at the old grid locations, and we perform pure interpolation. This also removes many constraints from the optimization, since we then only need to optimize over the new loading matrix entries, simplifying the problem.

To achieve (G2), we propose to minimize the FIN (41) (see Appendix A) of the resulting density on the finer grid.

VI. Normalization Constant

We assume the input density values originate from an underlying normalized density and we want to obtain an interpolated density that is normalized as well, approximating the underlying original density.

We can compute the normalization constant of $\widetilde{f}[\underline{n}] = \widetilde{r}^2[\underline{n}]$ in a tensorized manner

$$Z = \int_{\mathbb{R}^S} \widetilde{f}(\underline{x}) \, \mathrm{d}\underline{x} \tag{22}$$

$$\approx \sum_{l_1=1}^{L} \sum_{l_2=1}^{L} \prod_{s=1}^{S} \left(\sum_{\widetilde{n}_s=1}^{\widetilde{N}_s-1} \widetilde{\mathbf{\Phi}}_s[\widetilde{n}_s, l_1] \cdot \widetilde{\mathbf{\Phi}}_s[\widetilde{n}_s, l_2]_{\,\lrcorner\!\lrcorner} \right. \tag{23}$$

$$\left. \cdot \left(x_s^{\widetilde{\mathrm{g}}}[\widetilde{n}_s + 1] - x_s^{\widetilde{\mathrm{g}}}[\widetilde{n}_s] \right) \right) \ ,$$

which for equidistant grids simplifies to

$$Z = b \cdot \sum_{l_1=1}^{L} \sum_{l_2=1}^{L} \prod_{s=1}^{S} \left(\sum_{\widetilde{n}_s=1}^{\widetilde{N}_s-1} \widetilde{\mathbf{\Phi}}_s[\widetilde{n}_s, l_1] \cdot \widetilde{\mathbf{\Phi}}_s[\widetilde{n}_s, l_2] \right) \ , \tag{24}$$

$$b = \prod_{s=1}^{S} \Delta x_s^{\widetilde{\mathrm{g}}} \ . \tag{25}$$

The gradient of the normalization constant with respect to any entry of a loading matrix is given by

$$\frac{\partial Z}{\partial \widetilde{\mathbf{\Phi}}_s[n_s, l]} = 2b \cdot \sum_{l_1=1}^{L} \widetilde{\mathbf{\Phi}}_s[n_s, l_1] \cdot {}_{\lrcorner\!\lrcorner} \tag{26}$$

$$\cdot \prod_{\substack{s'=1 \\ s' \neq s}}^{S} \left(\sum_{n_{s'}=1}^{N_{s'}} \widetilde{\mathbf{\Phi}}_{s'}[n_{s'}, l_1] \cdot \widetilde{\mathbf{\Phi}}_{s'}[n_{s'}, l] \right) \ .$$

Algorithm 1: `wrap`: Extract optimization variables from loading matrices. The set $\mathcal{I}_s = \left\{ \widetilde{n}_s : x_s^{\widetilde{\mathrm{g}}}[\widetilde{n}_s] \neq x_s^{\mathrm{g}}[n_s] \ \forall \, n_s \right\}$ contains only the indices of grid points that are *not* part of the coarse grid.

Input: Loading matrices $\widetilde{\mathbf{\Phi}}_1, \dots, \widetilde{\mathbf{\Phi}}_S$; sets $\mathcal{I}_s$ for $s = 1, \dots, S$
Output: Vector $\underline{\varphi}$ of new loading matrix entries

```
k ← 0;
for s = 1, ..., S do
    for l = 1, ..., L do
        for ñ_s ∈ I_s do
            k ← k + 1;
            φ_k ← Φ̃_s[ñ_s, l];
        end
    end
end
return φ;
```

It is needed for the optimizer to enforce normalization via a nonlinear equality constraint (28).

VII. Optimization

We formulate the following optimization problem to compute the best new loading matrices

$$\left\{ \widehat{\widetilde{\mathbf{\Phi}}}_1, \dots, \widehat{\widetilde{\mathbf{\Phi}}}_S \right\} = \underset{\widetilde{\mathbf{\Phi}}_1, \dots, \widetilde{\mathbf{\Phi}}_S}{\arg\min} \, I_F^R(\widetilde{\mathbf{\Phi}}_1, \dots, \widetilde{\mathbf{\Phi}}_S) \tag{27}$$

$$\text{s.t. } Z(\widetilde{\mathbf{\Phi}}_1, \dots, \widetilde{\mathbf{\Phi}}_S) = 1 \tag{28}$$

$$\text{and } \widetilde{\mathbf{\Phi}}_s[\widetilde{n}_s, l] = \mathbf{\Phi}_s[n_s, l]_{\,\lrcorner} \tag{29}$$

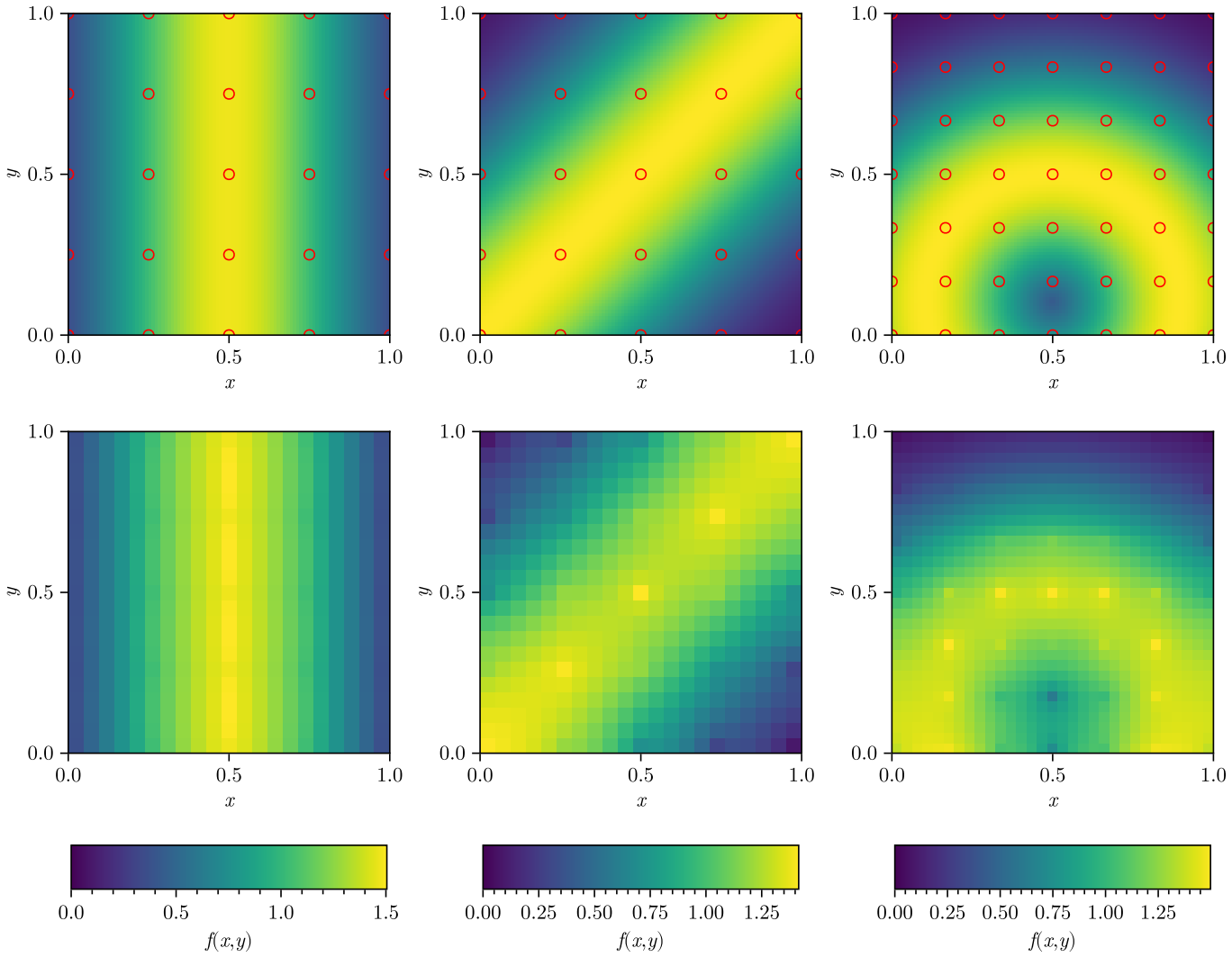
$$\text{for all } \widetilde{\mathbf{\Phi}}_s[\widetilde{n}_s, l] \text{ with } x_s^{\widetilde{\mathrm{g}}}[\widetilde{n}_s] = x_s^{\mathrm{g}}[n_s] \ .$$

Equivalently to the equality constraints (29), the optimization can be performed only over the new entries of the loading matrices $\widetilde{\mathbf{\Phi}}_s$, i.e., those for which there exists no n_s such that $x_s^{\mathrm{g}}[n_s] = x_s^{\widetilde{\mathrm{g}}}[\widetilde{n}_s]$. As the other entries can simply be kept out of the optimization, we can formulate and compute it as a nonlinear optimization problem with

$$\sum_{s=1}^{S} \left(\widetilde{N}_s - N_s \right) \cdot L_{\text{out}} \tag{30}$$

variables and one equality constraint (28).

The cost function is the FIN $I_F^R(\widetilde{r})$ given in (41), and the normalization constant Z in (23), all using the new loading matrices $\widetilde{\mathbf{\Phi}}_s$ (containing both old and added entries), where optimization is only performed over the added entries. Since multivariate optimizers usually require the format of the variables to be a vector, we need suitable wrapper and unwrapper functions to convert the new loading matrices to a vector (that does not contain the new loading matrix entries corresponding to the old grid points) and back, see Algorithms 1 and 2.



(a) vertical pattern, 36 s (b) diagonal pattern, 53 s (c) curved pattern, 21 s

Fig. 3: Examples of grid refinement via the proposed FIN minimization. Top: ground truth input density (coloring), evaluated on a coarse $N_1 \times N_2$ grid (red circles), with $N_1 = N_2 \in \{5, 7\}$. The density matrix is represented in CPD format with rank $L = N_1$. Bottom: interpolated density (coloring) on a refined $\widetilde{N}_1 = \widetilde{N}_2$ grid with $\widetilde{N}_1 = \widetilde{N}_2 = (N_1 - 1) \cdot 5 + 1 \in \{21, 31\}$, with rank $L = 10$, after optimizing the new loading matrix entries by minimizing the FIN under the normalization constraint. The 25 or 49 density values from the low-resolution grid that remain unchanged in the optimization stand out in some of the bottom figures. Computation times are given in the captions.

VIII. Implementation Recipe

For the convenience of the reader who wishes to reproduce the method, we summarize the implementation in a step-by-step recipe. Define the dimension S, the coarse and fine grid sizes N_s and $\widetilde{N}_s$, and the grid spacings Δx_s^{g} and $\Delta x_s^{\widetilde{\mathrm{g}}}$ such that the fine grid contains the coarse grid. Then, define the input density on the coarse grid as a tensor in CPD format, e.g., via a singular value decomposition of the square root of the density values on the full grid if $S = 2$. This yields the S coarse loading matrices $\mathbf{\Phi}_s \in \mathbb{R}^{N_s \times L_{\mathrm{in}}}$.

Next, define the S new loading matrices $\widetilde{\mathbf{\Phi}}_s \in \mathbb{R}^{\widetilde{N}_s \times L_{\mathrm{out}}}$ for the fine grid, where the N_s entries (out of $\widetilde{N}_s$ entries) that correspond to the coarse grid points are set to the coarse values, and the remaining entries are initialized, e.g., with random values or a linear interpolation of the coarse loading vectors. If the output rank L_{out} is larger than the input rank L_{in}, $(L_{\mathrm{out}} - L_{\mathrm{in}})$ additional columns are appended to the new loading matrices, with the entries corresponding to the coarse grid points fixed to zero.

Implement the FIN (16) and the normalization constant (24). For verification, the simpler full-grid forms (14) and (22) based on the explicit root density may be implemented first and used as a reference for testing the CPD versions. Also implement the wrapper and unwrapper, Algorithms 1 and 2. Solve the optimization problem using a gradient-based nonlinear optimizer that can handle nonlinear constraints, e.g., an interior-

Algorithm 2: `unwrap`: Update loading matrices with optimization variables.

Input: Vector $\underline{\varphi}$ of new loading matrix entries; sets $\mathcal{I}_s$
Output: Loading matrices $\widetilde{\mathbf{\Phi}}_1, \ldots, \widetilde{\mathbf{\Phi}}_S$

```
k ← 0;
for s = 1, ..., S do
    for l = 1, ..., L do
        for ñ_s ∈ I_s do
            k ← k + 1;
            Φ̃_s[ñ_s, l] ← φ_k;
        end
    end
end
return Φ̃_1, ..., Φ̃_S;
```

point optimizer, to minimize the cost function with respect to the new loading matrix entries under the normalization constraint, using automatic differentiation for the gradients.

Computation can then be accelerated by implementing the analytic gradients (26) and (42), which should be validated against the automatic-differentiation versions. An in-place implementation free of repeated allocations further increases computational efficiency. The method generalizes to non-equidistant grids by using the more general forms of the FIN and normalization constant in (41) and (23), respectively.

IX. Evaluation

A trivial 1D example, where no tensor decomposition is actually needed, is shown in Fig. 2 for illustrative purposes. We can see that the proposed interpolation is quite different from linear interpolation (Fig. 2a) but much more suitable for, e.g., Gaussian densities (Fig. 2b).

Then for 2D demonstrations, we defined exemplary continuous underlying densities, see top row in Fig. 3. We properly normalized those continuous densities via numerical integration with Julia's SciML Integrals.jl package [23] using the HCubature.jl backend [24], [25]. We evaluated those densities coarsely at the locations of the input grid, yielding a matrix of probability values. A tensorized representation of the root density of $\mathbf{P}$ was then obtained via element-wise square root followed by singular value decomposition

$$\mathrm{sqrt.}(\mathbf{P}) = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top\,, \quad \mathbf{\Phi}_1 = \mathbf{U}\sqrt{\mathbf{\Sigma}}\,, \quad \mathbf{\Phi}_2 = \mathbf{V}\sqrt{\mathbf{\Sigma}}\,. \tag{31}$$

We then refined the grid to a finer grid and optimized the new loading matrix entries as described above. Tensor ranks were $L_{\mathrm{in}} = N_s$ and $L_{\mathrm{out}} = 10$. The resulting interpolated densities (without normalization) are shown in the second row of Fig. 3. Optimization took between $21\,\mathrm{s}$ and $53\,\mathrm{s}$ on a desktop PC. There were no convergence issues; the interior-point optimization algorithm always converged to the same result.

We implemented the method in the Julia programming language [26], using the IPOPT algorithm [27] for constrained optimization and verified the correctness of the gradient implementations by comparing them to automatic differentiation using ForwardDiff.jl [28]. Source code of our implementation will be made available on CodeOcean and linked on the IEEE Xplore page.

X. Conclusion

A. Summary

We described a novel method for interpolating probability densities defined on rectilinear grids. Through tensor representation, we have only polynomial complexity in the dimension. In particular, the computational complexity per cost/gradient iteration is $\mathcal{O}(S^2 \cdot L^2 \cdot \widetilde{N}_{\max})$ with dimension S, rank L, and maximum grid size $\widetilde{N}_{\max}$. The method is based on the idea of minimizing the FIN of the resulting density on the finer grid under the normalization constraint. The FIN is a measure of the amount of information added to the density in the interpolation process. We demonstrated the method on 1D and 2D examples, showing that it can yield quite different results from linear interpolation, and that it can be more suitable for certain types of densities, e.g., Gaussian densities.

B. Limitations

A limitation is that the underlying density must be normalized. Furthermore, the resulting interpolated density often appears too "peaky" at some of the input grid locations, see Figs. 2c, 3b and 3c. We also observe that the diagonal strip in Fig. 3b exhibits more irregularities (yellow peaks) than the vertical strip in Fig. 3a, the cause of which remains to be investigated. Finally, the runtime is still too high for practical applications.

C. Outlook

In the future, we plan to investigate alternatives to the simple forward differences used here, other Riemann sums (e.g., the midpoint rule for the integral), other boundary conditions, and other derivations of the FIN that may be faster to compute. We also want to investigate unevenly spaced grids as well as the extrapolation capabilities. A natural extension is the generalization to nonlinear manifolds, building on prior work on the cylindrical manifold [8] and the spherical manifold [29], [30]. An automatic determination of an appropriate L would be helpful. It should be possible to estimate the normalization constant as well if the underlying density is not normalized. Another open question is how the relationship between density values on the old and new grid can be specified. Apart from "identical density values at identical grid locations" as used here, one could, for example, require that the coarse density values describe the average density in the coarse grid cell, or penalize the second derivative at the data points to avoid the peakiness. We also want to improve the computational efficiency via an implementation that self-normalizes and is thus free of the normalization constraint, which will make it run about 10 times faster.

References

[1] Tamara G. Kolda and Brett W. Bader. "Tensor Decompositions and Applications". In: *SIAM Review* 51.3 (2009), pp. 455–500. DOI: 10.1137/07070111X.

[2] Gregory Beylkin and Martin J. Mohlenkamp. "Algorithms for Numerical Analysis in High Dimensions". In: *SIAM Journal on Scientific Computing* 26.6 (2005), pp. 2133–2159. DOI: 10.1137/040604959.

[3] Frank L. Hitchcock. "The Expression of a Tensor or a Polyadic as a Sum of Products". In: *Journal of Mathematics and Physics* 6.1-4 (1927), pp. 164–189. DOI: 10.1002/sapm192761164.

[4] Nicholas D. Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E. Papalexakis, and Christos Faloutsos. "Tensor Decomposition for Signal Processing and Machine Learning". In: *IEEE Transactions on Signal Processing* 65.13 (2017), pp. 3551–3582. DOI: 10.1109/TSP.2017.2690524.

[5] Felix Govaers, Bruno Demissie, Altamash Khan, Martin Ulmke, and Wolfgang Koch. "Tensor Decomposition-Based Multitarget Tracking in Cluttered Environments". In: *Journal of Advances in Information Fusion* 14.1 (2019), p. 86. URL: https://isif.org/media/tensor-decomposition-based-multitarget-tracking-cluttered-environments.

[6] Felix Govaers. "On Statistics based Prediction of Decomposed 1–6 Tensor Probability Density Functions". In: *2023 IEEE Symposium Sensor Data Fusion and International Conference on Multisensor Fusion and Integration (SDF-MFI)*. 2023, pp. 1–6. DOI: 10.1109/SDF-MFI59545.2023.10361353.

[7] Bruno Demissie, Muhammad Altamash Khan, and Felix Govaers. "Nonlinear Filter Design Using Fokker-Planck Propagator in Kronecker Tensor Format". In: *2016 19th International Conference on Information Fusion (FUSION)*. 2016, pp. 1–8.

[8] Daniel Frisch and Uwe D. Hanebeck. "Fokker-Planck Prediction on the Cylindric Manifold using Tensor Decomposition of a Regular Grid". In: *17th Symposium Sensor Data Fusion: Trends, Solutions, Applications (SDF 2025)*. Bonn, Germany, Nov. 2025. DOI: 10.1109/SDF67080.2025.11330240.

[9] J. Matou v̲ sek, J. Krej v̲ cí, J. Duník, and R. Zanetti. *Tensor Decompositions for Online Grid-Based Terrain-Aided Navigation*. 2026. URL: https://arxiv.org/abs/2601.07728.

[10] Daniel Frisch and Uwe D. Hanebeck. "Density Estimation from Weighted Samples with the Localized Cumulative Distribution". In: *Proceedings of the 23rd IFAC World Congress (IFAC 2026)*. Busan, Republic of Korea, Aug. 2026.

[11] Daniel Frisch and Uwe D. Hanebeck. "Progressive Bayesian Filtering with Coupled Gaussian and Dirac Mixtures". In: *Proceedings of the 23rd International Conference on Information Fusion (Fusion 2020)*. Virtual, July 2020. DOI: 10.23919/FUSION45008.2020.9190540.

[12] Daniel Frisch and Uwe D. Hanebeck. "Gaussian Mixture Estimation from Weighted Samples". In: *Proceedings of the 2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2021)*. Karlsruhe, Germany, Sept. 2021. DOI: 10.1109/MFI52462.2021.9591161.

[13] Daniel Frisch and Uwe D. Hanebeck. "Gaussian Mixture Particle Filter Step based on Method of Moments". In: *Proceedings of the 27th International Conference on Information Fusion (FUSION 2024)*. Venice, Italy, July 2024. DOI: 10.23919/FUSION59988.2024.10706458.

[14] Georgii S. Novikov, Maxim E. Panov, and Ivan V. Oseledets. "Tensor-train density estimation". In: *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*. Ed. by Cassio de Campos and Marloes H. Maathuis. Vol. 161. Proceedings of Machine Learning Research. PMLR, July 2021, pp. 1321–1331. URL: https://proceedings.mlr.press/v161/novikov21a.html.

[15] Emanuel Parzen. "On Estimation of a Probability Density Function and Mode". In: *The Annals of Mathematical Statistics* 33.3 (1962), pp. 1065–1076. URL: http://www.jstor.org/stable/2237880 (visited on 02/28/2024).

[16] Nils-Bastian Heidenreich, Anja Schindler, and Stefan Sperlich. "Bandwidth selection for kernel density estimation: a review of fully automatic selectors". In: *AStA Advances in Statistical Analysis* 97.4 (2013), pp. 403–433. DOI: 10.1007/s10182-013-0216-y.

[17] Johan Frederik Steffensen. *Interpolation*. Williams & Wilkins, 1927.

[18] Giuseppe Toscani. "Score functions, generalized relative Fisher information and applications". In: *Ricerche di Matematica* 66.1 (2017), pp. 15–26. DOI: 10.1007/s11587-016-0281-0.

[19] Uwe D. Hanebeck, Daniel Frisch, and Dominik Prossel. "Closed-Form Information-Theoretic Roughness Measures for Mixture Densities". In: *Proceedings of the 2024 American Control Conference (ACC 2024)*. Toronto, Canada, July 2024. DOI: 10.23919/ACC60939.2024.10645015.

[20] Dominik Prossel and Uwe D. Hanebeck. "Three Approaches to Approximating the Fisher Information Number for Gaussian Mixture Densities". In: *Proceedings of the 2024 IEEE International Conference on Multisensor Fusion and Integration (MFI 2024)*. Pilsen, Czechia, Sept. 2024. DOI: 10.1109/MFI62651.2024.10705763.

[21] Dominik Prossel and Uwe D. Hanebeck. "Spline-Based Density Estimation Minimizing Fisher Information". In: *Proceedings of the 27th International Conference on Information Fusion (FUSION 2024)*. Venice, Italy, July 2024. DOI: 10.23919/FUSION59988.2024.10706290.

[22] Dana El-Rushaidat, Raine Yeh, and Xavier M. Tricoche. "Accurate parallel reconstruction of unstructured datasets on rectilinear grids". In: *Journal of Visualization* 24.4 (2021), pp. 787–806. DOI: 10.1007/s12650-020-00740-0.

[23] Christopher Rackauckas and Qing Nie. "DifferentialEquations.jl–a Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia". In: *Journal of Open Research Software* 5.1 (2017). DOI: 10.5334/jors.151.

[24] Steven G. Johnson. *The HCubature.jl package for multi-dimensional adaptive integration in Julia*. https://github.com/JuliaMath/HCubature.jl. 2017.

[25] Alan C. Genz and Aftab Ahmad Malik. "Remarks on Algorithm 006: An Adaptive Algorithm for Numerical Integration Over an N-Dimensional Rectangular Region". In: *Journal of Computational and Applied mathematics* 6.4 (1980), pp. 295–302.

[26] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. "Julia: A Fresh Approach to Numerical Computing". In: *SIAM review* 59.1 (2017), pp. 65–98. URL: https://doi.org/10.1137/141000671.

[27] Andreas Wächter and Lorenz T. Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical Programming* 106.1 (2006), pp. 25–57. DOI: 10.1007/s10107-004-0559-y.

[28] J. Revels, M. Lubin, and T. Papamarkou. "Forward-Mode Automatic Differentiation in Julia". In: *arXiv* (2016). URL: https://arxiv.org/abs/1607.07892.

[29] Daniel Frisch and Uwe D. Hanebeck. "Deterministic Sampling with Separation of Variables in Spherical Coordinates". In: *Proceedings of the 28th International Conference on Information Fusion (FUSION 2025)*. Rio de Janeiro, Brazil, July 2025. DOI: 10.23919/FUSION65864.2025.11124009.

[30] Daniel Frisch and Uwe D. Hanebeck. "Deterministic Von Mises-Fisher Sampling on the Sphere Using Fibonacci Lattices". In: *Proceedings of the combined IEEE 2023 Symposium Sensor Data Fusion and International Conference on Multisensor Fusion and Integration (SDF-MFI 2023)*. Bonn, Germany, Nov. 2023. DOI: 10.1109/SDF-MFI59545.2023.10361396.

APPENDIX

A. Fisher Information Number in Tensorized Form

To obtain a tensorized representation of the FIN, we have to insert the tensor decomposition form (5) of the root density into (14), which we will do in a gradual manner. We begin with the numerator of the fraction in (14)

$$r[\underline{n}+\underline{e}_s]-r[\underline{n}] \tag{32}$$
$$=\sum_{l=1}^{L}\prod_{s'=1}^{S}\mathbf{\Phi}_{s'}[n_{s'}+\delta_{s,s'},l]-\sum_{l=1}^{L}\prod_{s'=1}^{S}\mathbf{\Phi}_{s'}[n_{s'},l] \tag{33}$$
$$=\sum_{l=1}^{L}\left(\prod_{s'=1}^{S}\mathbf{\Phi}_{s'}[n_{s'}+\delta_{s,s'},l]-\prod_{s'=1}^{S}\mathbf{\Phi}_{s'}[n_{s'},l]\right) \tag{34}$$
$$=\sum_{l=1}^{L}(\mathbf{\Phi}_s[n_s+1,l]-\mathbf{\Phi}_s[n_s,l])\prod_{\substack{s'=1\\s'\neq s}}^{S}\mathbf{\Phi}_{s'}[n_{s'},l]\ . \tag{35}$$

Then take the square of this expression

$$(r[\underline{n}+\underline{e}_s]-r[\underline{n}])^2 \tag{36}$$
$$=\left(\sum_{l=1}^{L}(\mathbf{\Phi}_s[n_s+1,l]-\mathbf{\Phi}_s[n_s,l])\prod_{\substack{s'=1\\s'\neq s}}^{S}\mathbf{\Phi}_{s'}[n_{s'},l]\right)^2 \tag{37}$$
$$=\sum_{l_1=1}^{L}\sum_{l_2=1}^{L}(\mathbf{\Phi}_s[n_s+1,l_1]-\mathbf{\Phi}_s[n_s,l_1])_{\lrcorner} \tag{38}$$
$$\cdot(\mathbf{\Phi}_s[n_s+1,l_2]-\mathbf{\Phi}_s[n_s,l_2])_{\lrcorner}$$
$$\cdot\prod_{\substack{s'=1\\s'\neq s}}^{S}\mathbf{\Phi}_{s'}[n_{s'},l_1]\cdot\mathbf{\Phi}_{s'}[n_{s'},l_2]\ .$$

The S-fold multi-sum of this expression over the spatial multi-index $\underline{n}$ can be separated into a product of S single sums

$$\sum_{\underline{n}=\underline{1}}^{\underline{N}-\underline{e}_s}(r[\underline{n}+\underline{e}_s]-r[\underline{n}])^2 \tag{39}$$
$$=\sum_{l_1=1}^{L}\sum_{l_2=1}^{L}\left(\sum_{n_s=1}^{N_s-1}(\mathbf{\Phi}_s[n_s+1,l_1]-\mathbf{\Phi}_s[n_s,l_1])_{\lrcorner}\right. \tag{40}$$
$$\cdot(\mathbf{\Phi}_s[n_s+1,l_2]-\mathbf{\Phi}_s[n_s,l_2])_{\lrcorner}$$
$$\left.\cdot\prod_{\substack{s'=1\\s'\neq s}}^{S}\sum_{n_{s'}=1}^{N_{s'}}\mathbf{\Phi}_{s'}[n_{s'},l_1]\cdot\mathbf{\Phi}_{s'}[n_{s'},l_2]\right)\ .$$

In total, inserting this into (14), we obtain the following tensorized form of the FIN

$$I_F^R(\mathbf{\Phi}_1,\dots,\mathbf{\Phi}_S)=4\sum_{l_1=1}^{L}\sum_{l_2=1}^{L}\sum_{s=1}^{S}{}_{\lrcorner} \tag{41}$$
$$\left(\sum_{n_s=1}^{N_s-1}(\mathbf{\Phi}_s[n_s+1,l_1]-\mathbf{\Phi}_s[n_s,l_1])_{\lrcorner}\right.$$
$$\cdot(\mathbf{\Phi}_s[n_s+1,l_2]-\mathbf{\Phi}_s[n_s,l_2])_{\lrcorner}$$
$$\left.\cdot\frac{1}{x_s^{\mathrm{g}}[n_s+1]-x_s^{\mathrm{g}}[n_s]}\right)_{\lrcorner}$$
$$\cdot\prod_{\substack{s'=1\\s'\neq s}}^{S}\left(\sum_{n_{s'}=1}^{N_{s'}}\mathbf{\Phi}_{s'}[n_{s'},l_1]\cdot\mathbf{\Phi}_{s'}[n_{s'},l_2]_{\lrcorner}\right.$$
$$\left.\cdot(x_{s'}^{\mathrm{g}}[n_{s'}+1]-x_{s'}^{\mathrm{g}}[n_{s'}])\right)\ .$$

A simplified version for equidistant grids is shown in (16).

B. FIN Gradient in Tensorized Form

We are going to need the gradient of the FIN with respect to any entry of a loading matrix. For an equidistant grid, this is given by differentiating (16) and yields

$$\frac{\partial I_F^R(\mathbf{\Phi}_1,\dots,\mathbf{\Phi}_S)}{\partial\mathbf{\Phi}_s[n_s,l]}=c\cdot 2\sum_{l_1=1}^{L}\frac{1}{(\Delta x_s^{\mathrm{g}})^2} \tag{42}$$
$$\cdot\big(\delta_{n_s\in[2,N_s]}\cdot\mathbf{\Phi}_s[n_s,l_1]$$
$$-\delta_{n_s\in[1,N_s-1]}\cdot\mathbf{\Phi}_s[n_s+1,l_1]$$
$$-\delta_{n_s\in[2,N_s]}\cdot\mathbf{\Phi}_s[n_s-1,l_1]$$
$$+\delta_{n_s\in[1,N_s-1]}\cdot\mathbf{\Phi}_s[n_s,l_1]\big)$$
$$\cdot\prod_{\substack{s'=1\\s'\neq s}}^{S}\left(\sum_{n_{s'}=1}^{N_{s'}}\mathbf{\Phi}_{s'}[n_{s'},l_1]\cdot\mathbf{\Phi}_{s'}[n_{s'},l]\right)$$
$$+\mathbf{\Phi}_s[n_s,l_1]\cdot\sum_{\substack{s''=1\\s''\neq s}}^{S}\frac{1}{(\Delta x_{s''}^{\mathrm{g}})^2}\sum_{n_{s''}=1}^{N_{s''}-1}$$
$$(\mathbf{\Phi}_{s''}[n_{s''}+1,l_1]-\mathbf{\Phi}_{s''}[n_{s''},l_1])$$
$$\cdot(\mathbf{\Phi}_{s''}[n_{s''}+1,l]-\mathbf{\Phi}_{s''}[n_{s''},l])$$
$$\cdot\prod_{\substack{s'=1\\s'\neq s\\s'\neq s''}}^{S}\left(\sum_{n_{s'}=1}^{N_{s'}}\mathbf{\Phi}_{s'}[n_{s'},l_1]\cdot\mathbf{\Phi}_{s'}[n_{s'},l]\right)\ .$$