# Multidimensional approximation of nonlinear dynamical systems

Patrick Gelß[1], Stefan Klus[1], Jens Eisert[2], and Christof Schütte[1,3]

[1]Department of Mathematics and Computer Science, Freie Universität Berlin, Germany
[2]Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, Germany
[3]Zuse Institute Berlin, Germany

**Abstract**

A key task in the field of modeling and analyzing nonlinear dynamical systems is the recovery of unknown governing equations from measurement data only. There is a wide range of application areas for this important instance of system identification, ranging from industrial engineering and acoustic signal processing to stock market models. In order to find appropriate representations of underlying dynamical systems, various data-driven methods have been proposed by different communities. However, if the given data sets are high-dimensional, then these methods typically suffer from the curse of dimensionality. To significantly reduce the computational costs and storage consumption, we propose the method *MANDy* which combines data-driven methods with tensor network decompositions. The efficiency of the introduced approach will be illustrated with the aid of several high-dimensional nonlinear dynamical systems.

**Keywords:** Nonlinear dynamics, system identification, data-driven methods, tensor networks, tensor-train format

**2010 MSC:** 15A69, 34C15, 62J05, 65E05, 93B30

## 1 Introduction

The identification of governing equations from data plays an increasingly important role in the field of nonlinear dynamical systems. After all, in many practically relevant situations, the actual underlying model that captures a dynamical system is unknown, while the only information available is data arising from the natural time evolution. In this instance of system identification, given time-dependent measurement data, the task at hand is to reconstruct the underlying system, merely assuming that the governing equations can be accurately approximated by linear combinations of preselected basis functions. We consider autonomous dynamical systems given by ordinary differential equations (ODEs). Many physical – even chaotic – systems have a simple ODE structure comprising only a small number of terms, e.g., monomials or trigonometric functions and combinations thereof. That is, if we choose a large set of basis functions, then a system may be described in terms of only a few of these basis functions, or – in other words – the governing equations are sparse in the high-dimensional space of selected basis functions, cf. [1, 2, 3]. Essentially, we are interested in the relationship between two data sets, namely measurements of $X(t)$ and measurements of $\dot{X}(t)$ at certain time points $t$. The methods we will discuss below (as well as the approach we propose), however, can be applied to a wider range of problems. For instance, we could also consider discrete dynamical systems or stochastic differential equations (SDEs). The so-called SINDy approach [4] has also already been extended to identify the governing equations of SDEs, see [5].

The challenge here is to recover the dynamics (i.e., to recover $F$ such that $\dot{X}(t) = F(X(t))$) from a rather small number of measurements of the time-dependent states $X(t)$ and the corresponding derivatives. This can be accomplished by using different approaches, see, for example, [6]. One of them is *symbolic regression* [7], where evolutionary algorithms reveal appropriate models that fit the given measurement data by combining mathematical expressions. Unfortunately, symbolic regression is in general too expensive and the convergence might be slow for high-dimensional data sets because discovering the governing equations from measurement data in a high-dimensional space $\mathbb{R}^d$ can be prohibitively expensive in terms of memory consumption and computational costs. This phenomenon is often referred to as the *"curse of dimensionality"*. Also neural networks can be used to approximate the governing equations of dynamical systems (and, in fact, to approximate any multidimensional function), see [8]. As for the symbolic regression, knowledge about the governing equations is not needed a priori for these purely data-driven methods. However, the approximation based on neural networks does not necessarily lead to interpretable representations of the governing equations since the given data is only interpolated by using artificial activation functions, e.g., radial basis functions.

In [4], Brunton et al. proposed an algorithm for the *sparse identification of nonlinear dynamical systems* (SINDy). Given data measurements of states and corresponding derivatives, a certain set of basis functions has to be chosen a priori. Based on sequential least-squares approximations and an iterative selection of relevant basis functions – which is comparable to *iterative hard thresholding* [9] – the application of SINDy often results in a sparse representation of the underlying system, provided that we choose suitable basis functions. We could also add an $L^1$-regularization term to the regression problem if sparsity of the solution should be enforced. In the community of compressed sensing, this method is known as the *least absolute shrinkage and selection operator* (LASSO), see [10]. However, for large data sets (particularly for large dimensions $d$) the above methods still suffer from the curse of dimensionality. Thus, there is a need for efficient methods that are able to recover the governing equations of high-dimensional nonlinear dynamical systems.

In this work, we combine the data-driven discovery of the underlying dynamics with the computation of pseudoinverses of tensor-network decompositions. Tensors are generalizations of vectors and matrices represented by multidimensional arrays with multiple indices. The interest in low-rank tensor decompositions has been growing rapidly over the last years since several tensor formats such as the canonical format [11, 12], the Tucker format [13, 14], and the hierarchical Tucker format [15, 16] have shown that it is possible to mitigate the curse of dimensionality for many high-dimensional problems. Tensor-based methods have been successfully applied to many different application areas, e.g., quantum physics [17, 18, 19, 20, 21], chemical reaction dynamics [22, 23], machine learning [24, 25], and high-dimensional data analysis [26, 27].

It turned out that one of the most promising tensor formats is the so-called tensor-train format (TT format) [28, 29, 30] or, equivalently, the matrix product state (MPS) format [31, 32]. It is a special case of the hierarchical Tucker format and combines the advantages of the canonical format and the Tucker format, i.e., the storage consumption of a tensor train does not depend exponentially on the number of dimensions and there exist robust algorithms for the computation of best approximations. The fact that the TT format has emerged independently in several fields of science signifies its importance. In quantum physics, the MPS representation, on the one hand, dates back to work on the density-matrix renormalization group [17]. On the other hand, pure finitely correlated states [33] are again basically quantum states in the TT format, originating in mathematical physics in efforts to generalize notions of hidden Markov models. Within numerical mathematics, the use of tensor network decompositions and approximations is rather new, which has led to an exciting development. However, many questions are still open, specifically when it comes to the convergence when approximating solutions of systems of linear equations given in the TT format, cf. [34]. For an overview of different low-rank tensor approximation approaches, we refer to [35, 36, 19].

The new role of tensor decompositions for data analysis techniques and especially compressed sensing was already discussed in, e.g., [27, 37, 38]. Using the TT format we shift the focus from *sparse* structures to *low-rank* or *data-sparse structures*. That is, instead of determining sparse vectors or matrices, we aim at recovering dynamical systems by utilizing tensor decompositions with a moderate number of core elements. Applying the approach proposed in [27] to construct pseudoinverses in the TT format, we can extend the method to identify governing equations so that low-rank representations of the data can be utilized to reduce the computational complexity as well as the memory requirements. Referring to SINDy, we call our method MANDy, which is short for *multidimensional approximation of nonlinear dynamical systems*.

Data-driven discovery of dynamical systems will continue to play an increasingly important role. Here, we will show that rewriting mathematical methods by exploiting low-rank tensor decompositions enables the reconstruction of high-dimensional systems which cannot be analyzed by conventional methods. The main contributions of this paper are:

- combination of data-driven methods with tensor decomposition techniques for the recovery of governing equations, reducing computational costs as well as storage consumption,
- extension of the pseudoinverse computation in the TT format (developed for *tensor-based dynamic mode decomposition* in [27]),
- derivation of different basis decompositions in the TT format,
- demonstration on realistic problems from physics and engineering.

This work is organized as follows: In Section 2, we introduce SINDy and give a brief overview of the TT format and pseudoinverses in the TT format. In Section 3, we derive the tensor-based reformulation of SINDy. Numerical results are presented in Section 4. Section 5 concludes with a brief summary and a future outlook.

## 2 Notation and preliminaries

In this section, we will introduce an approach called SINDy to identify ordinary differential equations from data, the tensor-train format, and an algorithm to compute pseudoinverses in the tensor-train format. By combining these techniques, we are able to discover the governing equations of high-dimensional dynamical systems.

### 2.1 Sparse identification of nonlinear dynamics

We will consider autonomous ordinary differential equations of the form $\dot{X}(t) = F(X(t))$, where $X(t) \in \mathbb{R}^d$ is the state of the system at time $t$ and $F\colon \mathbb{R}^d \to \mathbb{R}^d$ is a function. Assuming we have $m$ measurements of the state of the system, given by $X_k$, $k = 1, \dots, m$, and the corresponding time derivatives, given by $Y_k = \dot{X}_k$, the goal is to reconstruct the function $F$. To this end, we choose a set of basis functions (also called dictionary) $\mathbb{D} = \{\psi_1, \psi_2, \dots, \psi_p\}$, with $\psi_j\colon \mathbb{R}^d \to \mathbb{R}$, and define the vector-valued function $\Psi\colon \mathbb{R}^d \to \mathbb{R}^p$ by

$$\Psi(X) = \begin{bmatrix} \psi_1(X) & \psi_2(X) & \dots & \psi_p(X) \end{bmatrix}^T .$$

The transformed data matrix $\Psi(\mathcal{X})$ is then defined by

$$\Psi(\mathcal{X}) = \begin{bmatrix} \Psi(X_1) & \dots & \Psi(X_m) \end{bmatrix} \in \mathbb{R}^{p \times m}. \tag{1}$$

We now wish to determine the coefficient matrix

$$\Xi = \begin{bmatrix} \xi_1 & \xi_2 & \dots & \xi_d \end{bmatrix} \in \mathbb{R}^{p \times d}$$

such that the cost function $\left\| \mathcal{Y} - \Xi^T \Psi(\mathcal{X}) \right\|_F$ is minimized. Each column vector $\xi_i$ of $\Xi$ then corresponds to one function $F_i$ via

$$(Y_k)_i = F_i(X_k) = \xi_i^T \Psi(X_k),$$

where the entries of $\xi_i$ determine which basis functions are used for the reconstruction. Thus, we obtain a model of the form $\dot{X}(t) = \Xi^T \Psi(X(t))$. One approach to compute an optimal coefficient matrix in the least-squares sense is the use of the pseudoinverse of $\Psi(\mathcal{X})$. That is, we determine $\Xi$ by $\Xi^T = \mathcal{Y} \cdot (\Psi(\mathcal{X}))^+$, where $^+$ denotes the pseudoinverse. Additionally, SINDy aims at finding a sparse coefficient matrix $\Xi$ [4]. This is accomplished by iteratively removing basis functions corresponding to small entries that might not be required for the reconstruction. Provided that the governing equations can be expressed in terms of the basis functions, SINDy may completely recover the dynamical system. The iterative elimination of basis functions could be omitted if the number of snapshots is large enough so that the solution of the initial least-squares problem is already close to the true solution. Alternatively, a LASSO-based (see, e.g., [10]) approach can be used to identify the coefficients.

If the time derivatives are not available and need to be approximated by finite differences, the resulting $\dot{X}$ data might be noisy and necessitate denoising techniques. Also measurement data will in general contain noise and require regularization. For more details, see [4] and references therein.

**Example 2.1** (Illustration of SINDy)**.** To illustrate the sparse identification process, let us begin with a simple example. Consider Chua's circuit, see, e.g., [39], given by

$$\begin{aligned} \dot{x}_1 &= \alpha(x_2 - x_1 - g(x_1)), \\ \dot{x}_2 &= x_1 - x_2 + x_3, \\ \dot{x}_3 &= -\beta x_2, \end{aligned} \tag{2}$$

where $\alpha$ and $\beta$ are real parameters and $g\colon \mathbb{R} \to \mathbb{R}$ is a nonlinear function. We will set $\alpha = 10$, $\beta = 14.87$, and $g(z) = \delta_1 z + \delta_2 z\,|z|$, with $\delta_1 = -\frac{8}{7}$ and $\delta_2 = \frac{4}{63}$. Let us try to recover the governing equations of Chua's circuit from data. We simulate the system for $t = 0, \dots, 20$ with a step size of $h = 0.01$ and the initial condition $X_1 = [-1.13, 0.004, 0.45]^T$, thus $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{3\times 2000}$. Here, we use the exact derivatives.

(i) Using a basis comprising monomials of order up to and including two in each dimension, i.e.,

$$\Psi_1(X) = \begin{bmatrix} 1 & x_1 & x_1^2 & x_2 & x_1x_2 & x_1^2x_2 & \dots & x_2^2x_3^2 & x_1x_2^2x_3^2 & x_1^2x_2^2x_3^2 \end{bmatrix}^T, \tag{3}$$

we obtain the following coefficient matrix

$$\Xi_1^T = \begin{array}{c} \begin{array}{ccccccccccc} 1 & x_1 & x_1^2 & x_2 & x_1x_2 & x_1^2x_2 & x_2^2 & x_1x_2^2 & x_1^2x_2^2 & x_3 & \dots \end{array} \\ \left[\begin{array}{ccccccccccc} -0.06 & 1.10 & 0.35 & 9.61 & -1.77 & -1.97 & 1.05 & 3.85 & 1.07 & -0.07 & \dots \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & \dots \\ 0 & 0 & 0 & -14.87 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \end{array}\right] \end{array}.$$

The coefficients for the second and third function are identified correctly, whereas the first equation cannot be represented by the basis functions and all coefficients are unequal to zero, approximating the missing basis function. The resulting trajectories are shown in Figure 1 (a).

(ii) If we use a different set of basis functions, given by

$$\Psi_2(X) = \begin{bmatrix} 1 & x_1 & x_2 & x_3 & |x_1| & x_1\,|x_1| & x_2\,|x_1| & x_3\,|x_1| & \dots & x_3\,|x_3| & x_3\,|x_3| \end{bmatrix}^T \tag{4}$$

the system is recovered correctly as

$$\Xi_2^T = \begin{array}{c} \begin{array}{ccccccc} 1 & x_1 & x_2 & x_3 & |x_1| & x_1|x_1| & \end{array} \\ \left[\begin{array}{ccccccc} 0 & 1.42 & 10 & 0 & 0 & -0.6349 & \dots \\ 0 & 1 & -1 & 1 & 0 & 0 & \dots \\ 0 & 0 & -14.87 & 0 & 0 & 0 & \dots \end{array}\right] \end{array}.$$

All remaining coefficients are numerically zero. The simulation results are shown in Figure 1 (b). △

**Remark 2.2** (Dependence of SINDy on basis functions)**.** The example illustrates that in order to obtain an accurate representation of the system, a priori knowledge about the governing equations might be required, which is in general not available when the method is applied to measurement data or data stemming from black-box models. If the basis functions are not chosen appropriately, then SINDy will not faithfully reproduce the dynamics of the system.

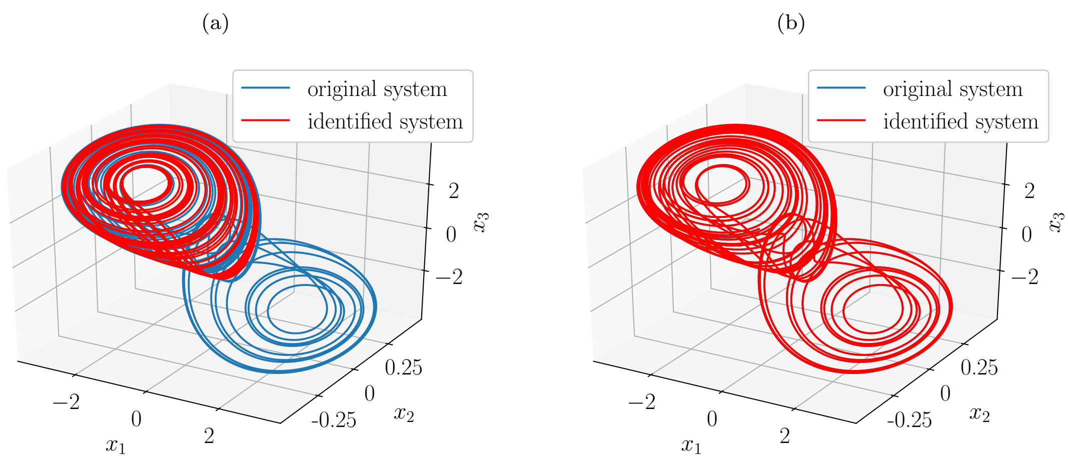

**Figure 1:** Simulation of Chua's circuit and the identified systems for $t = 0, \ldots, 100$ with a step size of $h = 0.01$. (a) Using a basis comprising only monomials leads to incorrect simulation results, the identified system does not capture the full dynamics. (b) Constructing a basis that contains also $x_1 |x_1|$ leads to correct results, the original and identified system are identical up to numerical errors.

## 2.2 The tensor-train format

Tensors of order $d$ are multidimensional arrays $\mathbf{T} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$. Figure 2 shows a number of simple examples. The different dimensions $n_i \in \mathbb{N}$ for $i = 1, \ldots, d$ are called *modes*. Elements of tensors are accessed by $\mathbf{T}_{j_1,\ldots,j_d} \in \mathbb{R}$, with $1 \leq j_i \leq n_i$. If we fix certain indices, colons are used to indicate the free modes (similar to Matlab's or Python's colon notation). The storage consumption of a tensor can be estimated as $O(n^d)$, where $n$ is the maximum of all mode sizes $n_1, \ldots, n_d$. Storing a $d$-dimensional tensor may be infeasible for large $d$ since the number of elements of a tensor grows exponentially with the order. This is typically called the *curse of dimensionality*. However, it is possible to mitigate this by exploiting low-rank tensor approximations. If the underlying correlation structure admits such a decomposition, an enormous reduction in complexity can be achieved. The basic idea is to decompose tensors $\mathbf{T} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ into different components using tensor products, see [40].

**(a)**

$$\begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

**(b)**

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

**(c)**

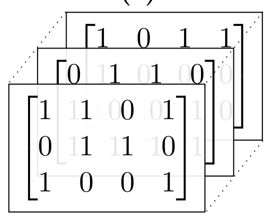

**Figure 2:** Low-dimensional tensors represented by arrays: (a) A tensor of order 1 is a vector. (b) A tensor of order 2 is a matrix. (c) A tensor of order 3 can be visualized as layers of matrices.

Examples of tensor formats are *rank-one tensors* [41, 42] and the *canonical format* [43, 44]. A tensor $\mathbf{T} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ of order $d$ is called a rank-one tensor if it can be written as the tensor product of $d$ vectors. A tensor in the canonical format is simply the sum of rank-one tensors. For our purposes, we will rely on the *tensor-train format* (TT format) [29, 30], a special case of the *hierarchical Tucker format* [15, 16]. A tensor in the TT format is given by

$$\mathbf{T} = \sum_{k_0=1}^{r_0} \cdots \sum_{k_d=1}^{r_d} \mathbf{T}^{(1)}_{k_0,:,k_1} \otimes \cdots \otimes \mathbf{T}^{(d)}_{k_{d-1},:,k_d}. \tag{5}$$

The TT ranks $r_0, \ldots, r_d$, where $r_0 = r_d = 1$, determine the storage consumption of a tensor train and its complexity. Lower ranks imply a lower memory consumption and lower computational costs. Therefore, our aim is to compute low-rank approximations of high-dimensional tensors in the TT format. Figure 3 shows the graphical representation – motivated by [34] – of a tensor train $\mathbf{T} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$. A core is depicted by a circle with different arms indicating the modes of the tensor and the rank indices. For more information, we refer to [36, 42].

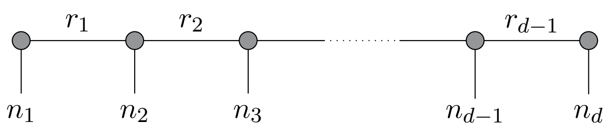

**Figure 3:** Graphical representation of tensor trains. Here, the first and the last core are considered as matrices, the other cores are tensors of order 3.

We represent the TT cores as two-dimensional arrays containing vectors as elements. For $\mathbf{T} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ with cores $\mathbf{T}^{(i)} \in \mathbb{R}^{r_{i-1} \times n_i \times r_i}$, a single core is written as

$$\left[\!\left[ \mathbf{T}^{(i)} \right]\!\right] = \left[\!\!\left[ \begin{matrix} \mathbf{T}^{(i)}_{1,:,1} & \cdots & \mathbf{T}^{(i)}_{1,:,r_i} \\ \vdots & \ddots & \vdots \\ \mathbf{T}^{(i)}_{r_{i-1},:,1} & \cdots & \mathbf{T}^{(i)}_{r_{i-1},:,r_i} \end{matrix} \right]\!\!\right]. \tag{6}$$

We then use the notation $\mathbf{T} = \left[\!\left[\mathbf{T}^{(1)}\right]\!\right] \otimes \cdots \otimes \left[\!\left[\mathbf{T}^{(d)}\right]\!\right]$ for representing tensor trains $\mathbf{T}$, see [45, 46]. This can be regarded as a generalization of the standard matrix multiplication where the cores contain vectors as elements instead of scalar values. Just like multiplying two matrices, we compute the tensor products of the corresponding elements and then sum over the columns and rows, respectively.

In order to construct *matricizations* and *vectorizations* – also called *tensor unfoldings* [34] – we define a bijection $\phi_N$ for the mode set $N = (n_1, \ldots, n_d)^T \in \mathbb{N}^d$ with

$$\phi_N \colon \{1, \ldots, n_1\} \times \cdots \times \{1, \ldots, n_d\} \to \{1, \ldots, \prod_{k=1}^{d} n_k\},$$
$$(j_1, \ldots, j_d) \mapsto \overline{j_1, \ldots, j_d} := \phi_N(j_1, \ldots, j_d),$$

where $\overline{j_1, \dots, j_d}$ is the single-index representation of the corresponding multi-index $(j_1, \dots, j_d)$. Typically, bijections based on reverse lexicographic ordering (column-major order in Matlab and Fortran) or colexicographic ordering (row-major order in Numpy and C++) are used, see, e.g., [47].

**Definition 2.3** (Matricization and vectorization)**.** *Let $N = (n_1, \dots, n_d)^T$ be a mode set and $\mathbf{T} \in \mathbb{R}^{n_1 \times \dots \times n_d}$ a tensor. For two ordered subsets $N' = (n_1, \dots, n_l)^T$ and $N'' = (n_{l+1}, \dots, n_d)^T$ of $N$ with $1 \leq l < d$, the* matricization *of $\mathbf{T}$ with respect to $N'$ and $N''$ is given by*

$$\left( \mathbf{T} \Big|_{N'}^{N''} \right)_{\overline{j_1, \dots, j_l}, \overline{j_{l+1}, \dots, j_d}} = \mathbf{T}_{j_1, \dots, j_d}. \tag{7}$$

*The* vectorization *of $\mathbf{T}$ is given by*

$$\operatorname{vec}(\mathbf{T}) := \left( \mathbf{T} \Big|_{N} \right)_{\overline{j_1, \dots, j_d}} = \mathbf{T}_{j_1, \dots, j_d}.$$

Definition 2.3 can also be generalized to arbitrary subsets of the mode set $N$, see, e.g., [36]. We will here, however, only need the special case described above. Given a TT core $\mathbf{T}^{(i)} \in \mathbb{R}^{r_{i-1} \times n_i \times r_i}$, we now define

$$\mathcal{L}\left(\mathbf{T}^{(i)}\right) = \mathbf{T}^{(i)} \Big|_{r_{i-1}, n_i}^{r_i} \qquad \text{and} \qquad \mathcal{R}\left(\mathbf{T}^{(i)}\right) = \mathbf{T}^{(i)} \Big|_{r_{i-1}}^{n_i, r_i}. \tag{8}$$

The matricization $\mathcal{L}\left(\mathbf{T}^{(i)}\right)$ is called the *left-unfolding* of $\mathbf{T}^{(i)}$ and $\mathcal{R}\left(\mathbf{T}^{(i)}\right)$ the *right-unfolding* of $\mathbf{T}^{(i)}$, see [34]. A TT core is called *left-orthonormal* if its left-unfolding is orthonormal with respect to the columns, i.e.,

$$\left(\mathcal{L}\left(\mathbf{T}^{(i)}\right)\right)^T \cdot \mathcal{L}\left(\mathbf{T}^{(i)}\right) = \mathbf{T}^{(i)} \Big|_{r_i}^{r_{i-1}, n_i} \cdot \mathbf{T}^{(i)} \Big|_{r_{i-1}, n_i}^{r_i} = I \in \mathbb{R}^{r_i \times r_i},$$

and *right-orthonormal* if its right-unfolding is orthonormal with respect to the rows, i.e.,

$$\mathcal{R}\left(\mathbf{T}^{(i)}\right) \cdot \left(\mathcal{R}\left(\mathbf{T}^{(i)}\right)\right)^T = \mathbf{T}^{(i)} \Big|_{r_{i-1}}^{n_i, r_i} \cdot \mathbf{T}^{(i)} \Big|_{n_i, r_i}^{r_{i-1}} = I \in \mathbb{R}^{r_{i-1} \times r_{i-1}}.$$

Algorithms for the left- and right-orthonormalization, respectively, can be found in [30, 36]. Note that a tensor $\mathbf{T}$ remains the same if we apply an (exact) orthonormalization procedure to it. The algorithms simply compute a different but equivalent representation.

## 2.3 Pseudoinverses in the TT format

We now explain how to compute pseudoinverses of matricizations of tensors in the TT format. Later, we aim at applying this technique to the tensorized counterpart of the matrix $\Psi(\mathcal{X})$ defined in (1). It was shown that the pseudoinverse of certain tensor unfoldings can be computed directly in the TT format. We will briefly recapitulate the main idea, details can be found in [27]. Given a tensor $\mathbf{T} \in \mathbb{R}^{n_1 \times \dots \times n_d \times m}$ (e.g., a tensor containing $m$ snapshots of a $d$-dimensional system with dimensions $n_1, \dots, n_d$) in the TT format, we consider the matricization

$$T = \mathbf{T} \Big|_{n_1, \dots, n_d}^{m}. \tag{9}$$

The standard way to calculate the pseudoinverse $T^+$ is based on using its singular value decomposition (SVD), i.e., $T = U\Sigma V^T$ with $U^TU = V^TV = I$. The pseudoinverse of $T$ is then given by

$$T^+ = V\Sigma^{-1}U^T. \tag{10}$$

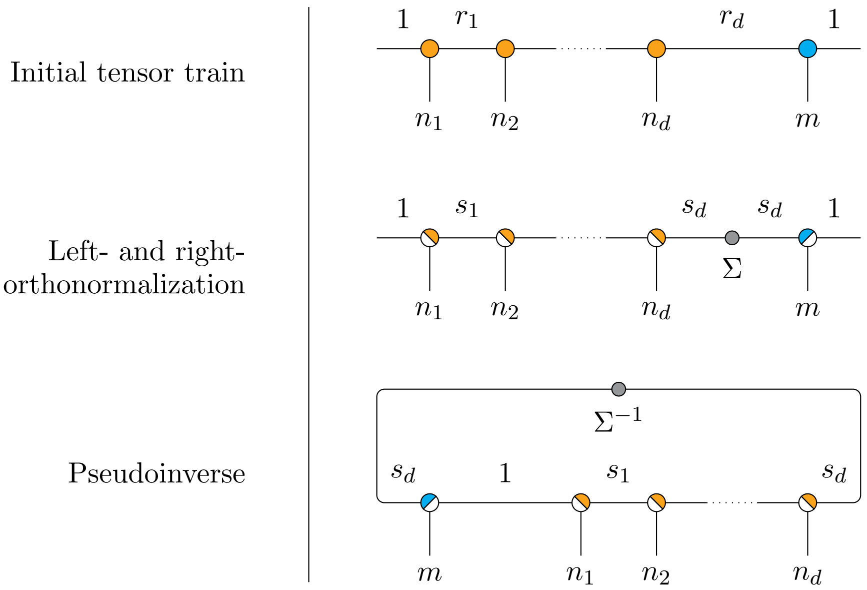

**Figure 4:** Computation of the pseudoinverse of a tensor train: After left- and right-orthonormalization of the initial tensor train (half-filled circles), the pseudoinverse can be represented as a cyclic tensor train with reordered cores. Note that we here depict the special case of the pseudoinversion of a tensor $\mathbf{T} \in \mathbb{R}^{n_1 \times \cdots \times n_d \times m}$, the general case is considered in [27].

In order to directly construct the pseudoinverse $T^+$ from a TT representation of $\mathbf{T}$, we apply the two orthonormalization procedures. That is, we left-orthonormalize the TT cores from $\mathbf{T}^{(1)}$ to $\mathbf{T}^{(d)}$ and right-orthonormalize the core $\mathbf{T}^{(d+1)}$. Doing this, we obtain a global SVD of the matricization $T$. Similar to (10), the pseudoinverse $T^+$ can then be obtained by reordering the cores, see Figure 4. A detailed description of the corresponding algorithms can be found in [27], where we have shown how to generalize the orthonormalization procedure to arbitrary matricizations of tensor trains. Algorithm 1 describes the pseudoinversion procedure for a tensor train $\mathbf{T} \in \mathbb{R}^{n_1 \times \cdots \times n_d \times m}$ in detail.

An important aspect is that we do not need to compute the pseudoinverse of $T$ explicitly. Instead, we only orthonormalize the TT cores and compute the matrix $\Sigma$ by executing the lines 1 to 5 of Algorithm 1. We then store the representation

$$\mathbf{T}^+ = \sum_{k_1=1}^{r_1} \cdots \sum_{k_d=1}^{r_d} \sigma_{k_d}^{-1} \cdot \mathbf{T}^{(d+1)}_{k_d,:,1} \otimes \mathbf{T}^{(1)}_{1,:,k_1} \otimes \cdots \otimes \mathbf{T}^{(d)}_{k_{d-1},:,k_d} \in \mathbb{R}^{m \times n_1 \times \cdots \times n_d},$$

which can be either regarded as the sum of $r_d$ tensor trains scaled by $\sigma_1^{-1}, \ldots, \sigma_{r_d}^{-1}$ or as a

**Algorithm 1** Pseudoinversion of tensor trains.

**Input:** Tensor train $\mathbf{T} \in \mathbb{R}^{n_1 \times \cdots \times n_d \times m}$.

**Output:** Pseudoinverse of $T = \mathbf{T}\Big|^{m}_{n_1,\dots,n_d}$.

```
1: Left-orthonormalize $\mathbf{T}^{(1)}, \dots, \mathbf{T}^{(d-1)}$ and right-orthonormalize $\mathbf{T}^{(d+1)}$.
2: Compute SVD of $\mathcal{L}\left(\mathbf{T}^{(d)}\right)$, i.e., $\mathcal{L}\left(\mathbf{T}^{(d)}\right) = U \Sigma V^T$ with $\Sigma \in \mathbb{R}^{s \times s}$.
3: Define $\mathbf{U} \in \mathbb{R}^{r_{d-1} \times n_d \times s}$ as a reshaped version of $U$ with $\mathbf{U}_{k,x,l} = U_{\overline{k,x},l}$.
4: Define $\mathbf{V} \in \mathbb{R}^{s \times m}$ by $\mathcal{R}(\mathbf{V}) = V^T \cdot \mathcal{R}\left(\mathbf{T}^{(d+1)}\right)$.
5: Set $\mathbf{T}^{(d)}$ to $\mathbf{U}$, $\mathbf{T}^{(d+1)}$ to $\mathbf{V}$, and $r_d$ to $s$.
6: Define $\tilde{U} = \left(\sum_{k_0=1}^{r_0} \cdots \sum_{k_{d-1}=1}^{r_{d-1}} \mathbf{T}^{(1)}_{k_0,:,k_1} \otimes \cdots \otimes \mathbf{T}^{(d)}_{k_{d-1},:,:}\right)\Big|^{r_d}_{n_1,\dots,n_d}$.
7: Define $\tilde{V} = \mathbf{T}^{(d+1)}\Big|^{r_d}_{m}$.
8: Define $T^+ = \tilde{V}\, \Sigma^{-1}\, \tilde{U}^T$.
```

cyclic tensor train [42] as depicted in Figure 4. It holds that

$$\mathbf{T}^+ \Big|^{n_1,\dots,n_d}_{m} = \left(\mathbf{T}\Big|^{m}_{n_1,\dots,n_d}\right)^+ .$$

For the orthonormalization procedures, we only consider compact/reduced SVDs, i.e., only the nonzero singular values and corresponding singular vectors are stored. It is also possible to use truncated SVDs within Algorithm 1, i.e., we discard all singular values $\sigma_k$ with $\sigma_k/\sigma_{\max} < \varepsilon$, where $\sigma_{\max}$ is the largest singular value and $\varepsilon$ a given threshold. In this way, we can reduce the computational costs and the storage consumption even further.

**Lemma 2.4** (Complexity of the pseudoinverse computation)**.** *The computational effort of Algorithm 1 can be estimated as $O(d \cdot n \cdot r^3 + m \cdot r^2)$, where $n$ is the maximum of the first $d$ modes and $r$ the maximum of all TT ranks.*

*Proof.* The overall computational costs of Algorithm 1 can be estimated by the number of applied SVDs. The complexity of calculating an SVD of a left-/right-unfolding can be estimated as $O(n \cdot r^3)$. Since we compute the left-orthonormalizations of the first $d$ cores and the right-orthonormalization of $\mathcal{R}(\mathbf{T}^{(d+1)}) \in \mathbb{R}^{r \times m}$, we obtain the estimation $O(d \cdot n \cdot r^3 + m \cdot r^2)$, assuming that $r \leq m$. □

## 3 Tensor-based reformulation of SINDy

After introducing SINDy and the TT format, we will now show how to combine the data-driven recovery of dynamical systems with tensor decompositions. We restrict ourselves to a set of basis functions $\mathbb{D} = \{\psi_1, \dots, \psi_p\}$, $\psi_j \colon \mathbb{R} \to \mathbb{R}$, and corresponding basis matrices whose entries are given by products of the given basis functions applied to the different dimensions of all snapshots. Note that the basis functions $\psi_j$, $j = 1, \dots, p$, are now defined on $\mathbb{R}$ and not on $\mathbb{R}^d$ as in Section 2.1. Exploiting the TT format for the construction of the tensorized counterpart of the basis matrix $\Psi(\mathcal{X})$, we are able to express large numbers of combinations of the basis functions $\psi_1, \dots, \psi_p$ in a compact way as tensor-products of the one-dimensional basis functions. By doing this, we can not only reduce the storage consumption of the basis matrix but may also lower the computational costs compared to the conventional SINDy-like implementation.

In what follows, we will describe different approaches for the tensor-based construction and show how to solve the minimization problem

$$\min_{\mathbf{\Xi}} \|\mathcal{Y} - \mathbf{\Xi}^T \mathbf{\Psi}(\mathcal{X})\|_F \tag{11}$$

directly in the TT format. Here, $\mathbf{\Xi}$ and $\mathbf{\Psi}(\mathcal{X})$ denote the tensorized counterparts of the matrices $\Xi$ and $\Psi(\mathcal{X})$ introduced in Section 2. As mentioned in the introduction, we call our method MANDy.

## 3.1 Basis decomposition

We will consider two different types of basis decompositions. Let $X = [x_1, \dots, x_d]^T \in \mathbb{R}^d$ be a vector and $\psi_j : \mathbb{R} \to \mathbb{R}$, $j = 1, \dots, p$, basis functions. We consider the two rank-one decompositions

$$\mathbf{\Psi}_{\mathrm{cm}}(X) = \mathbf{\Psi}_{\mathrm{cm}}^{(1)}(X) \otimes \dots \otimes \mathbf{\Psi}_{\mathrm{cm}}^{(d)}(X) = \begin{bmatrix} \psi_1(x_1) \\ \vdots \\ \psi_p(x_1) \end{bmatrix} \otimes \dots \otimes \begin{bmatrix} \psi_1(x_d) \\ \vdots \\ \psi_p(x_d) \end{bmatrix} \in \mathbb{R}^{p \times \dots \times p} \tag{12}$$

and

$$\mathbf{\Psi}_{\mathrm{fm}}(X) = \mathbf{\Psi}_{\mathrm{fm}}^{(1)}(X) \otimes \dots \otimes \mathbf{\Psi}_{\mathrm{fm}}^{(p)}(X) = \begin{bmatrix} \psi_1(x_1) \\ \vdots \\ \psi_1(x_d) \end{bmatrix} \otimes \dots \otimes \begin{bmatrix} \psi_p(x_1) \\ \vdots \\ \psi_p(x_d) \end{bmatrix} \in \mathbb{R}^{d \times \dots \times d}. \tag{13}$$

That is, for (12) we apply all basis functions to one specific element of the vector $X$ in each core while we apply only one basis function to all elements of $X$ in each core of (13). Analogously to the row and column major order of multidimensional arrays, we call $\mathbf{\Psi}_{\mathrm{cm}}$ a *coordinate-major decomposition* and $\mathbf{\Psi}_{\mathrm{fm}}$ a *function-major decomposition*. It holds that

$$(\mathbf{\Psi}_{\mathrm{cm}}(X))_{j_1,\dots,j_d} = \psi_{j_1}(x_1) \cdot \ldots \cdot \psi_{j_d}(x_d) \quad \text{and} \quad (\mathbf{\Psi}_{\mathrm{fm}}(X))_{j_1,\dots,j_p} = \psi_1(x_{j_1}) \cdot \ldots \cdot \psi_p(x_{j_p}).$$

Using the decompositions (12) and (13), we can express the tensors $\mathbf{\Psi}_{\mathrm{cm}}(X)$ and $\mathbf{\Psi}_{\mathrm{fm}}(X)$ in a storage-efficient way. The memory consumption of the full representations of the tensors is $O(p^d)$ and $O(d^p)$, respectively, whereas the memory consumption of both rank-one representations is $O(p \cdot d)$.

**Remark 3.1** (Choice of rank-one decompositions)**.** What we present here are only two types of rank-one decompositions for a set of basis functions. In fact, other decompositions are also possible and specific problems might necessitate more complex representations in the future. However, we will focus on decompositions of the form (12) or (13).

## 3.2 Multidimensional approximation of nonlinear dynamical systems

For $m$ different vectors $X_k = [x_{k,1}, \dots, x_{k,d}]^T \in \mathbb{R}^d$, $k = 1, \dots, m$, stored in a matrix $\mathcal{X}$ (see Section 2.1), we aim at using the decompositions (12) and (13) to construct counterparts of the matrices

$$\Psi_{\mathrm{cm}}(\mathcal{X}) = \begin{bmatrix} \Psi_{\mathrm{cm}}(X_1) & \Psi_{\mathrm{cm}}(X_2) & \dots & \Psi_{\mathrm{cm}}(X_m) \end{bmatrix} \in \mathbb{R}^{p^d \times m} \tag{14}$$

and

$$\Psi_{\mathrm{fm}}(\mathcal{X}) = \begin{bmatrix} \Psi_{\mathrm{fm}}(X_1) & \Psi_{\mathrm{fm}}(X_2) & \dots & \Psi_{\mathrm{fm}}(X_m) \end{bmatrix} \in \mathbb{R}^{d^p \times m} \tag{15}$$

directly in the TT format. Here, $\Psi_{\mathrm{cm}}(X_k) \in \mathbb{R}^{p^d}$ and $\Psi_{\mathrm{fm}}(X_k) \in \mathbb{R}^{d^p}$ denote the vectorizations $\mathrm{vec}\left(\boldsymbol{\Psi}_{\mathrm{cm}}(X_k)\right)$ and $\mathrm{vec}\left(\boldsymbol{\Psi}_{\mathrm{fm}}(X_k)\right)$, respectively. Collecting the rank-one decompositions given in (12) for all vectors $X_1, \dots, X_m$ in a single TT decomposition, we obtain

$$\begin{aligned}
\boldsymbol{\Psi}_{\mathrm{cm}}(\mathcal{X}) &= \left[\!\left[ \boldsymbol{\Psi}^{(1)}_{\mathrm{cm}}(\mathcal{X}) \right]\!\right] \otimes \left[\!\left[ \boldsymbol{\Psi}^{(2)}_{\mathrm{cm}}(\mathcal{X}) \right]\!\right] \otimes \dots \otimes \left[\!\left[ \boldsymbol{\Psi}^{(d)}_{\mathrm{cm}}(\mathcal{X}) \right]\!\right] \otimes \left[\!\left[ \boldsymbol{\Psi}^{(d+1)}_{\mathrm{cm}}(\mathcal{X}) \right]\!\right] \\
&= \left[\!\left[ \begin{matrix} \boldsymbol{\Psi}^{(1)}_{\mathrm{cm}}(X_1) & \cdots & \boldsymbol{\Psi}^{(1)}_{\mathrm{cm}}(X_m) \end{matrix} \right]\!\right] \otimes \left[\!\left[ \begin{matrix} \boldsymbol{\Psi}^{(2)}_{\mathrm{cm}}(X_1) & & 0 \\ & \ddots & \\ 0 & & \boldsymbol{\Psi}^{(2)}_{\mathrm{cm}}(X_m) \end{matrix} \right]\!\right] \otimes \cdots \\
&\quad \cdots \otimes \left[\!\left[ \begin{matrix} \boldsymbol{\Psi}^{(d)}_{\mathrm{cm}}(X_1) & & 0 \\ & \ddots & \\ 0 & & \boldsymbol{\Psi}^{(d)}_{\mathrm{cm}}(X_m) \end{matrix} \right]\!\right] \otimes \left[\!\left[ \begin{matrix} e_1 \\ \vdots \\ e_m \end{matrix} \right]\!\right] \in \mathbb{R}^{p \times \dots \times p \times m},
\end{aligned} \tag{16}$$

where $e_k$, $k = 1, \dots, m$, denote the unit vectors of the standard basis in the $m$-dimensional Euclidean space. Analogously, we can write

$$\begin{aligned}
\boldsymbol{\Psi}_{\mathrm{fm}}(\mathcal{X}) &= \left[\!\left[ \boldsymbol{\Psi}^{(1)}_{\mathrm{fm}}(\mathcal{X}) \right]\!\right] \otimes \left[\!\left[ \boldsymbol{\Psi}^{(2)}_{\mathrm{fm}}(\mathcal{X}) \right]\!\right] \otimes \dots \otimes \left[\!\left[ \boldsymbol{\Psi}^{(p)}_{\mathrm{fm}}(\mathcal{X}) \right]\!\right] \otimes \left[\!\left[ \boldsymbol{\Psi}^{(p+1)}_{\mathrm{fm}}(\mathcal{X}) \right]\!\right] \\
&= \left[\!\left[ \begin{matrix} \boldsymbol{\Psi}^{(1)}_{\mathrm{fm}}(X_1) & \cdots & \boldsymbol{\Psi}^{(1)}_{\mathrm{fm}}(X_m) \end{matrix} \right]\!\right] \otimes \left[\!\left[ \begin{matrix} \boldsymbol{\Psi}^{(2)}_{\mathrm{fm}}(X_1) & & 0 \\ & \ddots & \\ 0 & & \boldsymbol{\Psi}^{(2)}_{\mathrm{fm}}(X_m) \end{matrix} \right]\!\right] \otimes \cdots \\
&\quad \cdots \otimes \left[\!\left[ \begin{matrix} \boldsymbol{\Psi}^{(p)}_{\mathrm{fm}}(X_1) & & 0 \\ & \ddots & \\ 0 & & \boldsymbol{\Psi}^{(p)}_{\mathrm{fm}}(X_m) \end{matrix} \right]\!\right] \otimes \left[\!\left[ \begin{matrix} e_1 \\ \vdots \\ e_m \end{matrix} \right]\!\right] \in \mathbb{R}^{d \times \dots \times d \times m}.
\end{aligned} \tag{17}$$

Both TT decompositions (16) and (17) have a TT rank of $m$ and it holds that

$$\boldsymbol{\Psi}_{\mathrm{cm}}(\mathcal{X}) \Big|^{m}_{p,\dots,p} = \Psi_{\mathrm{cm}}(\mathcal{X}) \quad \text{and} \quad \boldsymbol{\Psi}_{\mathrm{fm}}(\mathcal{X}) \Big|^{m}_{d,\dots,d} = \Psi_{\mathrm{fm}}(\mathcal{X}).$$

We could also express the basis tensors in the canonical format, but using the TT format enables us to construct the pseudoinverse of $\boldsymbol{\Psi}_{\mathrm{cm}}(\mathcal{X})$ and $\boldsymbol{\Psi}_{\mathrm{fm}}(\mathcal{X})$ directly as a TT decomposition, see Section 2.3. That is, we solve the minimization problem (11) by computing the pseudoinverse of $\boldsymbol{\Psi}_{\mathrm{cm}}(\mathcal{X})$ or $\boldsymbol{\Psi}_{\mathrm{fm}}(\mathcal{X})$, respectively, in the TT format and obtain

$$\boldsymbol{\Xi}^T = \mathcal{Y} \cdot \boldsymbol{\Psi}_{\mathrm{cm/fm}}(\mathcal{X})^+ \tag{18}$$

with

$$\boldsymbol{\Xi}^T \Big|_{d}^{p/d,\dots,p/d} = \mathcal{Y} \cdot \boldsymbol{\Psi}_{\text{cm/fm}}(\mathcal{X})^+ \Big|_{m}^{p/d,\dots,p/d} = \mathcal{Y} \cdot \Psi_{\text{cm/fm}}(\mathcal{X})^+.$$

For a detailed description of the tensor contraction in (18), see Figure 5.

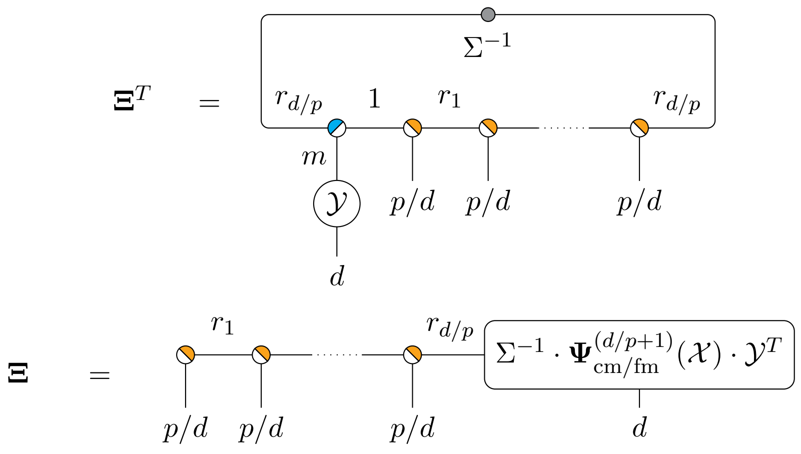

**Figure 5:** Construction of the coefficient tensor $\boldsymbol{\Xi}$: After the left-orthonormalization of the first $d$ or $p$ cores, respectively, of $\boldsymbol{\Psi}_{\text{cm/fm}}(\mathcal{X})$, the tensor $\boldsymbol{\Xi}$ can be obtained by contracting the pseudoinverse with the data matrix $\mathcal{Y}$. Since $\boldsymbol{\Xi}^T$ is a cyclic tensor train, the tensor $\boldsymbol{\Xi}$ can be expressed as a standard tensor train where we only replace the last core.

Storing the TT cores of (16) and (17) in a sparse format, the memory consumption of both representations can be estimated as $O(p \cdot d \cdot m)$. The memory consumption of the corresponding matricizations would be $O(p^d \cdot m)$ and $O(d^p \cdot m)$, respectively. This is the first main advantage of the proposed decompositions: If the number of snapshots is not too large ($m \ll p^d$ or $m \ll d^p$), we are able to efficiently store all entries of the matrices (14) and (15) in the TT decompositions (16) and (17), respectively. The second advantage is then the fast computation of the pseudoinverse. The computational effort needed to compute an SVD of the matrices (14) and (15) and to construct the pseudoinverse given in (10) can be estimated as $O(p^d \cdot m^2)$ and $O(d^p \cdot m^2)$, respectively. In contrast to that, we only need $O(p \cdot d \cdot m^3)$ steps to compute the pseudoinverse of the TT representations (16) and (17), see Lemma 2.4. On the other hand, if $m \gtrsim p^d$ (or $m \gtrsim d^p$), then the tensor-train based approach is computationally more expensive than the classical methods. However, we still benefit from the reduced storage consumption of the basis tensor using MANDy.

**Remark 3.2** (Right-orthonormality of the last core)**.** Note that it is not necessary to right-orthonormalize the last core of $\boldsymbol{\Psi}_{\text{cm/fm}}(\mathcal{X})$ for the construction of its pseudoinverse as shown in Algorithm 1 since the last core is already right-orthonormal by construction, cf. (16) and (17).

**Remark 3.3** (Special case of monomial basis functions)**.** For the special case of monomial basis functions, i.e., for a dictionary given by $\mathbb{D} = \{\psi_1, \dots, \psi_p\} = \{1, x, x^2, \dots, x^{p-1}\}$ with $p = 2^q$, we can decompose the components of the coordinate-major rank-one tensor given in (12) even further by writing

$$\begin{bmatrix} \psi_1(x_i) \\ \psi_2(x_i) \\ \vdots \\ \psi_p(x_i) \end{bmatrix} = \begin{bmatrix} 1 \\ x_i \\ \vdots \\ x_i^{2^q-1} \end{bmatrix} \hat{=} \begin{bmatrix} 1 \\ x_i \end{bmatrix} \otimes \begin{bmatrix} 1 \\ x_i^2 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ x_i^4 \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} 1 \\ x_i^{2^{q-1}} \end{bmatrix},$$

which can be easily shown using $\sum_{l=0}^{q-1} 2^l = 2^q - 1$. We will not make use of this kind of sub-decomposition in this work. Nevertheless, it may be advantageous for the identification of governing equations involving higher-order monomials.

**Remark 3.4** (Constraints on the coefficients)**.** Even when using the TT format for the construction of the tensors $\boldsymbol{\Psi}_{\text{cm}}(\mathcal{X})$ and $\boldsymbol{\Psi}_{\text{fm}}(\mathcal{X})$, respectively, it is possible to directly include linear (dimensionwise) constraints on the coefficients stored in $\boldsymbol{\Xi}$. A tensor train that encodes the extra conditions on the coefficients can be seen as an additional snapshot. Together with a corresponding vector appended to $\mathcal{Y}$, we can incorporate the (weighted) contraints.

## 4 Numerical results

In this section, we will present three numerical examples for the application of MANDy, namely Chua's circuit (which was already introduced in Example 2.1), the Fermi–Pasta–Ulam–Tsingou problem [48], and the Kuramoto model [49]. The numerical experiments have been performed on a Linux machine with 128 GB RAM and an Intel Xeon processor with a clock speed of 3 GHz and 8 cores. The algorithms have been implemented in Python 3.6 and collected in the toolbox Scikit-TT available on GitHub: https://github.com/PGelss/scikit_tt.

### 4.1 Chua's circuit

As a first experiment, let us consider our example from Section 2.1 again with the intent to illustrate how using tensor products of simple functions allows for the construction of rich sets of basis functions. The first set of basis functions (3) used in Example 2.1 corresponds to the coordinate-major rank-one decomposition

$$\boldsymbol{\Psi}_1(X) = \begin{bmatrix} 1 \\ x_1 \\ x_1^2 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ x_2 \\ x_2^2 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ x_3 \\ x_3^2 \end{bmatrix} \in \mathbb{R}^{3\times3\times3}.$$

As we have seen, these ansatz functions do not lead to a correct recovery of the system dynamics. Instead we have to use a basis set including the absolute values of the coordinates. The basis vector given in (4) then corresponds to the function-major rank-one decomposition

$$\boldsymbol{\Psi}_2(X) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ |x_1| \\ |x_2| \\ |x_3| \end{bmatrix} \in \mathbb{R}^{4\times4}. \tag{19}$$

As for the matrix-based approach in Example 2.1, the approximate coefficient tensor is numerically equal to the exact coefficient tensor (see Appendix A.1) when we apply MANDy with the basis decomposition $\mathbf{\Psi}_2$ and choose the same parameters and number of snapshots. This small example already shows the advantage of using tensor decompositions in terms of storage consumption. For the sparse TT representation of the basis tensor (see (13) and (17)), we only have to store 18000 entries whereas the matricized counterpart would have 32000 entries.

### 4.2 Fermi–Pasta–Ulam–Tsingou problem

Let us now consider the Fermi–Pasta–Ulam–Tsingou model, which was introduced by the eponymous physicists and mathematicians in 1953. The underlying model represents a vibrating string by a system of coupled oscillators fixed at the respective end points of the string.

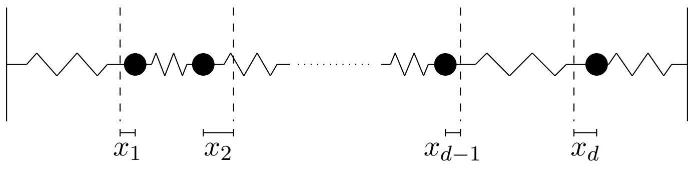

**Figure 6:** Fermi–Pasta–Ulam–Tsingou model: Representation of a vibrating string by a set of masses coupled by springs.

Here, we consider a dynamical system described by a second-order differential equation $\ddot{X}(t) = F(X(t))$ where the right-hand side does not depend on $\dot{X}$. A very large number of problems in astronomy, molecular dynamics, and other areas of physics are of this form, as this type of differential equation is an immediate consequence of Newtonian mechanical laws. Moreover, it makes no difference for our methods whether we have given data measurements $[X_1, \dots, X_m]$ and $[\dot{X}_1, \dots, \dot{X}_m]$ or $[X_1, \dots, X_m]$ and $[\ddot{X}_1, \dots, \ddot{X}_m]$. Let us consider the model variant with cubic forcing terms, which is of the form

$$\ddot{x}_i = (x_{i+1} - 2x_i + x_{i-1}) + \beta\left[(x_{i+1} - x_i)^3 - (x_i - x_{i-1})^3\right], \tag{20}$$

$i = 1, \dots, d$, where $x_i$ represents the displacement of the $i$th oscillator from its original position, see Figure 6. We assume the ends of the chain to be fixed, i.e., $x_0 = x_{d+1} = 0$. The parameter $\beta \in \mathbb{R}$ represents the nonlinear force between the oscillators.

For our first numerical experiment, we consider $d = 10$ oscillators. We compare our proposed method with the matrix-based solution of the least-squares problem given in (11). For this purpose, we choose random displacements in $[-0.1, 0.1]$ for each oscillator and compute the (exact) second derivatives using (20) with $\beta = 0.7$. As basis functions we choose $\mathbb{D} = \{1, x, x^2, x^3\}$. This then results in the representation of $4^{10}$ combinations of function evaluations for every snapshot if we use the coordinate-major ansatz, see (12), such that

$$\mathbf{\Psi}_{\mathrm{cm}}(\mathcal{X}) \in \mathbb{R}^{4\times\cdots\times 4\times m} \quad \text{for} \quad \mathcal{X} = \begin{bmatrix} X_1 & X_2 & \dots & X_m \end{bmatrix} \in \mathbb{R}^{10\times m},$$

where $m$ is the number of considered snapshots.

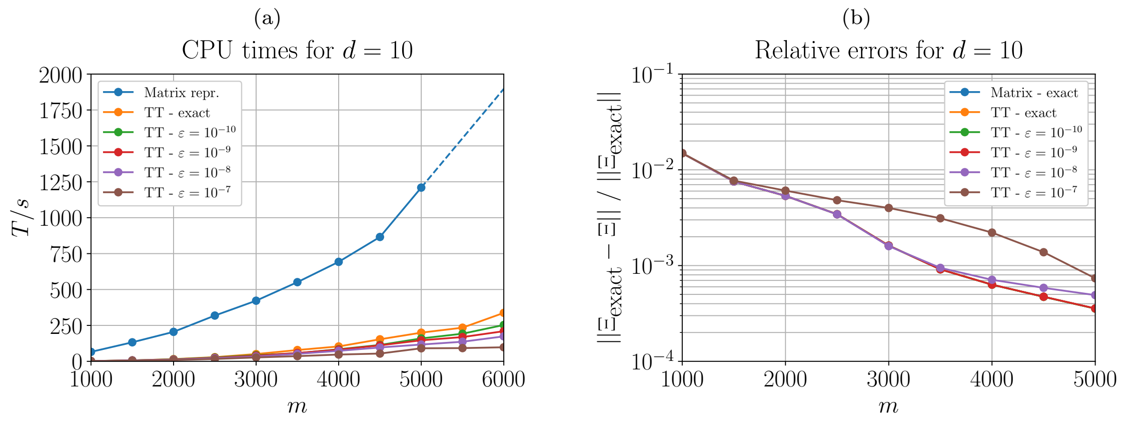

**Figure 7:** Application of MANDy to the Fermi–Pasta–Ulam–Tsingou problem: (a) CPU times needed for the computation of the coefficient tensor $\mathbf{\Xi}$ and the matricized counterpart $\Xi$, respectively. For $m = 6000$, the CPU time of the matrix-based approach is extrapolated since storing the matrices for calculations with $m > 5000$ would require too much memory. (b) Relative errors between the approximate solutions and the exact solution. Results are shown for the tensor-based approach (using different thresholds for orthonormalizations) as well as for the matrix-based approach. For $\varepsilon \leq 10^{-9}$, the relative errors of MANDy are numerically indistinguishable from those of the SINDy-like approach.

Figure 7 shows the CPU times and relative errors of MANDy for varying $m$. We see that we are able to reduce the time needed for computing the coefficient tensor/matrix significantly – at least within the considered range of snapshot numbers with $m \ll 4^{10}$. For MANDy, we also included the construction of the tensor train $\mathbf{\Psi}_{\text{cm}}(\mathcal{X})$ in the CPU times, whereas the runtimes for the matrix-based approach only consist of the times needed to solve the minimization problem. Without using truncated SVDs for the orthonormalization procedures, we obtain a speed-up of approximately up to five. The runtimes decrease even further when we set a threshold $\varepsilon > 0$ for the SVDs.

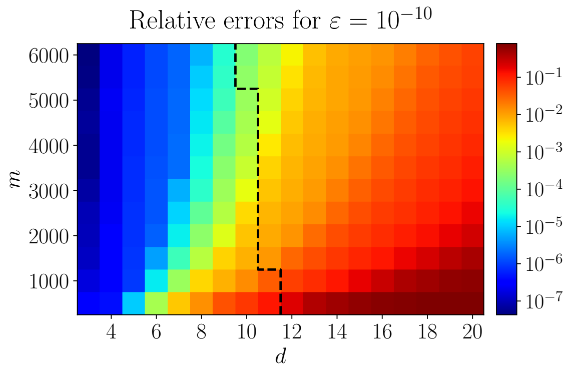

**Figure 8:** Relative errors for different parameters: The relative errors between the approximate and the exact solutions for the coefficient tensor depending on $d$ and $m$ are shown. All parameter combinations for $m$ and $d$ to the right of the dashed line could only be handled using the tensor-based method MANDy.

At the same time, we obtain nearly the same relative errors between the approximate and the exact solution, cf. Appendix A.2. As shown in Figure 7 (b), there is in fact no noticeable difference between the error produced by the matrix-based approach and the tensor-based approach with thresholds of $\varepsilon \leq 10^{-9}$. Moreover, we can consider higher numbers of snapshots and dimensions using MANDy. This is shown more clearly in Figure 8, where we plot the relative errors in dependence of $m$ and $d$. In particular, we were not able to compute a matrix-based solution using standard SINDy for $d \geq 12$ since the basis matrix $\Psi_{\text{cm}}(\mathcal{X})$ simply becomes too large. Using MANDy, we can even approximate the solution with a relative error smaller than $10^{-1}$ for $d = 20$ and $m = 6000$ (which implies a basis tensor $\boldsymbol{\Psi}_{\text{cm}}(\mathcal{X})$ with $4^{20} \cdot 6000 \approx 6.6 \cdot 10^{15}$ elements).

## 4.3 Kuramoto model

As a third example for the application of MANDy, we consider the Kuramoto model. The model describes the behavior of a large number of coupled oscillators on a circle. First introduced by Yoshiki Kuramoto in 1975, see [50], it has been studied extensively over the last decades.

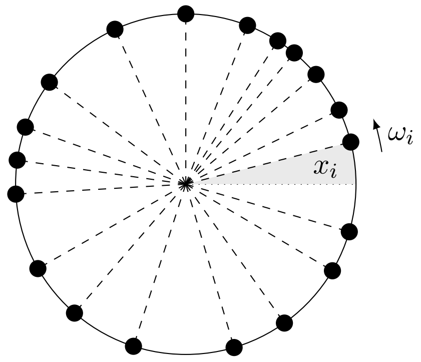

**Figure 9:** Kuramoto model: Simulation of coupled oscillators on a ring.

The governing equations – including an external forcing, see [49] – can be written as

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = \omega_i + \frac{K}{d} \sum_{j=1}^{d} \sin(x_j - x_i) + h \sin(x_i), \quad i = 1, \ldots, d, \tag{21}$$

where $d$ is the number of oscillators, $K$ the coupling strength, $h$ the external forcing parameter, $\omega_i$ the $i$th natural frequency, and $x_i$ the angular position of the $i$th oscillator, see Figure 9. Here, we assume an identical coupling between all oscillators as well as a constant forcing parameter. In detail, we consider $d = 100$ oscillators and set $K = 2$ and $h = 0.2$. Furthermore, we choose intrinsic frequencies equidistantly distributed in $[-5, 5]$.

Now, we intend to identify the governing equations from simulation data. We choose the set of basis functions given by $\mathbb{D} = \{\sin(x), \cos(x)\}$ and use the function-major decomposition (13) which leads to tensors of the form

$$\mathbf{\Psi}_{\mathrm{fm}}(X) = \begin{bmatrix} 1 \\ \sin(x_1) \\ \vdots \\ \sin(x_d) \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \cos(x_1) \\ \vdots \\ \cos(x_d) \end{bmatrix} \in \mathbb{R}^{d+1 \times d+1}. \tag{22}$$

Similar to the dictionary used for Chua's circuit, see Section 4.1, we added the basis function 1 to both cores in order to ensure that also functions of the form $\sin(x_i)$ and $\cos(x_i)$ appear in the large basis set. Note that $\sin(x \pm y) = \sin(x)\cos(y) \pm \cos(x)\sin(y)$ so that the system can indeed be represented by the chosen basis. We simulate the Kuramoto model from $t_0 = 0$ to $t_1 = 1020$ using an implementation of the BDF method as described in [51] and take 10 snapshots within each second of simulation time, i.e., the time points corresponding to the snapshots are $\{0, 0.1, 0.2, \dots, 1019.9, 1020\}$. That is, we consider 10201 data points which would for the matrix case mean that the matrix $\Psi_{\mathrm{fm}}(\mathcal{X})$ is square. As the initial distribution, we randomly place the oscillators on the ring. Then, based on the obtained data points $X_1, \dots, X_{10201}$ and corresponding derivatives $\dot{X}_1, \dots, \dot{X}_{10201}$, we recover the dynamics using MANDy with a threshold of $\varepsilon = 10^{-16}$. As in the previous examples, we are able to compare our result with the exact solution given in Appendix A.3. We repeated the experiment several times and never observed relative errors larger than $10^{-4}$. Applying our recovered dynamics to another randomly chosen initial distribution of the oscillators, we see that we are able to approximate the dynamics of the Kuramoto model accurately within a time interval of nearly up to 100 seconds, see Figure 10. In terms of matrix representations, the reason for the inaccuracy is that the matrix $\Psi_{\mathrm{fm}}(\mathcal{X})$ does not have full rank.

Even if the computational overhead is bigger using MANDy compared to the classical method, we are able to reduce the storage consumption by a factor of more than 50. Here, a sparse representation of the tensor cores of $\mathbf{\Psi}_{\mathrm{fm}}(\mathcal{X})$ enables us to significantly reduce the memory consumption.

## 5 Conclusion and outlook

### 5.1 Conclusion

In this work, we have proposed an approach for the data-driven recovery of governing equations of nonlinear dynamical systems. The presented approach – called MANDy, short for *multidimensional approximation of nonlinear dynamical systems* – combines data-driven methods with tensor-train decompositions. The aim of this approach is to reduce the memory consumption as well as the computational costs significantly and to mitigate the curse of dimensionality. After explaining the SINDy method for identifying governing equations based on a given basis set as well as the TT format, we have first shown how to use tensor products of simple functions for the construction of rich sets of basis functions. We have then explained how to rewrite the data-driven recovery in terms of tensor decompositions and the tensor-based computation of pseudoinverses. The results, which have been illustrated with several examples of dynamical systems, clearly demonstrate that the proposed algorithm needs less computational costs than the classical method if the number of data points is small enough. However, even if a large number of snapshots is needed for the

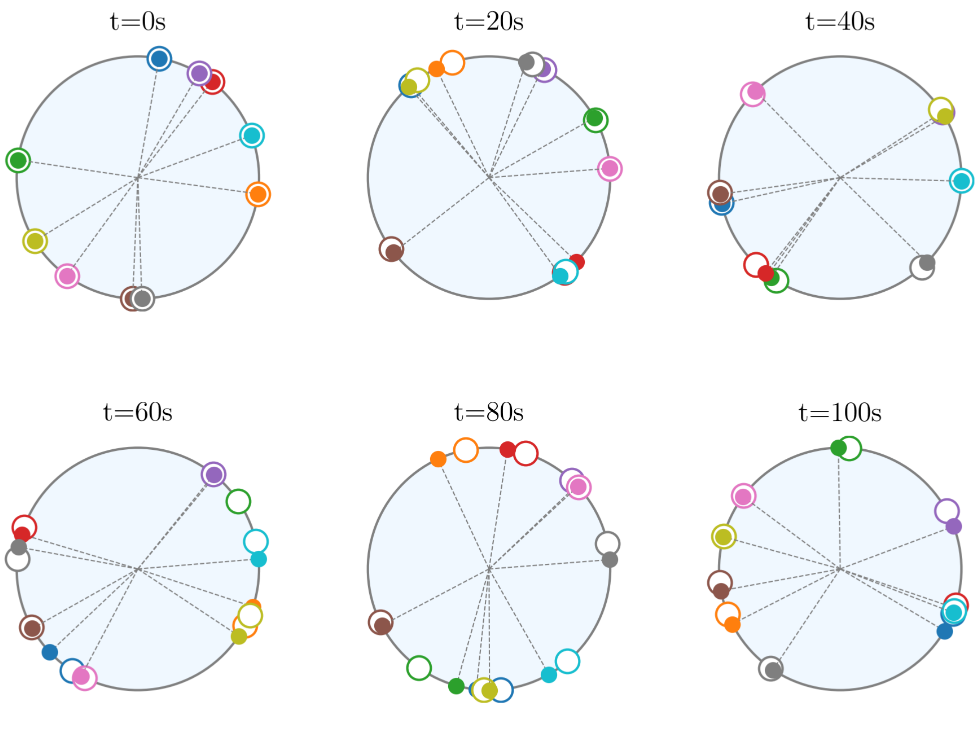

**Figure 10:** Application of MANDy to the Kuramoto model: The recovered dynamics are applied to another random initial distribution of the oscillators and then compared with the exact dynamics. Here, we only visualize the oscillators with indices $i = 1, 11, \dots, 91$. The solid dots represent the positions according to the approximated dynamics whereas the circles with the same color represent the positions according to the real dynamics.

recovery of the governing equations, we still benefit from the reduction of the memory consumption. The work presented in this paper constitutes a first step towards tensor-based techniques for the reconstruction of unknown systems purely from data measurements.

## 5.2 Outlook

The investigations carried out in this work offer a number of further promising considerations. The above numerical experiments show convincingly that one can expect a significant improvement of MANDy over SINDy, not only concerning the memory requirements used in the recovery, but also in the computational effort. This advantage originates from exploiting the underlying data structure by using low-rank tensor decompositions. Key to this is an understanding of the correlations in the matrices that are approximated in the TT format. Appendix A.4 provides some details on the correlation structure required to obtain accurate TT approximations, requiring further research on the local correlation structure.

Other future research will include the investigation of perturbations of the data and their influence on the reconstructed systems. We have observed that MANDy (as well as methods like SINDy) is highly sensitive to noisy data, even if we only add a slight Gaussian noise to the measurements. In this context, the inclusion of approximate derivatives for the recovery will also be of interest. A common way of adding noise is to take data generated by $\dot{X}(t) = F(X(t)) + \xi$, where $\xi \in \mathbb{R}^d$ is a vector capturing errors, either drawn i.i.d. reflecting Gaussian noise, or generally with $\|\xi\|_F < \eta$ for some constant $\eta > 0$. One might even hope for proving recovery guarantees as they are common in the context of compressed sensing [52, 53], given certain suitably structured data and noise levels.

Moreover, the construction of elaborate sets of basis functions has to be examined more closely in the future. Here, we used a priori knowledge for the construction of the dictionary, which is generally not known in practice. Additionally, parts of this work – in particular the proposed basis decompositions – may be used for the development of other tensor-based counterparts of methods such as EDMD or kernel EDMD. Accepting that the improved performance originates from exploiting the correlation structure, one can hope that other tensor networks are suitable for other kinds of data. Hierarchical tensor formats [15, 16, 54] and entangled project pair states [19, 20] may be suitable as well. The hope is that the present work stimulates such endeavors.

## Acknowledgements

This research has been partially funded by Deutsche Forschungsgemeinschaft (DFG) through grant CRC 1114 and EI 519/9-1, as well as the Templeton Foundation and the European Research Council (ERC).

## References

[1] J. C. Sprott. Some simple chaotic flows. *Physical Review E*, 50:R647–R650, 1994.

[2] J. E. S. Socolar. Nonlinear dynamical systems. In T. S. Deisboeck and J. Y. Kresh, editors, *Complex Systems Science in Biomedicine*, pages 115–140. Springer US, 2006.

[3] G. Teschl. *Ordinary Differential Equations and Dynamical Systems*, volume 140 of *Graduate Studies in Mathematics*. American Mathematical Society, 2012.

[4] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113:3932–3937, 2016.

[5] L. Boninsegna, F. Nüske, and C. Clementi. Sparse learning of stochastic dynamical equations. *The Journal of Chemical Physics*, 148(24):241723, 2018.

[6] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P.-Y. Glorennec, H. Hjalmarsson, and A. Juditsky. Nonlinear black-box modeling in system identification: A unified overview. *Automatica*, 31(12):1691–1724, 1995.

[7] J. R. Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, 4(2):87–112, 1994.

[8] H. Muzhou and H. Xuli. The multidimensional function approximation based on constructive wavelet RBF neural network. *Applied Soft Computing*, 11(2):2173–2177, 2011.

[9] T. Blumensath. Accelerated iterative hard thresholding. *Signal Processing*, 92(3):752–756, 2012.

[10] R. Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[11] J. D. Carroll and J. J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of 'Eckart-Young' decomposition. *Psychometrika*, 35(3):283–319, 1970.

[12] R. A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970.

[13] L. R. Tucker. Implications of factor analysis of three-way matrices for measurement of change. In C. W. Harris, editor, *Problems in measuring change*, pages 122–137. University of Wisconsin Press, 1963.

[14] L. R. Tucker. The extension of factor analysis to three-dimensional matrices. In H. Gulliksen and N. Frederiksen, editors, *Contributions to mathematical psychology*, pages 110–127. Holt, Rinehart and Winston, 1964.

[15] W. Hackbusch and S. Kühn. A new scheme for the tensor representation. *The Journal of Fourier Analysis and Applications*, 15(5):706–722, 2009.

[16] A. Arnold and T. Jahnke. On the approximation of high-dimensional differential equations in the hierarchical Tucker format. *BIT Numerical Mathematics*, 54(2):305–341, 2013.

[17] S. R. White. Density matrix formulation for quantum renormalization groups. *Physical Review Letters*, 69(19):2863–2866, 1992.

[18] H. D. Meyer, F. Gatti, and G. A. Worth (eds.). *Multidimensional quantum dynamics: MCTDH theory and applications*. Wiley-VCH Verlag GmbH & Co. KGaA, 2009.

[19] R. Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics*, 349:117–158, 2014.

[20] F. Verstraete, J. I. Cirac, and V. Murg. Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. *Advances in Physics*, 57:143, 2008.

[21] J. Eisert. Entanglement and tensor network states. *Modelling and Simulation*, 3:520, 2013.

[22] T. Jahnke and W. Huisinga. A dynamical low-rank approach to the chemical master equation. *Bulletin of Mathematical Biology*, 70(8):2283–2302, 2008.

[23] P. Gelß, S. Matera, and C. Schütte. Solving the master equation without kinetic Monte Carlo: Tensor train approximations for a CO oxidation model. *Journal of Computational Physics*, 314:489–502, 2016.

[24] G. Beylkin, J. Garcke, and M. J. Mohlenkamp. Multivariate regression and machine learning with sums of separable functions. *SIAM Journal on Scientific Computing*, 31(3):1840–1857, 2009.

[25] A. Novikov, D. Podoprikhin, A. Osokin, and D. Vetrov. Tensorizing neural networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28 (NIPS)*, pages 442–450. Curran Associates, Inc., 2015.

[26] S. Klus and C. Schütte. Towards tensor-based methods for the numerical approximation of the Perron–Frobenius and Koopman operator. *Journal of Computational Dynamics*, 3(2), 2016.

[27] S. Klus, P. Gelß, S. Peitz, and C. Schütte. Tensor-based dynamic mode decomposition. *Nonlinearity*, 31(7), 2018.

[28] I. V. Oseledets. A new tensor decomposition. *Doklady Mathematics*, 80(1):495–496, 2009.

[29] I. V. Oseledets and E. E. Tyrtyshnikov. Breaking the curse of dimensionality, or how to use SVD in many dimensions. *SIAM Journal on Scientific Computing*, 31(5):3744–3759, 2009.

[30] I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[31] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac. Matrix product state representations. *Quantum Information and Computation*, 5&6:401, 2006.

[32] U. Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326:96, 2011.

[33] M. Fannes, B. Nachtergaele, and R.F. Werner. Finitely correlated states on quantum spin chains. *Communications in Mathematical Physics*, 144(3):443–490, 1992.

[34] S. Holtz, T. Rohwedder, and R. Schneider. The alternating linear scheme for tensor optimization in the tensor train format. *SIAM Journal on Scientific Computing*, 34(2):A683–A713, 2012.

[35] L. Grasedyck, D. Kressner, and C. Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1):53–78, 2013.

[36] P. Gelß. *The Tensor-Train Format and Its Applications: Modeling and Analysis of Chemical Reaction Networks, Catalytic Processes, Fluid Flows, and Brownian Dynamics.* dissertation, Freie Universität Berlin, 2017.

[37] H. Rauhut, R. Schneider, and Z. Stojanac. Tensor completion in hierarchical tensor representations. In H. Boche, R. Calderbank, G. Kutyniok, and J. Vybíral, editors, *Compressed Sensing and its Applications: MATHEON Workshop 2013*, pages 419–450. Springer International Publishing, 2015.

[38] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE Signal Processing Magazine*, 32(2):145–163, 2015.

[39] R. Kiliç. *A Practical Guide for Studying Chua's Circuits*, volume 71 of *Series on Nonlinear Science: Series A*. World Scientific, 2010.

[40] E. B. Wilson and J. W. Gibbs. *Vector analysis: A text-book for the use of students of mathematics & physics, Founded upon the lectures of J. W. Gibbs.* Charles Scribner's Sons, 1901.

[41] S. Friedland, V. Mehrmann, R. Pajarola, and S. K. Suter. On best rank one approximation of tensors. *Numerical Linear Algebra with Applications*, 20:942–955, 2013.

[42] W. Hackbusch. *Tensor spaces and numerical tensor calculus*, volume 42 of *Springer Series in Computational Mathematics.* Springer, 2012.

[43] F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6:164–189, 1927.

[44] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[45] P. Gelß, S. Klus, S. Matera, and C. Schütte. Nearest-neighbor interaction systems in the tensor-train format. *Journal of Computational Physics*, 341:140–162, 2017.

[46] V. Kazeev, O. Reichmann, and C. Schwab. Low-rank tensor structure of linear diffusion operators in the TT and QTT formats. *Linear Algebra and its Applications*, 438(11):4204–4221, 2013.

[47] A. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, and D. P. Mandic. Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *Foundations and Trends in Machine Learning*, 9(4–5):249–429, 2016.

[48] E. Fermi, J. Pasta, and S. Ulam. Studies of nonlinear problems. Technical Report LA-1940, 1955.

[49] J. A. Acebrón, L. L. Bonilla, C. J. Pérez Vicente, F. Ritort, and R. Spigler. The Kuramoto model: A simple paradigm for synchronization phenomena. *Reviews of Modern Physics*, 77:137–185, 2005.

[50] Y. Kuramoto. Self-entrainment of a population of coupled non-linear oscillators. In H. Araki, editor, *Mathematical Problems in Theoretical Physics*, volume 39 of *Lecture Notes in Physics, International Symposium on Mathematical Problems in Theoretical Physics*, pages 420–422. Springer, 1975.

[51] L. F. Shampine and M. W. Reichelt. The MATLAB ODE suite. *SIAM Journal on Scientific Computing*, 18(1):1–22, 1997.

[52] S. Foucart and H. Rauhut. *A mathematical introduction to compressive sensing.* Springer, Heidelberg, 2013.

[53] H. Boche, R. Calderbank, G. Kutyniok, and J. Vybiral. *Compressed sensing and its applications.* Springer, Berlin.

[54] M. Gerster, P. Silvi, M. Rizzi, R. Fazio, T. Calarco, and S. Montangero. Unconstrained tree tensor network: An adaptive gauge picture for enhanced performance. *Physical Review B*, 90:125154, 2014.

[55] F. Verstraete and J. I. Cirac. Matrix product states represent ground states faithfully. *Physical Review B*, 73:094423, 2006.

[56] J. Eisert, M. Cramer, and M. B. Plenio. Area laws for the entanglement entropy. *Reviews of Modern Physics*, 82:277, 2010.

## A Exact coefficient tensors

For any pair of data matrices $\mathcal{X}, \mathcal{Y}$ related by one of the discussed ODE systems, the following tensor product representations of the coefficient tensors $\boldsymbol{\Xi}$ satisfy $\mathcal{Y} = \boldsymbol{\Xi}^T \boldsymbol{\Psi}(\mathcal{X})$ where $\boldsymbol{\Psi}(\mathcal{X})$ denotes the considered tensor of basis functions applied to the elements of $\mathcal{X}$.

### A.1 Exact solution for Chua's circuit

For the chosen basis functions $\mathbb{D} = \{x, |x|\}$ and the function-major decomposition given in (19), an exact tensor product representation of the coefficient tensor $\boldsymbol{\Xi} \in \mathbb{R}^{(d+1)\times(d+1)\times d}$ corresponding to the ODE (2) is given by

$$\boldsymbol{\Xi} = \left[\!\!\left[ \begin{bmatrix} 0 \\ -\alpha(1+\delta_1) \\ \alpha \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ -\alpha\delta_2 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -\beta \\ 0 \end{bmatrix} \right]\!\!\right] \otimes \left[\!\!\left[ \begin{matrix} e_1 & 0 & 0 \\ e_2 & 0 & 0 \\ 0 & e_1 & 0 \\ 0 & 0 & e_1 \end{matrix} \right]\!\!\right] \otimes \left[\!\!\left[ \begin{matrix} \tilde{e}_1 \\ \tilde{e}_2 \\ \tilde{e}_3 \end{matrix} \right]\!\!\right],$$

with $e_i \in \mathbb{R}^4$ and $\tilde{e}_j \in \mathbb{R}^3$ being the $i$th/$j$th unit vector of the standard basis in the respective space.

## A.2 Exact solution for the Fermi–Pasta–Ulam–Tsingou problem

For the chosen basis functions $\mathbb{D} = \{1, x, x^2, x^3\}$ and the coordinate-major decomposition given in (16), an exact tensor product representation of the coefficient tensor $\boldsymbol{\Xi} \in \mathbb{R}^{4\times\cdots\times 4\times d}$ corresponding to the ODE (20) is given by

$$\begin{aligned}
\boldsymbol{\Xi} \;=\; & \left[\!\left[ -2e_2 - 2\beta e_4 \quad e_1 + 3\beta e_3 \quad -3\beta e_2 \quad \beta e_1 \right]\!\right] \otimes \left[\!\left[ \begin{matrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{matrix} \right]\!\right] \otimes e_1 \otimes \cdots \otimes e_1 \otimes \tilde{e}_1 \\
& + \sum_{k=2}^{d-1} e_1 \otimes \cdots \otimes e_1 \otimes \left[\!\left[ e_1 \quad e_2 \quad e_3 \quad e_4 \right]\!\right] \\
& \qquad \otimes \underbrace{\left[\!\left[ \begin{matrix} -2e_2 - 2\beta e_4 & e_1 + 3\beta e_3 & -3\beta e_2 & \beta e_1 \\ e_1 + 3\beta e_3 & 0 & 0 & 0 \\ -3\beta e_2 & 0 & 0 & 0 \\ \beta e_1 & 0 & 0 & 0 \end{matrix} \right]\!\right]}_{k\text{th TT core}} \\
& \qquad \otimes \left[\!\left[ \begin{matrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{matrix} \right]\!\right] \otimes e_1 \otimes \cdots \otimes e_1 \otimes \tilde{e}_k \\
& + e_1 \otimes \cdots \otimes e_1 \otimes \left[\!\left[ e_1 \quad e_2 \quad e_3 \quad e_4 \right]\!\right] \otimes \left[\!\left[ \begin{matrix} -2e_2 - 2\beta e_4 \\ e_1 + 3\beta e_3 \\ -3\beta e_2 \\ \beta e_1 \end{matrix} \right]\!\right] \otimes \tilde{e}_d,
\end{aligned}$$

with $e_i \in \mathbb{R}^4$ and $\tilde{e}_j \in \mathbb{R}^d$ being the $i$th/$j$th unit vector of the standard basis in the respective space.

## A.3 Exact solution for the Kuramoto model

For the chosen basis functions $\mathbb{D} = \{\sin(x), \cos(x)\}$ and the function-major decomposition given in (22), an exact tensor product representation of the coefficient tensor $\boldsymbol{\Xi} \in \mathbb{R}^{(d+1)\times(d+1)\times d}$ corresponding to the ODE (21) is given by

$$\begin{aligned}
\boldsymbol{\Xi} \;=\; & e_1 \otimes e_1 \otimes \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_d \end{bmatrix} + h \cdot \sum_{k=1}^{d} e_{k+1} \otimes e_1 \otimes \tilde{e}_k \\
& + \frac{K}{d} \cdot \sum_{k=1}^{d} \left( \sum_{\substack{l=1 \\ l\neq k}}^{d} e_{k+1} \right) \otimes e_{k+1} \otimes \tilde{e}_k - \frac{K}{d} \cdot \sum_{k=1}^{d} e_{k+1} \otimes \left( \sum_{\substack{l=1 \\ l\neq k}}^{d} e_{k+1} \right) \otimes \tilde{e}_k,
\end{aligned}$$

with $e_i \in \mathbb{R}^{d+1}$ and $\tilde{e}_j \in \mathbb{R}^d$ being the $i$th/$j$th unit vector of the standard basis in the respective space.

## A.4 Role of entropies, locality and correlations

Since the considerations in this work rely strongly on the TT format, we briefly note what type of correlation structure is required to arrive at an efficient TT approximation of a given vector, cf. [55, 56]. Let us discuss to what extent this format can approximate a given unstructured vector $X \in \mathbb{R}^d$ (as they are arising in SINDy). It is known, see [55], that for any vector $X$, there exists a tensor train $\mathbf{X}$ with TT ranks $r_1, \ldots, r_{d-1} = r$ ($r_0 = r_d = 1$) for which the standard Euclidean vector norm satisfies

$$\|\mathbf{X} - X\|^2 \leq 2 \sum_{l=0}^{d-1} \epsilon_l(r),$$

with

$$\epsilon_l(r) := \sum_{i=r+1}^{N_i} \mu_l^i,$$

where $\{\mu_l^i : i = 1, \ldots, n_i\}$ are the eigenvalues of the partial traces over indices labeled $l+1, \ldots, d$ of the real symmetric matrices

$$R_l := \mathrm{tr}_{l+1,\ldots,d}(XX^T).$$

In this sense, the global quality of the approximation of $X$ by a suitable tensor train $\mathbf{X}$ can be related to thresholding errors. These errors are expected to be small when correlations are local and short-ranged. These correlations follow what is called an area law [56]. The above bounds can then be brought into contact with entropic correlation measures, i.e.,

$$\log(\epsilon_l(r)) \leq S^{1/2}(R_l) - \log(2r),$$

where $S^{1/2}(R_l) = 2 \log \mathrm{tr} \left( R_l^{1/2} \right)$ denotes the 1/2-Renyi entropy which appropriately captures correlations. If the correlations in the vectors are short-ranged in this sense, an efficient TT approximation is possible.