Discrete Function Bases and Convolutional Neural Networks

Andreas Stöckel
Centre for Theoretical Neuroscience
University of Waterloo

January 26, 2021

Abstract

We discuss the notion of "discrete function bases" with a particular focus on the discrete basis derived from the Legendre Delay Network (LDN). We characterize the performance of these bases in a delay computation task, and as fixed temporal convolutions in neural networks. Networks using fixed temporal convolutions are conceptually simple and yield state-of-the-art results in tasks such as psMNIST.

Main Results

(1) We present a numerically stable algorithm for constructing a matrix of DLOPs $\mathbf{L}$ in $\mathcal{O}(qN)$.

(2) The Legendre Delay Network (LDN) can be used to form a discrete function basis with a basis transformation matrix $\mathbf{H} \in \mathbb{R}^{q \times N}$.

(3) If $q < 300$, convolving with the LDN basis *online* has a lower run-time complexity than convolving with arbitrary FIR filters.

(4) Sliding window transformations exist for some bases (Haar, cosine, Fourier) and require $\mathcal{O}(q)$ operations per sample and $\mathcal{O}(N)$ memory.

(5) LTI systems similar to the LDN can be constructed for many discrete function bases; the LDN system is superior in terms of having a finite impulse response.

(6) We compare discrete function bases by linearly decoding delays from signals represented with respect to these bases. Results are depicted in fig. 20. Overall, decoding errors are similar. The LDN basis has the highest and the Fourier and cosine bases have the smallest errors.

(7) The Fourier and cosine bases feature a uniform decoding error for all delays. These bases should be used if the signal can be represented well in the Fourier domain.

(8) Neural network experiments suggest that fixed temporal convolutions can outperform learned convolutions. The basis choice is not critical; we roughly observe the same performance trends as in the delay task.

(9) The LDN is the right choice for small q, if the $\mathcal{O}(q)$ Euler update is feasible, and if the low $\mathcal{O}(q)$ memory requirement is of importance.

1 Introduction

The "Delay Network" is a recurrent neural network that approximates a time-delay of θ seconds (Voelker and Eliasmith, 2018). That is, given an input signal $u(t)$, the output of the network is approximately $u(t-\theta)$. Voelker (2019) points out that the impulse response of a variant of the dynamical system underlying this network traces out the Legendre polynomials. We hence refer to the Delay Network as the "Legendre Delay Network" (LDN), and to the linear time-invariant (LTI) system underlying the LDN as the "LDN system".

Voelker, Kajić, and Eliasmith (2019) demonstrate that a generalised neural network architecture derived from the LDN, the "Legendre Memory Unit" (LMU), can outperform other recurrent neural network architectures such as Long Short-Term Memories (LSTMs) in a wide variety of tasks. Preliminary work by Chilkuri and Eliasmith (publication in preparation) furthermore suggests that most weights in the LMU can be kept constant without negatively impacting the performance of the network. Surprisingly, this includes the recurrent connections in the LMU. Constant recurrent weights can be replaced by a set of static feed-forward Finite Impulse Response (FIR) filters arranged in a basis transformation matrix $\mathbf{H}$. This facilitates parallel training, leading to significant speed-ups.

The basis transformation matrix $\mathbf{H}$ can be interpreted as a discrete function basis. This report is concerned with characterizing such function bases, including the related "Discrete Legendre Orthogonal Polynomials" (DLOPs) introduced by Neuman and Schonbach (1974). Our goal is to gain a better understanding of the LDN system and to explore whether it could make sense to instead use other discrete function bases.

Structure of this report We first review the notion of a "discrete function basis" and "generalised Fourier coefficients". In particular, we discuss the Fourier and cosine series, as well as the Legendre polynomials. We review Discrete Legendre Orthogonal Polynomials (DLOPs), a discrete version of the Legendre polynomials proposed by Neuman and Schonbach (1974). We compare DLOPs to a discrete version of the LDN basis used by Chilkuri et al., followed by a method to reverse this process, i.e., to derive an LTI system from a (discrete) function basis. Furthermore, we discuss applying anti-aliasing filters to discrete function bases. We perform a series of experiments in which we characterise these bases in terms of the decoding error when computing delayed versions of signals represented in each basis. Lastly, we test each basis as a fixed temporal convolution in multi-layer neural networks and compare their performance to fully learned convolutions.

Many of the equations in this technical report are accompanied by a Python reference implementation; the name of the corresponding Python function is indicated in the margin. The latest version of this code is on GitHub, see

`https://github.com/astoeckel/dlop_ldn_function_bases`

2 Function Bases

As we will discuss in more detail in Section 4, the LDN system can be characterized as continuously computing the generalised Fourier coefficients of an input signal $u(t)$ over a window $[t - \theta, t]$ with respect to the orthonormal Legendre function basis. The point of this section is to define more thoroughly what we mean by that.

To this end, in Section 2.1, we first review some basic concepts from functional analysis, a field of mathematics that generalises linear algebra to infinite-dimensional function spaces. In Section 2.2, we review the orthonomal Legendre polynomials as an example of an orthonormal continuous function basis. Readers already familiar with the topic are welcome to skip ahead to Section 2.3, where we introduce the non-canonical notion of a discrete function basis and the corresponding notation, roughly following Neuman and Schonbach (1974).

2.1 Review: Function and Hilbert Spaces

The concept of vector spaces in linear algebra is general enough to include infinite-dimensional spaces, or, in other words, spaces that can only be spanned by an infinite number of basis vectors. A mathematically useful subset of possible vector spaces that encompasses both finite- and infinite-dimensional spaces are so-called "Hilbert spaces". We review this concept and discuss function bases that span the $L^2(a, b)$ Hilbert space, such as the Fourier and cosine bases.

Most of the material in this subsection closely follows Young (1988). We strongly advise the reader to consult this book for a more thorough (and undoubtedly more correct) treatment of the topic. A recommended gentle introduction to linear algebra itself is Hefferon (2020). Since we are not concerned with complex numbers in this report, we generally define all concepts over $\mathbb{R}$ instead of $\mathbb{C}$.

Definition 1 (Inner product space, induced norm, induced metric; cf. Young, 1988, Definitions 1.2, 1.6, Theorem 2.3)**.** An *inner product space* is a vector space[1] V with an associated inner product $\langle \cdot, \cdot \rangle : V \times V \longrightarrow \mathbb{R}$. The inner product must fulfil the following properties

1. *Symmetry:* $\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle$.
2. *Linearity:* $\langle \alpha\mathbf{x} + \mathbf{y}, \mathbf{z} \rangle = \alpha\langle \mathbf{x}, \mathbf{y} \rangle + \langle \mathbf{y}, \mathbf{z} \rangle$ for any $\alpha \in \mathbb{F}$.
3. *Positive definite:* $\langle \mathbf{x}, \mathbf{x} \rangle > 0$ if $\mathbf{x} \neq 0$.

If $\langle \mathbf{x}, \mathbf{y} \rangle = 0$, then $\mathbf{x}$ and $\mathbf{y}$ are called orthogonal. The norm $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$ is the *induced norm* of an inner product space; the metric $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$ is its *induced metric.*

Definition 2 (Function space)**.** A *function space* is an inner product space with $V = \{f \mid f : X \longrightarrow Y\}$. In other words, each $f \in V$ is a function mapping from a domain X onto a codomain Y. In this report we are concerned with $X, Y \subseteq \mathbb{R}$.

[1] A vector space is a set V with an addition and scalar multiplication operation over a field F. These operations must fulfil a set of requirements; see Hefferon, 2020, Definition 1.1.

Definition 3 (Function basis)**.** A *function basis* of a function space V is an infinite sequence $(e_n)_{n\in\mathbb{N}}$ of linearly independent functions $e_n \in V$ that span V. This is equivalent to demanding (cf. Theorem 1.12 in Hefferon, 2020) that each function $f \in V$ must be representable as a *unique* linear combination of e_n. There exists a unique sequence $(\xi_n)_{n\in\mathbb{N}}$ over $\mathbb{R}$ such that $f(x) = \sum_{n=0}^{\infty} \xi_n e_n(x)$ for each $f \in V$. Conversely, each function f constructed through such a linear combination must be an element of V.

Definition 4 (Orthogonal and orthonormal function bases)**.** A function basis is *orthogonal* if $\langle f_i, f_j \rangle = 0 \Leftrightarrow i \neq j$. A function basis is *orthonormal* if, additionally, $\langle f_i, f_j \rangle = 1 \Leftrightarrow i = j$.[2]

Example 1 (Continuous function space $\mathcal{C}[a, b]$)**.** An example of a function space would be the set of continuous scalar functions over an interval $[a, b]$, denoted as

$$\mathcal{C}[a, b] = \{f \mid f : [a, b] \longrightarrow \mathbb{R} \text{ and } f \text{ is continuous}\} .$$

This set is a vector space when coupled with addition $(f + g)(x) = f(x) + g(x)$ and scalar multiplication $(\lambda f)(x) = \lambda f(x)$ for $\lambda \in \mathbb{R}$. Furthermore, it can be shown that the following inner product over $\mathcal{C}[a, b]$ fulfils the above properties:

$$\langle f, g \rangle = \int_a^b f(x)g(x)\,\mathrm{d}x \,. \tag{1}$$

One might be inclined to think that the concept of a continuous function space $\mathcal{C}[a, b]$ is sufficient for most purposes. However, when trying to find a basis that spans $\mathcal{C}[a, b]$, one would eventually notice that any candidate basis can be used to generate discontinuous functions. Thus, the candidate basis does not span $\mathcal{C}[a, b]$, but a slightly larger space. The next example illustrates this.

Example 2 (Sign function as a series of continuous functions)**.** Consider the following sequence of basis functions $(f'_n)_{n\in\mathbb{N}}$ over $\mathcal{C}[-\pi, \pi]$

$$f'_0(x) = \frac{1}{\sqrt{2\pi}} \,, \qquad f'_{2n+1}(x) = \frac{\sin(nx)}{\sqrt{\pi}} \,, \qquad f'_{2n} = \frac{\cos(nx)}{\sqrt{\pi}} \,. \tag{2}$$

This basis is a variant of the "canonical Fourier series", an orthonormal function basis. Each individual f'_n is obviously in $\mathcal{C}[-\pi, \pi]$, and a linear combination of these basis functions can approximate any function in $\mathcal{C}[-\pi, \pi]$ (this follows from Theorem 5.1 in Young, 1988). However, the same basis can be used to express discontinuous functions. For example, one can show that a weighted series of the sine terms of f_n is equal to the sign function (see also fig. 1):

$$\operatorname{sign}(x) = \lim_{q\to\infty} \sum_{n=1}^{q} \frac{4\sin\big((2n-1)x\big)}{(2n-1)\pi} = \begin{cases} 1 & \text{if } x > 0\,, \\ 0 & \text{if } x = 0\,, \\ -1 & \text{if } x < 0\,. \end{cases} \tag{3}$$

Hence $\mathcal{C}[-\pi, \pi]$ has no basis that spans the space, which is slightly problematic.

[2]Note that the concept of a (function) basis being orthogonal is confusingly different from that of a matrix $\mathbf{A}$ being orthogonal, which is defined as $\mathbf{A}^T\mathbf{A} = \mathbf{I}$ and thus closer to the concept of an orthonormal basis.

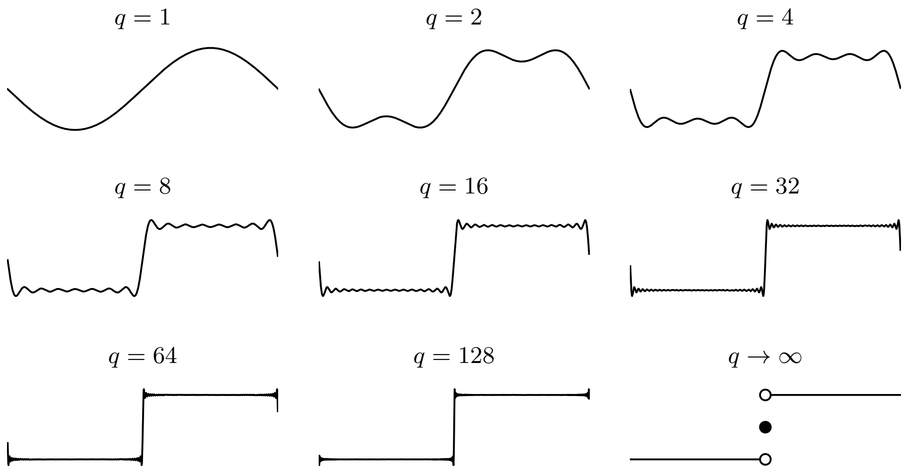

Figure 1: Approximating the discontinuous sign function (bottom right) using a sum of continuous sine waves according to eq. (3). Cauchy sequences of continuous functions can converge to a discontinuous function.

The notion of a "Hilbert space" restricts inner product spaces to those in which "well-behaved" sequences, so-called Cauchy sequences, converge to an element within that space. The sum of sines sequence implicitly defined in eq. (3) is an example of such a Cauchy sequence. Correspondingly, the function space $\mathcal{C}[a, b]$ cannot be a Hilbert space.

Definition 5 (Hilbert space; cf. Young, 1988, Definition 3.4)**.** A *Hilbert space* is an inner product space V in which all Cauchy sequences (relative to the metric induced by the inner product) converge to an element in V.

Example 3 (The Hilbert space $L^2(a, b)$; cf. Young, 1988, Example 3.5, Theorem 5.1)**.** The Fourier series in eq. (2) spans $L^2(-\pi, \pi)$. In general $L^2(a, b)$ is a function space V with the inner product defined in eq. (1). Each $f \in V$ is a function $f : [a, b] \longrightarrow \mathbb{R}$ for which the following Lebesgue integral converges; i.e., the function is square Lebesgue integrable:

$$\int_a^b f(x)^2 \,\mathrm{d}t < \infty \,, \qquad \text{where "}\textstyle\int\text{" is the Lebesque integral.}$$

Note that $L^2(a, b)$ is a superset of all square Riemann integrable functions.

Since the canonical Fourier series defined in eq. (2) spans $L^2(-\pi, \pi)$, any function in $L^2(-\pi, \pi)$ can be represented as a linear combination of Fourier basis functions. Of course, the same holds for any orthonormal function basis.

Definition 6 (Generalised Fourier series and coefficients; cf. Young, 1988, Definition 4.3)**.** Consider an orthonormal basis $(e_n)_{n \in \mathbb{N}}$, where each $e_n \in L^2(a, b)$, as well as a function $f \in L^2(a, b)$. Then the series

$$f = \sum_{n=0}^{\infty} \langle f, e_n \rangle e_n = \sum_{n=0}^{\infty} \xi_n e_n$$

is the generalised Fourier series of f and ξ_n are the generalised Fourier coefficients, also called the *spectrum* of f.

Note that the canonical Fourier series from eq. (2) and the associated Fourier coefficients ξ_n are related to, but not to be confused with, the Fourier *transformation.* The Fourier transformation represents any integrable function f in terms of a another function $\hat{f}(\xi)$.

In the following, we provide equations for the Fourier, cosine, and Legendre basis. We already introduced the canonical Fourier series in an example above over the interval $[-\pi, \pi]$. From now on, we define all bases over the interval $[0, 1]$ for the sake of consistency. Any orthonormal basis function e_n over $[0, 1]$ can be easily converted to an orthonormal basis function e'_n over $[a, b]$:

$$e'_n(x) = \frac{1}{\sqrt{|b-a|}} e_n \left(\frac{x-a}{b-a} \right) . \tag{4}$$

Definition 7 (Fourier series)**.** The Fourier series $(f_n)_{n \in \mathbb{N}}$ over $[0, 1]$ is given as

$$f_0(x) = 1 \,, \qquad f_{2n+1}(x) = \sqrt{2} \sin\left(2\pi n x\right) , \qquad f_{2n} = \sqrt{2} \cos\left(2\pi n x\right) . \tag{5}$$

This orthonormal basis spans $L^2(0, 1)$ and is depicted in Figure 2a.

Definition 8 (Cosine series)**.** An arguably simpler alternative to the Fourier series is the cosine series. The cosine series skips the "sine" terms of the Fourier series and increments the frequency in steps of π instead of 2π.

$$c_0(x) = 1 \,, \qquad c_n(x) = \sqrt{2} \cos(\pi n x) \,. \tag{6}$$

The orthonormal cosine series spans $L^2(0, 1)$ and is depicted in Figure 2b.

2.2 Review: Legendre Polynomials

In this report, we are mostly interested in the orthonormal Legendre basis generated by the Legendre polynomials. We first review the *orthogonal* Legendre polynomials p'_n; the *orthonormal* polynomials p_n are a scaled version of p'_n

Definition 9 (Legendre polynomials)**.** Legendre polynomials are uniquely defined as a sequence of functions $(p'_n)_{n \in \mathbb{N}}$ over $[-1, 1]$ with the following properties

1. *Polynomial:* p'_n is a linear combination of n monomials $p'_n(x) = \sum_{i=0}^{n} \alpha_{n,i} x^n$.
2. *Orthogonal:* $\langle p'_i, p'_j \rangle = 0$ exactly if $i \neq j$.
3. *Normalisation:* $p'_n(1) = 1$.

Below we summarize two methods to construct p'_n that fulfil these properties.

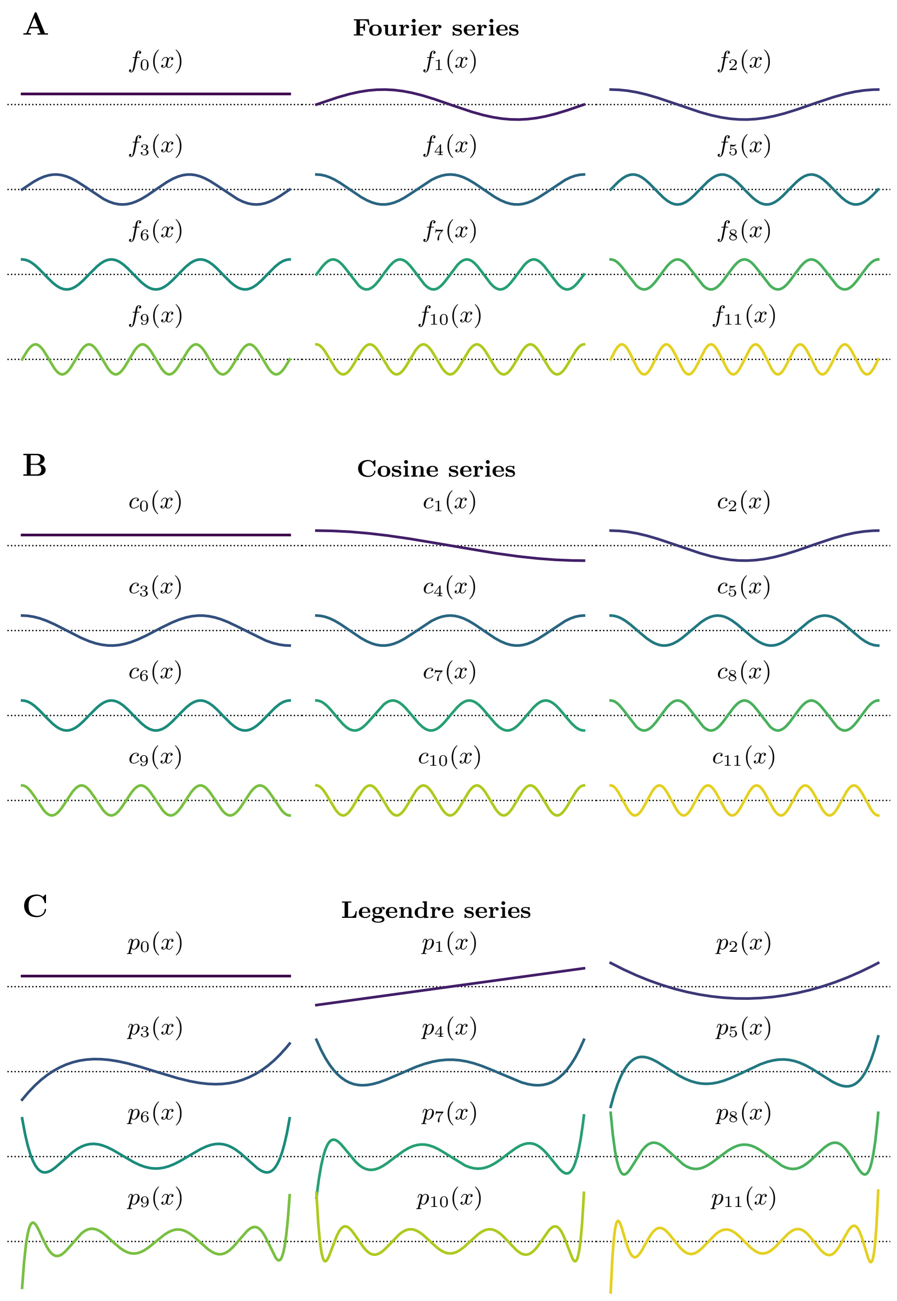

Figure 2: First functions in the orthonormal bases discussed in this section. All functions are plotted to the same scale; axes were omitted as the functions can be rescaled as described in eq. (4). Dotted line is zero.

Recurrence relation Starting with the base cases $p'_0(x) = 1$ and $p'_1(x) = x$, $p'_n(x)$ is given as a recurrence relation (Press et al., 2007, Section 5.4, p. 219):

$$(n+1)p'_{n+1}(x) = (2n+1)xp'_n(x) - np'_{n-1}(x)\,. \tag{7}$$

Rewriting this in terms of the polynomial coefficients $\alpha_{n,i}$ (see above) we get

$$(n+1)\alpha_{n+1,i} = \begin{cases} -n\alpha_{n-1,i} & \text{if } i = 0\,, \\ (2n+1)\alpha_{n,i-1} - n\alpha_{n-1,i} & \text{if } i > 0\,. \end{cases}$$

Closed form equation Alternatively, the Legendre polynomial p'_n is given in closed form as

$$p'_n(x) = \sum_{i=0}^{n} \binom{n}{i}\binom{n+i}{i}\left(\frac{x-1}{2}\right)^i\,.$$

The Legendre Delay Network approximates the shifted Legendre polynomials $\tilde{p}_n$ over $[0,1]$ given as $\tilde{p}_n(x) = p_n(2x-1)$. This substitution yields

$$\tilde{p}_n(x) = (-1)^n \sum_{i=0}^{n} (-1)^i \binom{n}{i}\binom{n+i}{i} x^i\,. \tag{8}$$

This equation can be easily decomposed into the monomial coefficients $\alpha_{n,i}$.

Definition 10 (Orthonormal Legendre series)**.** We can derive an orthnormal function basis $(p_n)_{n\in\mathbb{N}}$ over $[0,1]$ simply by dividing each shifted polynomial by the norm $\|\tilde{p}_n\|$. The basis is depicted in Figure 2c. It spans $L^2(0,1)$, just like the Fourier and cosine basis.[3]

$$p_n(x) = \frac{\tilde{p}_n(x)}{\|\tilde{p}_n\|} = \sqrt{2n+1}\,(-1)^n \sum_{i=0}^{n} (-1)^i \binom{n}{i}\binom{n+i}{i} x^i\,. \tag{9}$$

2.3 Discrete Function Bases

From a mathematical perspective, the notion of "discrete function bases" is at most moderately exciting. Once we discretise functions over an interval, we end up with boring, finite-dimensional vectors. Unfortunately, in practice, we more often than not have to work with discrete functions. Still, there is some potential for defining the concept of "discrete function bases" in relation to their continuous counterparts more rigorously.

Below, we define the notion of a "discrete function basis", as well as the corresponding "basis transformation matrix". Although our definitions are non-canonical, our notation roughly follows Neuman and Schonbach (1974).

[3]Sketch of a proof: for a bounded function space, an orthogonal polynomial function basis can be used to construct any analytic function over that interval. This includes sine and cosine, which can be made to span $L^2(0,1)$ by forming a canonical Fourier series.

Definition 11 (Discrete Function Basis)**.** A *discrete function basis* with an associated continuous function basis $(e_n)_{n \in \mathbb{N}}$ is a finite sequence of discrete basis functions $(E_n(k; N))_{n<N}$. $n \in \{0, \ldots, N-1\}$ is the basis function index, $k \in \{0, \ldots, N-1\}$ is the sample index, and $N \geq 1 \in \mathbb{N}$ is the number of samples. The codomain of $E_n(k; N)$ is $\mathbb{R}$. In the limit of $N \to \infty$ it must hold

$$\lim_{N \to \infty} E_n\left(k; N\right) = \frac{1}{\sqrt{N}} e_n\left(\frac{k}{N-1}\right) . \tag{10}$$

Intuitively, when sampling densely, an E_n fulfilling the above definition is indistinguishable from a scaled continuous basis function e_n. The scaling factor $1/\sqrt{N}$ ensures that inner products are preserved. It holds:

$$\langle e_i, e_j \rangle = \lim_{N \to \infty} \sum_{k=0}^{N-1} E_i(k; N) E_j(k; N) .$$

Definition 12 (Basis Transformation Matrix)**.** Let $E_n(k; N)$ be a discrete function basis. Given an order $q \leq N$, the basis transformation matrix $\mathbf{E} \in \mathbb{R}^{q \times N}$ is defined as

$$\left(\mathbf{E}\right)_{ij} = \frac{E_{i-1}(j-1; N)}{\sqrt{\sum_{k=0}^{N-1} E_{i-1}(k; N)^2}} , \quad \text{where } i \in \{1, \ldots, q\} , j \in \{1, \ldots, N\} .$$

The denominator ensures that each basis vector $(\mathbf{E})_n$ (the nth row in $\mathbf{E}$) has unit length. We call $\mathbf{E}$ *orthogonal* if $\mathbf{E}^T \mathbf{E} = \mathbf{I}$, where $\mathbf{I}$ is the $q \times q$ identity matrix. In contrast to the canonical meaning of "orthogonal", this includes non-square $\mathbf{E}$.

Interpreting E as a basis transformation Let $\mathbf{u} = (u_1, \ldots, u_N)$ be a discrete signal consisting of N samples. The matrix-vector product $\mathbf{E}\mathbf{u} = \mathbf{m}$ results in q inner products $\mathbf{m} = (m_1, \ldots, m_q)$ between the input signal $\mathbf{u}$ and each of the discrete basis functions in $\mathbf{E}$. For orthogonal $\mathbf{E}$, the resulting $\mathbf{m}$ can be interpreted as a set of discrete generalised Fourier coefficients. That is, the vector $\mathbf{m}$ represents the signal $\mathbf{u}$ with respect to a normalised version of the discrete function basis $E_n(k; N)$. Correspondingly, this operation is a basis transformation in the same sense as the discrete Fourier and cosine transformations (discussed below).

Interpreting E as a set of FIR filters Another way to think about $\mathbf{E}$ is as a set of finite impulse-response (FIR) filters. Let $\mathbf{u}_t$ represent the last N samples of an input signal relative to a time t. Specifically, $\mathbf{u}_t = (u_{t-(N-1)}, \ldots, u_t)$; i.e., the newest sample is shifted in from the right. Then, $\mathbf{m} = \mathbf{E}\mathbf{u}_t$ can be written as

$$m_n = \left\langle \left(\mathbf{E}\right)_n, \mathbf{u} \right\rangle = \sum_{k=0}^{N-1} \left(\mathbf{E}\right)_{n,N-k} u_{t-k} . \tag{11}$$

This is exactly the definition of a FIR filter of order $N-1$ (cf. Press et al., 2007, Section 13.5.1, p. 668); but notice the inverted matrix column order $N-k$.

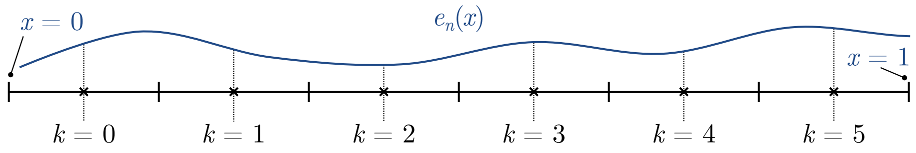

Figure 3: Illustration of the sampling process in eq. (12). The function $e_n(x)$ is sampled at the centre of $N = 6$ intervals.

Now that we have defined discrete function bases and the corresponding normalised basis transformation matrix, we should discuss how to construct discrete function bases. Unfortunately, there is no universal method, and the next two examples are only two of many possible methods.

Example 4 (Naive sampling)**.** Equation (10) directly suggests a way to generate discrete function bases. This "naively sampled discrete function basis" is

$$E_n(k; N) = \frac{1}{\sqrt{N}} e_n\left(\frac{k + \frac{1}{2}}{N}\right) . \tag{12}$$

The offset of one half ensures that samples are taken at the centre of the N discrete intervals (cf. fig. 3).

Example 5 (Mean sampling)**.** Another way to construct a discrete function basis is to compute the mean over each of the N intervals. That is

$$E_n(k; N) = \sqrt{N} \int_{x_0}^{x_1} e_n(x) \,\mathrm{d}x \,, \text{ where } x_0 = k/N \text{ and } x_1 = (k + 1)/N \,. \tag{13}$$

Unfortunately, one caveat with both sampling methods is that they do not necessarily preserve orthogonality of the function basis that is being sampled. This is illustrated in the three examples below. While naive sampling perfectly preserves orthogonality of the Fourier and cosine series, neither method results in an orthogonal basis transformation matrix for the Legendre polynomials.

Maintaining orthogonality can be important. Mathematically, having orthogonal matrices can simplify some equations, as we will see later. From an information-theoretical perspective, orthogonal bases minimize correlations between individual state dimensions and thus (when considering a probability distribution of input signals) minimize pairwise mutual information between the generalised Fourier coefficients, maximizing their negative entropy (cf. Comon, 1994, Sections 2.1-2.3 for definitions and the relationship between mutual information and negentropy). It should be noted that this can be undesirable if the resulting representation is subject to noise.

Example 6 (Discrete Fourier Basis)**.** As mentioned above, applying naive sampling from eq. (12) to the Fourier series in eq. (5) yields an orthogonal basis transformation matrix $\mathbf{F}$. This $\mathbf{F}$ can be interpreted as the linear operator implementing the discrete Fourier transformation (DFT). That is, multiplying a real signal **u** with **F** computes the DFT of **u**. The individual discrete basis functions are given as

$$\begin{aligned} F_0(k; N) &= \frac{1}{\sqrt{N}}\,, \\ F_{2n+1}(k; N) &= \frac{\sqrt{2}}{\sqrt{N}} \sin\left(2\pi n \frac{k + \frac{1}{2}}{N}\right)\,, \\ F_{2n}(k; N) &= \frac{\sqrt{2}}{\sqrt{N}} \cos\left(2\pi n \frac{k + \frac{1}{2}}{N}\right)\,. \end{aligned} \tag{14}$$

This equation is implemented in the function `mk_fourier_basis`.

Normalisation of the matrix **F** as defined in eq. (10) is not required if one special case is taken into account. The normalisation factor needs to be updated in the case $n + 1 = q = N$ for even N

$$F_{N-1}(k; N) = \frac{1}{\sqrt{N}} \sin\left(2\pi n \frac{k + \frac{1}{2}}{N}\right) = \frac{(-1)^k}{2\pi} \quad \text{if } N = q \text{ even.}$$

Taking this special case into account, the discrete Fourier function basis is orthonormal, i.e., it holds

$$\sum_{k=0}^{N-1} F_i(k; N) F_j(k; N) = \begin{cases} 1 & \text{if } i = j\,, \\ 0 & \text{if } i \neq j\,. \end{cases}$$

The corresponding basis transformation matrix **F** is depicted in Figure 4.

Example 7 (Discrete Cosine Basis)**.** Similarly to the discretisation of the Fourier series, applying naive discretisation from eq. (12) to the cosine series results in the discrete Cosine transformation (DCT). Again, the resulting equations are significantly simpler than the discrete Fourier transformation:

$$C_0(k; N) = \frac{1}{\sqrt{N}}\,, \qquad C_n(k; N) = \frac{\sqrt{2}}{\sqrt{N}} \cos\left(\pi n \frac{k + \frac{1}{2}}{N}\right)\,. \tag{15}$$

This equation is implemented in `mk_cosine_basis`.

This discrete function basis is orthonormal. The corresponding basis transformation matrix **C** is depicted in Figure 4 as well.

Time complexity The matrices **F** and **C** can be computed in time $\mathcal{O}(qN)$; a constant number of operations is required to evaluate each cell.

Multiplication of a vector **u** with **F** or **C** can be performed in $\mathcal{O}(qN \log(N))$. This is due each left and right half of **F** and **C** resembling a scaled and mirrored version of the full matrix. This suggests a divide and conquer algorithm if N is a power of two—the Fast Fourier Transformation (FFT; Cooley and Tukey, 1965) and the related Fast Cosine Transformation (FCT; Makhoul, 1980). Both algorithms can be generalised to non-power of two N, and the publications cited above discuss how to accomplish this.

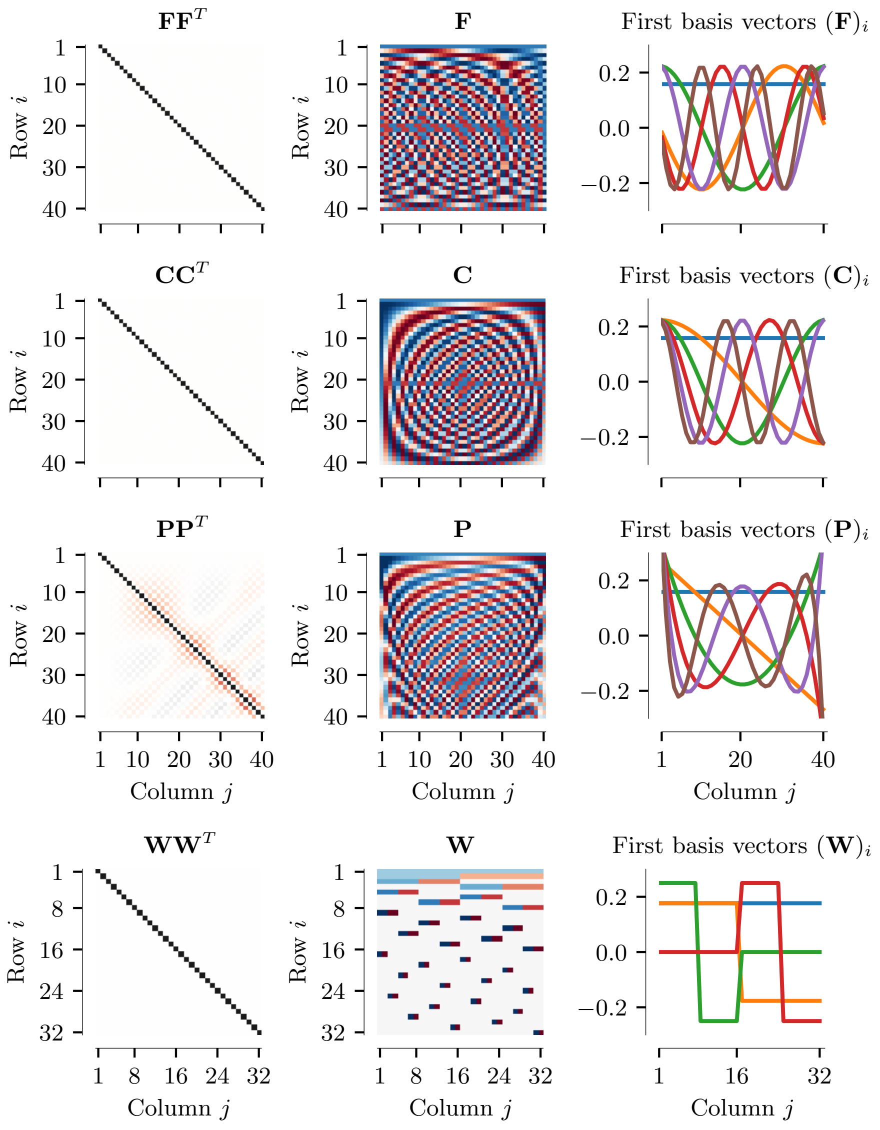

Figure 4: The discrete Fourier, cosine, Legendre and Haar basis transformation matrices (top to bottom). *Left:* Visualisation of the outer product of each matrix. Each pixel is a matrix cell i, j. Zero is white, one is black, negative values are red. The matrices $\mathbf{F}$, $\mathbf{C}$, $\mathbf{W}$ are orthogonal; $\mathbf{P}$ is not orthogonal. *Centre:* Basis matrices themselves, where white corresponds to zero, red to negative and blue to positive numbers (colour maps rescaled to cover 95% of the represented values without saturating). *Right:* Visualisation of the first basis vectors.

Example 8 (Naive and Mean Sampled Discrete Legendre Basis)**.** We can similarly apply the naive sampling (eq. 12) to the shifted Legendre polynomials (eq. 8). For the sake of consistency with the bases presented in the next sections, we furthermore "mirror" the shifted Legendre polynomials, i.e., we compute $\tilde{p}_n(1-x)$ instead of $\tilde{p}(x)$. We get

$$P'_n(k; N) = \tilde{p}_n \left(1 - \frac{k + \frac{1}{2}}{N} \right) .$$

Unfortunately, as mentioned above, the corresponding basis transformation matrix $\mathbf{P}'$ is not orthogonal.

A slightly "more orthogonal" (in terms of the off-diagonal elements having a smaller magnitude) discrete function basis can be obtained by applying mean sampling as defined in eq. (13), resulting in

$$P_n(k; N) = \sqrt{N} \int_a^b \tilde{p}_n(x) \, \mathrm{d}x \, , \qquad \text{where } a = 1 - \frac{k+1}{N} \text{ and } b = 1 - \frac{k}{N} \, .$$

Since the $\tilde{p}_n$ are polynomials, the antiderivatives $\tilde{P}_n$ are given in closed form:

$$P_n(k; N) = \sqrt{N} \left(\tilde{P}_n \left(1 - \frac{k}{N} \right) - \tilde{P}_n \left(1 - \frac{k+1}{N} \right) \right) . \tag{16}$$

This equation is implemented in `mk_leg_basis`.

The corresponding basis transformation matrix $\mathbf{P}$ is depicted in Figure 4.

Time complexity Computing the basis transformation matrix $\mathbf{P}$ has a time-complexity in $\mathcal{O}(q^2 N)$—evaluating one of the $q \times N$ polynomials in $\mathbf{P}$ requires up tp q multiplications using Horner's method. The standard run-time costs $\mathcal{O}(qN)$ for a matrix-vector multiplication apply when evaluating $\mathbf{Pu}$.

Example 9 (Discrete Haar Wavelet Basis)**.** Continuous and discrete wavelet transformations are a popular alternative to Fourier-like transformations. The basic idea of wavelet bases is to have a single "mother" wavelet from which the individual basis functions are derived. In contrast to the Fourier series, wavelet bases are sparse; basis functions tend to be zero for most of the covered interval.

One popular wavelet basis is the Haar basis $(w_n)_{n \in \mathbb{N}}$. Aside from the first basis function $w_0(x) = 1$, each $w_n(x)$ for $n \geq 2$ is a scaled and shifted version of $w_1(x)$. The complete orthonormal Haar basis over $[0, 1]$ is given as

$$w_1(x) = \begin{cases} 1 & \text{if } 0 \leq x < \frac{1}{2} \, , \\ -1 & \text{if } \frac{1}{2} \leq x \leq 1 \, , \\ 0 & \text{otherwise} \, , \end{cases} \qquad \begin{aligned} \text{and } w_n(x) &= \sqrt{\varphi} \, w_1\big(\varphi x - n + \varphi\big) \, , \\ \text{where } \varphi &= 2^{\lfloor \log_2(n) \rfloor} \, . \end{aligned} \tag{17}$$

A discrete version of this basis can be obtained using `mk_haar_basis`.

A discrete version $W_n(k; N)$ with basis transformation matrix $\mathbf{W}$ can be easily computed in $\mathcal{O}(qN)$; such a (reordered) $\mathbf{W}$ is depicted in Figure 4. An interesting property of this basis is that the "Fast Haar transformation" can be computed in $\mathcal{O}(N)$ (Kaiser, 1998). This is even faster than the fast Fourier or cosine transformations, which require $\mathcal{O}(N \log(N))$ operations.

3 A Discrete Orthogonal Legendre Basis: DLOPs

The previous section introduced the notion of a discrete function basis. We saw that the cosine and Fourier series could be trivially discretised while preserving orthogonality. However, doing the same for the Legendre polynomials did not preserve orthogonality.

In this section, we construct a discrete, orthogonal Legendre function basis. In Section 3.1 we translate the definition of a Legendre polynomial to discrete function basis, resulting in "Discrete Legendre Orthogonal Polynomials", or "DLOPs" in short. DLOPs were originally proposed by Neuman and Schonbach (1974). Fortunately, Neuman and Schonbach present a simple equation that can be used to construct DLOPs. We review this equation in Section 3.2. We close in Section 3.3 with the description of an efficient and numerically stable algorithm to compute DLOPs in $\mathcal{O}(qN)$.

3.1 Discrete Legendre Orthogonal Polynomials

We can apply the definition of a Legendre polynomial (Definition 9) to a discrete function basis. The unique basis fulfilling this definition is a discrete function basis of the Legendre polynomials in the strict sense of Definition 11.[4]

Definition 13 (Discrete Legendre Orthogonal Polynomials, DLOPs; adapted from Neuman and Schonbach, 1974)**.** DLOPs are defined as the discrete function basis $L_n(k; N)$ with the following properties

1. *Polynomial:* Each $L_n(k; N)$ is a linear combination of n monomials. It holds $L_n(k; N) = \sum_{j=0}^{n} \alpha_{n,i} \left(\frac{k}{N-1}\right)^i$ for $k \in \{0, \dots, N-1\}$.
2. *Orthogonal:* $\sum_{k=0}^{N-1} L_i(k; N) L_j(k; N) = 0$ exactly if $i \neq j$.
3. *Normalisation:* $L_n(0; N) = \frac{1}{\sqrt{N}}$.

The function `mk_dlop_basis_linsys` *implements this algorithm.*

Numerically solving for DLOPs This definition suggests a simple algorithm that can be used to construct a discrete orthogonal Legendre basis matrix $\mathbf{L} \in \mathbb{R}^{q \times N}$. We initialize the first row of $\mathbf{L}$ as ones. To obtain a row n, we solve for the polynomial coefficients $\alpha_{n,i}$ such that the above conditions are fulfilled. That is, the new row is orthogonal to all preceding rows and the entry in last column is equal to one. Finding coefficients $\alpha_{n,i}$ that fulfil these requirements is simply a matter of solving a system of linear equations.

While this algorithm works in theory, it is numerically unstable in practice. Monomials x^k with $|x| \ll 1$ and $k \gg 20$ cannot be represented well using double-precision floating point arithmetic. This mandates the use of arbitrary-precision rational numbers. Fortunately, there is no need to actually implement this algorithm since a closed-form solution exists.

[4]Sketch of a proof: for $N \to \infty$ the sums over N turn into integrals that match the exact definition of Legendre polynomials.

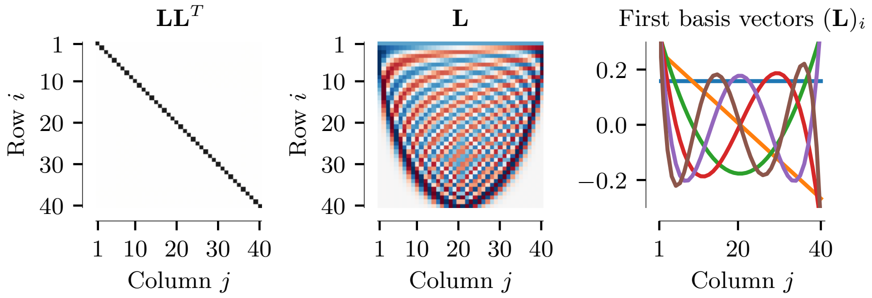

Figure 5: Visualisation of the DLOP basis defined in eq. (18) for $q = N = 40$. See fig. 4 for the complete legend and a description of the colour scheme.

3.2 Closed-Form Solution for DLOPs

Neuman and Schonbach show that the nth DLOP is given in closed form as

$$L_n(k; N) = \frac{1}{\sqrt{N}} \sum_{i=0}^{n} (-1)^i \binom{n}{i} \binom{n+i}{i} \frac{k^{(i)}}{(N-1)^{(i)}} \,, \tag{18}$$

$$\text{where } k^{(i)} = \prod_{j=0}^{i-1} (k-j) = \frac{k!}{(k-i)!} \quad \text{is the } i\text{th } \textit{fading factorial} \text{ of } k. \tag{19}$$

Factoring out $(N-1)^{(n)}$ facilitates the use of arbitrary precision integers

`mk_dlop_basis_direct` uses arbitrary precision integers to evaluate this equation.

$$L_n(k; N) = \frac{1}{\sqrt{N}(N-1)^{(n)}} \sum_{i=0}^{n} (-1)^i \binom{n}{i} \binom{n+i}{i} k^{(i)} (N-1-i)^{(n-i)} \,. \tag{20}$$

The corresponding normalised matrix $\mathbf{L} \in \mathbb{R}^{q \times N}$ for $q = N = 40$ is depicted in Figure 5. Notice that the resulting matrix is perfectly orthogonal. Comparing the DLOP matrix $\mathbf{L}$ to the naive discrete Legendre basis $\mathbf{P}$ (cf. fig. 4), we find that the two bases are strikingly different for higher-order terms. The last rows in $\mathbf{L}$ have many near-zero entries with non-zero values centred around $k = N/2$.

The time-complexity of evaluating the corresponding basis transformation matrix $\mathbf{L} \in \mathbb{R}^{q \times N}$ using eq. (18) in $\mathcal{O}(q^2 N)$.

3.3 Efficiently Computing DLOP Coefficients

As noted above, the time-complexity of evaluating eq. (18) is in $\mathcal{O}(q^2 N)$. Furthermore, the equation can only evaluated reliably using arbitrary precision integers. Neuman and Schonbach (1974) propose an $\mathcal{O}(qN)$ algorithm that relies on a variant of the recurrence relation from eq. (7). In this section, we discuss a version of this algorithm that generates an orthonormal matrix $\mathbf{L}$ using standard double-precision floating point arithmetic.

The discrete Legendre recurrence relation presented in the paper is

$$
\begin{aligned}
L_0(k;N) &= \frac{1}{\sqrt{N}}\,, \\
L_1(k;N) &= \frac{1}{\sqrt{N}}\frac{(2k-N+1)}{N-1}\,, \\
L_n(k;N) &= \quad L_{n-1}(k;N)\frac{(2n-1)(N-2k-1)}{n(N-n)} \\
&\quad - L_{n-2}(k;N)\frac{(n-1)(N+n-1)}{n(N-n)}\,.
\end{aligned}
$$

The function `mk_dlop_basis_recurrence` *implements this particular equation, using an algorithm proposed by Neuman and Schonbach.*

Naively evaluating this recurrence numerically is not stable for $n > 40$. Some of the columns k grow exponentially in magnitude with n.

For a numerically stable algorithm we suggest to ensure that each $L_n(k;N)$ is normalised. This normalised discrete function basis $L'_n(k;N)$ has the property

$$
\sum_{k=0}^{N-1} L'_n(k;N)^2 = \sum_{k=0}^{N-1} \left(\sqrt{\alpha_n(N)}L_n(k;N)\right)^2 = 1\,,
$$

i.e., it is orthonormal. According to Neuman and Schonbach (p. 746), the normalisation factor $\alpha_n(N)$ is

$$
\alpha_n(N) = \frac{(2n+1)(N-1)^{(n)}}{(N+n)^{(n+1)}}\,,
$$

where $k^{(i)}$ is the ith fading factorial of k, as defined in eq. (19). Applying the normalisation, we get the following recurrence relation for L'_n:

$$
\begin{aligned}
L'_0(k;N) &= \frac{1}{\sqrt{N}}\,, \\
L'_1(k;N) &= \frac{(2k-N+1)}{N-1}\sqrt{\frac{3(N-1)}{N(N+1)}}\,, \\
L'_n(k;N) &= \quad L'_{n-1}(k;N)\frac{(2n-1)(N-2k-1)}{n(N-n)}\sqrt{\frac{\alpha_n(N)}{\alpha_{n-1}(N)}} \\
&\quad - L'_{n-2}(k;N)\frac{(n-1)(N+n-1)}{n(N-n)}\sqrt{\frac{\alpha_n(N)}{\alpha_{n-2}(N)}}\,.
\end{aligned}
\tag{21}
$$

The function `mk_dlop_basis` *implements this set of equations and addresses the numerical instability mentioned below.*

Multiplying with the square root of $\alpha_n(N)$ applies the normalisation, dividing by the square roots of $\alpha_{n-1}(N)$ and $\alpha_{n-2}(N)$ reverts the normalisation applied to the lower-order discrete basis function. These fractions can be simplified to

$$
\frac{\alpha_n(N)}{\alpha_{n-1}(N)} = \frac{(2n+1)(N-n)}{(2n-1)(N+n)}\,, \quad \frac{\alpha_n(N)}{\alpha_{n-2}(N)} = \frac{(2n+1)(N-n)(N-n+1)}{(2n-3)(N+n)(N+n-1)}\,.
$$

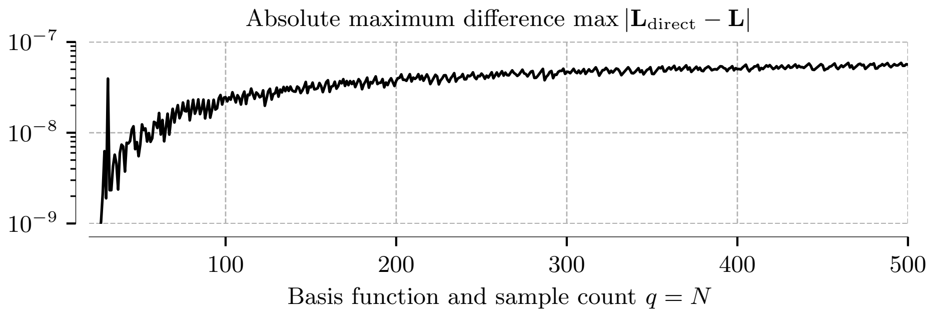

Figure 6: Comparing the basis transformation matrix $\mathbf{L}$ obtained when evaluating the recurrence relation $\mathbf{L}$ to the matrix $\mathbf{L}_{\mathrm{direct}}$ obtained when evaluating the closed-form equation eq. (18). Even for large q, errors do not exceed 10^{-7}.

Numerical stability Applications requiring utmost numerical robustness should use eq. (20) with arbitrary precision integers and subsequent normalisation; the recurrence relation in eq. (21) inadvertently propagates numerical errors.

In particular, special care must be taken when implementing eq. (21). Whenever a cell in column k is close to zero in two consecutive rows $n-1$, n, all consecutive cells in column k of rows $n+i$ must be zero as well. Ensuring this is important, since small non-zero values caused by numerical instabilities can rebound exponentially when applying the recurrence relation.

Figure 6 shows the maximum absolute difference between the matrix $\mathbf{L}_{\mathrm{direct}}$ obtained when using eq. (20) and $\mathbf{L}$ computed using our proposed recurrence relation in eq. (21). Errors do not exceed 10^{-7}, even for $q = 500$.

4 Constructing a Discrete LDN Basis

In this section, we focus on the Legendre Delay Network (LDN) and the corresponding basis transformation matrix $\mathbf{H} \in \mathbb{R}^{q \times N}$. We first review the Linear Time Invariant (LTI) system underlying the LDN and observe that the impulse response of the LDN system resembles a continuous Legendre function basis over the interval $[0, \theta]$. The impulse response sharply decays to zero for $t > \theta$, that is, the system has an *almost* finite impulse response. Second, as originally proposed by Chilkuri, we construct the matrix $\mathbf{H}$ to approximate the impulse response.

We close by discussing why the LDN system may be particularly useful. To summarize, since $\mathbf{H}$ was derived from an LTI system with an (almost) finite impulse response, we can either use the basis transformation matrix $\mathbf{H}$ (i.e., a set of FIR filters) *or* the LTI system itself to compute generalised Fourier coefficients $\mathbf{m}$ of an input signal $\mathbf{u}_t$. The FIR filter representation is useful when training neural networks; during inference the LDN LTI system can directly be used as a fast "sliding transformation". In contrast to other sliding transformations the LDN LTI system requires a minimal amount of state memory.

4.1 Review: The LDN System

The LDN system (Voelker and Eliasmith, 2018) can be thought of as continuously compressing a θ second long time-window of a function $u(t)$ into a q-dimensional vector $\mathbf{m}(t)$. The system is derived from the Padé approximants (Baker, 2012) of a Laplace-domain delay $e^{-s\theta}$, along with a set of transformations that make the system numerically stable. Let $\mathbf{A} \in \mathbb{R}^{q \times q}$, $\mathbf{B} \in \mathbb{R}^{q \times 1}$. Then, the LDN system is given as

The function **`mk_ldn_lti`** *generates the LDN system matrices* $\mathbf{A}$, $\mathbf{B}$.

$$\frac{d\theta \mathbf{m}(t)}{dt} = \mathbf{A}\mathbf{m}(t) + \mathbf{B}u(t)\,,$$
$$\big(\mathbf{A}\big)_{ij} = (2i+1)\begin{cases} -1 & \text{if } i \leq j\,, \\ (-1)^{i-j+1} & \text{if } i > j\,, \end{cases} \qquad \big(\mathbf{B}\big)_{ij} = (2i+1)(-1)^i\,. \tag{22}$$

As discovered by Voelker (2019, Section 6.1.3, p. 134), the normalised impulse response $\tilde{\mathbf{m}}^q(t)$ of this system over a time window $[0, \theta]$ resembles the first q shifted Legendre polynomials scaled to the interval $[0, \theta]$. Judging from numerical evidence, it seems reasonable to assume that for $q \to \infty$ the impulse response and the Legendre polynomials are exactly equal. Correspondingly, $(\tilde{m}^q_n(t))_{n \in \mathbb{N}}$ for $q \to \infty$ forms a function basis over the interval $[0, \theta]$. While empirical evidence suggests that this is true, we do *not* have a rigorous proof for this.

Mathematically, the impulse response $\tilde{\mathbf{m}}^q(t)$ of the normalised LDN system and the presumed relationship to the Legendre polynomials is given as

$$\tilde{m}^q_n(t) = \frac{\sqrt{\theta}}{\sqrt{2n+1}} e^{\mathbf{A}t}\mathbf{B} = \frac{1}{\sqrt{\theta}} p_n\left(\frac{t}{\theta}\right)\,, \quad \text{where } t \in [0, \theta],\ \text{for } q \to \infty\,. \tag{23}$$

$\mathbf{A} \in \mathbb{R}^{q \times q}$ and $\mathbf{B} \in \mathbb{R}^{q \times 1}$ are as defined in eq. (22); the orthonormal Legendre polynomial p_n is as defined in eq. (9) with the re-scaling from eq. (4) applied.

The LDN impulse response and the corresponding Legendre polynomials are depicted in Figure 7. Two observations are worth being pointed out.

First, notice how the impulse response of the LDN system sharply converges to zero for $t > \theta$. In other words, the LDN system has no memory of anything happening more than θ seconds ago. This is one of the key properties of the LDN system, and we revisit this in the next section, when we discuss how to construct LTI systems from discrete function bases (i.e., the inverse of what we are doing in this section).

Second, for finite q, the implicit basis created by the LDN system is *not exactly* the Legendre basis, but an *approximation.* We refer to the finite sequence of q functions generated by the LDN as the LDN "basis", although, technically, a finite sequence of functions cannot form a continuous function basis.

4.2 Constructing the LDN Discrete Function Basis

To compute $\mathbf{H}$, we could just apply the "mean sampling" discussed in Example 5 to the impulse responses $\tilde{m}^q_n(t)$. In fact, this is exactly what we will end up doing. However, this was not how we derived $\mathbf{H}$ in the first place, and we find it

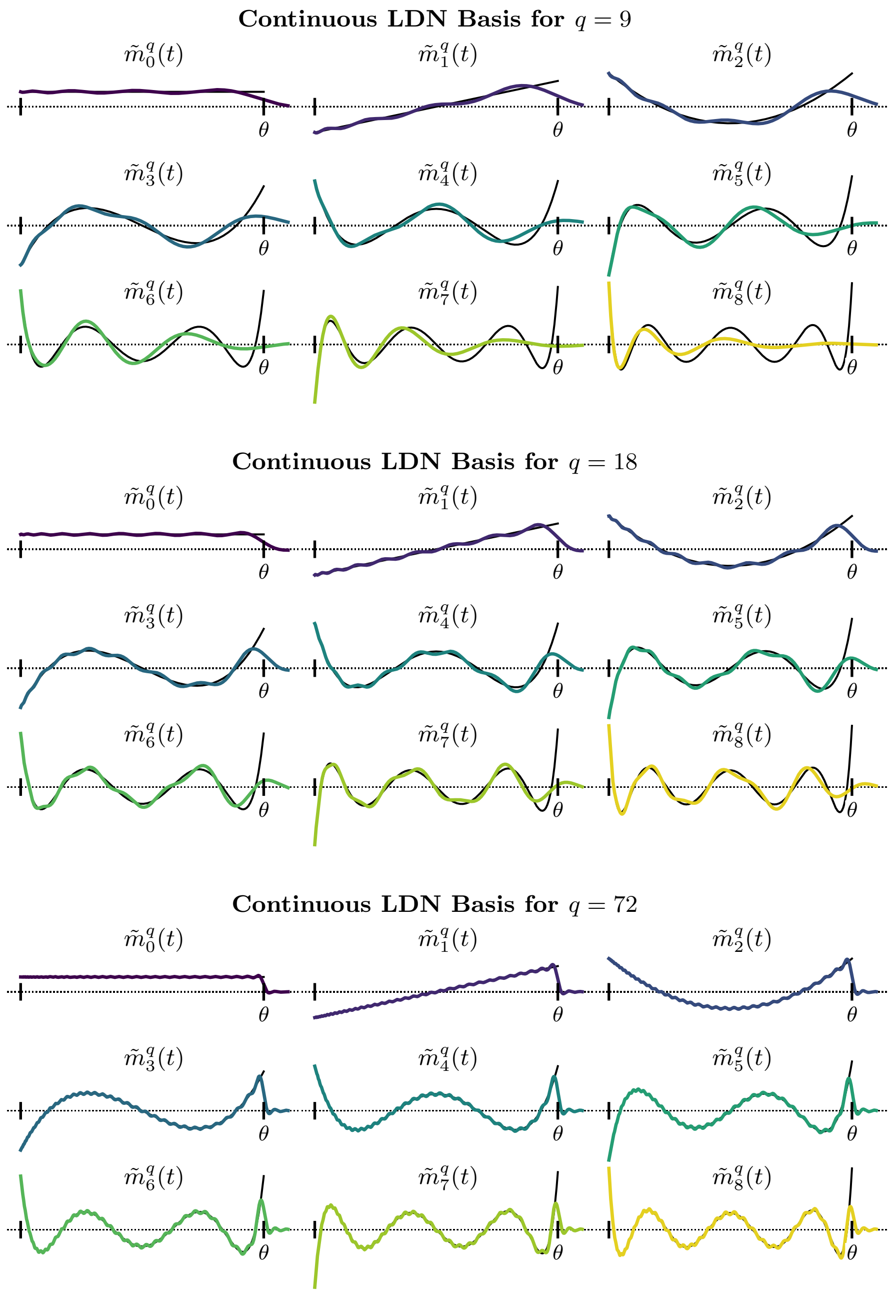

Figure 7: Normalised impulse response of the first six state dimensions of the LDN as defined in eq. (23) for different state dimensionalities q (coloured lines). Solid black lines correspond to the orthonormal Legendre basis p_n. As q increases, the impulse response more closely resembles the Legendre basis.

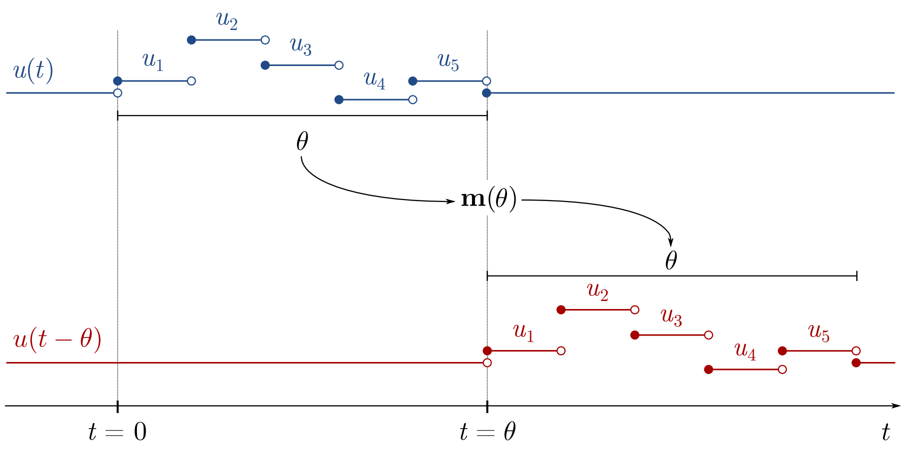

Figure 8: Diagram illustrating a perfect delay. A stair-step function $u(t)$ representing $N = 5$ values $u_1, \ldots, u_5$ between $t = 0$ and $t = \theta$ is delayed by exactly θ seconds, resulting in $u(t - \theta)$. When implementing the delay, the LDN system must represent information about the last θ seconds in its state vector $\mathbf{m}(t)$. In this example, all samples *must* be represented in $\mathbf{m}(\theta)$.

considerably more instructive to discuss our original derivation. We then prove equivalence of the resulting expression to mean sampling. Impatient readers interested in the main result may wish to skip ahead to the end of this subsection.

Compressing signals using the LDN Feeding a signal $u(t)$ into the Legendre Delay Network allows us, as the name suggests, to decode a delayed signal $u(t - \theta)$ from its state vector $\mathbf{m}(t)$. Consider what happens if we present N samples $u_1, \ldots, u_N$ to the LDN system over the time-window θ, for example using a stair-step function

$$u(t) = \begin{cases} u_i & \text{if } 0 \leq t < \theta\,, \\ 0 & \text{otherwise,} \end{cases} \qquad \text{where } i = 1 + \left\lfloor \frac{Nt}{\theta} \right\rfloor . \tag{24}$$

At time $t = \theta$ all samples have been presented to the LDN. If the LDN were to implement a perfect delay, we would decode $u(t - \theta)$, which is equal to the first sample u_1. A bit later, at time $t = \frac{N+1}{N}\theta$, the network would output u_2, and so on (cf. fig. 8). This means that at $t = \theta$, the network has "compressed" all N samples into its q-dimensional state vector $\mathbf{m}(t)$. Of course, the LDN system acts as a basis function transformation that represents $\mathbf{u} = (u_1, \ldots, u_N)$ as a vector $\mathbf{m}(\theta) = (m_1(\theta), \ldots, m_q(\theta))$ with respect to a discrete function basis.

Our goal is to find a linear expression mapping $\mathbf{u}$ onto $\mathbf{m}(\theta)$. Given such an expression, we could extract the corresponding basis transformation matrix $\mathbf{H}$.

Naive Euler recurrence relation In a first step, let us coarsely discretise the so far continuous functions. Let $\theta = N\Delta t$, where Δt is the timestep. In this

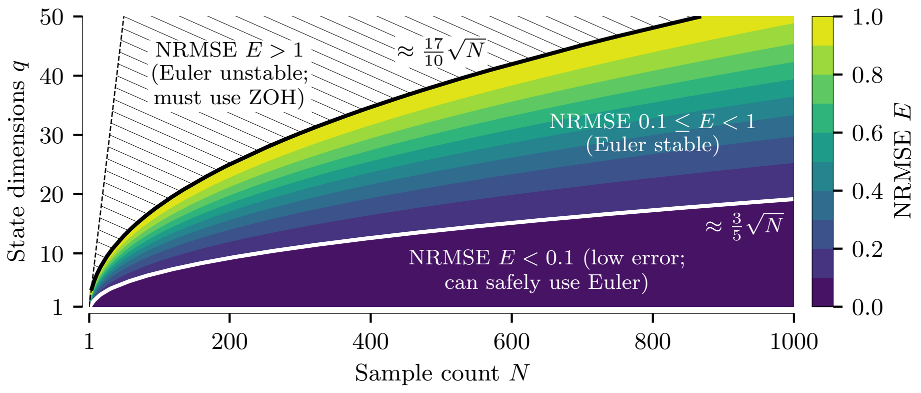

Figure 9: Normalised RMSE between the matrix $\mathbf{H}$ computed using zero-order hold discretisation (eq. 28) and a $\mathbf{H}'$ computed using Euler's method (eq. 25). For Euler's method to yield NRMSEs below 0.1, N must be larger than $\frac{25}{9}q^2$.

case, each sample u_i will be presented for exactly one timestep, and we need to derive an expression for the state vector at timestep N, i.e., $\mathbf{m}_N$.

Let $\mathbf{m}_0 = 0$. Using Euler integration, we get the recurrence relation

$$\mathbf{m}_{i+1} = \mathbf{m}_i + \Delta t \left(\frac{\mathbf{A}\mathbf{m}_i}{\theta} + \frac{\mathbf{B}}{\theta} u_i \right) = \frac{1}{N}\big((\mathbf{A} + N\mathbf{I})\mathbf{m}_i + \mathbf{B}u_i\big) \,. \qquad (25)$$

This equation is implemented in ***`mk_ldn_basis_euler`***.

Unfortunately, using an Euler integrator in this manner without finer-grained update steps is generally a bad idea. Judging from numerical experiments (fig. 9), it must approximately hold $N > 2.78q^2$ for Euler's method to not introduce large errors. If $N < 0.35q^2$, this method will diverge within the first N timesteps.

Closed-form solution with zero-order hold assumption There is a relatively simple solution to this problem that is in the spirit of the above idea. Since our system is purely linear, we can advance the system Δt seconds into the future given an initial state $\mathbf{m}(t)$, under the condition that $u(t)$ stays constant for the next Δt seconds. This is exactly how we defined $u(t)$ in eq. (24) if $\theta = N\Delta t$. In general, assuming that $u(t)$ is a stair-step function is called a "zero-order hold assumption". We obtain a new recurrence relation that uses a matrix exponential

$$\begin{aligned}\mathbf{m}_i &= \tilde{\mathbf{A}}\mathbf{m}_{i-1} + \tilde{\mathbf{B}}u_i \,, \\ \text{where } \tilde{\mathbf{A}} &= \exp\left(\Delta t \frac{\mathbf{A}}{\theta}\right) = \exp\left(\frac{\mathbf{A}}{N}\right) \,, \\ \tilde{\mathbf{B}} &= \mathbf{A}^{-1}(\tilde{\mathbf{A}} - \mathbf{I})\mathbf{B} \,.\end{aligned} \qquad (26)$$

This equation is implemented in the function ***`discretize_lti`***.

This is a standard technique for the discretisation of LTI systems (cf. Voelker,

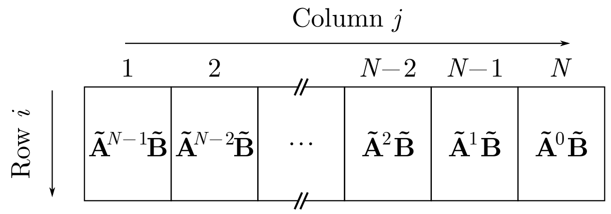
Column j
1
2
$N-2$
$N-1$
N
Row i
$\tilde{\mathbf{A}}^{N-1}\tilde{\mathbf{B}}$
$\tilde{\mathbf{A}}^{N-2}\tilde{\mathbf{B}}$
$\cdots$
$\tilde{\mathbf{A}}^{2}\tilde{\mathbf{B}}$
$\tilde{\mathbf{A}}^{1}\tilde{\mathbf{B}}$
$\tilde{\mathbf{A}}^{0}\tilde{\mathbf{B}}$

Figure 10: Illustration of the unnormalised LDN basis transformation matrix $\mathbf{H}'$ (eq. 28). The same principle can be applied to other LTI systems.

2019, Section 5.1.1, p. 100). Expanding the recurrence relation for $\mathbf{m}_N$ we get

$$\begin{aligned}
\mathbf{m}_N &= \tilde{\mathbf{A}}\mathbf{m}_{N-1} + \tilde{\mathbf{B}}u_N \\
&= \tilde{\mathbf{A}}\big(\tilde{\mathbf{A}}\mathbf{m}_{N-2} + \tilde{\mathbf{B}}u_{N-1}\big) + \tilde{\mathbf{B}}u_N \\
&= \tilde{\mathbf{A}}^2\mathbf{m}_{N-2} + \tilde{\mathbf{A}}\tilde{\mathbf{B}}u_{N-1} + \tilde{\mathbf{B}}u_N \\
&= \tilde{\mathbf{A}}^3\mathbf{m}_{N-3} + \tilde{\mathbf{A}}^2\tilde{\mathbf{B}}u_{N-2} + \tilde{\mathbf{A}}\tilde{\mathbf{B}}u_{N-1} + \tilde{\mathbf{B}}u_N \\
&= \ldots \\
&= \sum_{i=1}^{N} \tilde{\mathbf{A}}^{N-i}\tilde{\mathbf{B}}u_i = \sum_{i=0}^{N-1} \tilde{\mathbf{A}}^{i}\tilde{\mathbf{B}}u_{N-i}\,. \qquad (27)
\end{aligned}$$

As desired, this is a linear equation. We can write this sum in terms of a matrix-vector product $\mathbf{H}'\mathbf{u}$, where $\mathbf{H}' \in \mathbb{R}^{q\times N}$ is an unnormalised basis transformation matrix. The kth column of $\mathbf{H}'$, denoted $\big(\mathbf{H}'^T\big)_k$ is simply given as (cf. fig. 10)

$$\big(\mathbf{H}'^T\big)_k = \tilde{\mathbf{A}}^{N-k}\tilde{\mathbf{B}}\,. \qquad (28)$$

The time-complexity of evaluating this matrix is in $\mathcal{O}(q^2N)$. Importantly, and in contrast to all other discrete function bases discussed so far, the corresponding discrete function basis $H_n'^q(k;N) = \mathbf{H}'_{n,k}$ depends on the state-dimensionality q. In other words, when q is changed, all q discrete basis functions change; in general, $H_n^q(k;N) \neq H_n^{q'}(k;N)$.

Qualitative comparison to the Legendre and DLOP bases The normalised function basis transformation matrix $\mathbf{H}$ is depicted in Figure 11. $\mathbf{H}$ is *almost* orthogonal and is similar to both the DLOP and the Legendre basis in some respects. The left half of $\mathbf{H}$ somewhat resembles DLOPs, whereas the right half is very similar to the discrete Legendre basis from eq. (16).

Equivalence to the mean-sampled impulse response Applying mean sampling as defined in eq. (13) to the LDN system impulse response for $\theta = 1$

$$H_n^q(k;N) = \frac{\sqrt{N\theta}}{\sqrt{2n+1}} \int_a^b \big(e^{\mathbf{A}t}\mathbf{B}\big)_n \,\mathrm{d}t\,, \quad \text{where } a = \frac{k}{N}, \quad b = \frac{k+1}{N}\,.$$

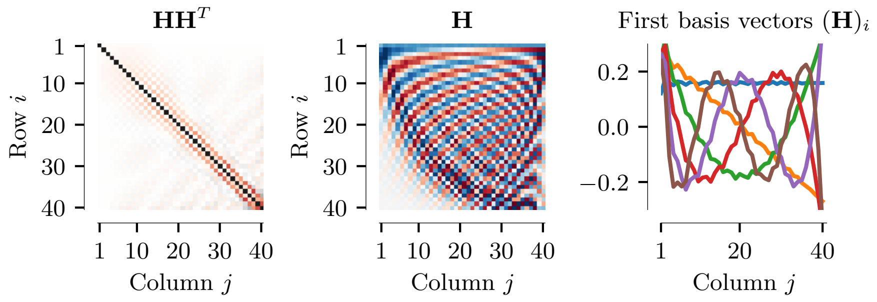

Figure 11: Visualisation of the LDN basis as defined in eq. (28) for $q = N = 40$. See Figure 4 for the complete legend and a description of the colour scheme.

The integral of a matrix exponential is (cf. DeRusso et al., 1998, pp. 171-172)

$$\int_0^b e^{\mathbf{A}t}\,\mathrm{d}t = \big(e^{\mathbf{A}b} - \mathbf{I}\big)\mathbf{A}^{-1} \quad \Rightarrow \quad \int_a^b e^{\mathbf{A}t}\,\mathrm{d}t = \big(e^{\mathbf{A}b} - e^{\mathbf{A}a}\big)\mathbf{A}^{-1}\,.$$

Abbreviating the normalisation term as γ and factoring out $\mathbf{B}$ we get

$$\begin{aligned} H_n^q(k;N) &= \gamma\Big(\Big(\int_a^b e^{\mathbf{A}t}\,\mathrm{d}t\Big)\mathbf{B}\Big)_n = \gamma\Big(\Big(e^{\mathbf{A}b} - e^{\mathbf{A}a}\Big)\mathbf{A}^{-1}\mathbf{B}\Big)_n \\ &= \gamma\Big(\Big(\tilde{\mathbf{A}}^{k+1} - \tilde{\mathbf{A}}^{k}\Big)\mathbf{A}^{-1}\mathbf{B}\Big)_n = \gamma\Big(\Big(\tilde{\mathbf{A}}^{k}(\tilde{\mathbf{A}} - \mathbf{I}\big)\mathbf{A}^{-1}\mathbf{B}\Big)_n\,. \end{aligned}$$

The exponential of matrix $e^{\alpha\mathbf{A}}$ and its inverse $\mathbf{A}^{-1}$ are commutative (cf. DeRusso et al., 1998, p. 170 for the definition of the matrix exponential)

$$\mathbf{A}^{-1}e^{\alpha\mathbf{A}} = \mathbf{A}^{-1}\sum_{i=0}^{\infty}\frac{\mathbf{A}^i\alpha^i}{i!} = \sum_{i=0}^{\infty}\frac{\mathbf{A}^{i-1}\alpha^i}{i!} = \sum_{i=0}^{\infty}\frac{\mathbf{A}^i\alpha^i}{i!}\mathbf{A}^{-1} = e^{\alpha\mathbf{A}}\mathbf{A}^{-1}\,.$$

Hence, we can move the $\mathbf{A}^{-1}$ to the left-hand side of the term $\tilde{\mathbf{A}} - \mathbf{I}$. We get

$$H_n^q(k;N) = \gamma\Big(\Big(\tilde{\mathbf{A}}^{k}\mathbf{A}^{-1}(\tilde{\mathbf{A}} - \mathbf{I})\mathbf{B}\Big)_n = \gamma\Big(\tilde{\mathbf{A}}^{k}\tilde{\mathbf{B}}\Big)_n\,. \tag{29}$$

Scaling and ordering (k and not $N - k$, cf. eq. 11) aside, this is exactly eq. (28).

4.3 When to Use the LDN Basis

Up to this point, it may seem as if the LDN was merely a convoluted way to construct a discrete function basis that resembles the Legendre polynomials. So, why—compelling connections to biology aside (Voelker and Eliasmith, 2018)—should we care about the LDN system at all, and not just use exactly orthogonal discrete function basis such as DLOPs?

The answer to this is not clear-cut. The gist is that the LDN is the optimal online, zero-delay (or "sliding") transformation that weighs each point in time equally and only requires $\mathcal{O}(q)$ *state* memory (see Gu et al., 2020, for the "equal weight" aspect). Still, there are some trade-offs that are worth discussing.

Sliding transformations As mentioned at the beginning of this section, one way to think about the LDN system is as a means to convolve an input signal $u(t)$ with the Legendre polynomials at every point in time t, or, in other words, to compute the generalised Fourier coefficients $\xi_n(t)$ *online*; i.e.,

$$\xi_n(t) = \left\langle u_{[t-\theta,t]}, \tilde{m}_n^q \right\rangle ,$$

where $u_{[t-\theta,t]} : [0, \theta] \longrightarrow \mathbb{R}$ corresponds to a function representing the past θ seconds of the input u, and $\tilde{m}_n^q$ is as defined in eq. (23). That is, by simply advancing a q-dimensional LTI system for an input $u(t)$, the generalised Fourier coefficients are stored in the momentary LTI system state $\mathbf{m}(t)$.

A transformation that is evaluated at every point in time over a window of the input history is also called a *sliding transformation*. Most of the discrete transformations we discussed so far have sliding versions; examples being the sliding discrete Fourier (SDFT; Jacobsen and Lyons, 2003), cosine (SCT; Kober, 2004) and Haar transformations (Kaiser, 1998). These "classic" sliding transformations mandate that a window $u_{[t-\theta,t]}$ is kept in memory, i.e., $\mathcal{O}(N)$ memory in the discrete case. Each update step requires only $\mathcal{O}(q)$ operations.

Efficiency of the LDN compared to FIR filters In contrast, the discrete LDN LTI system only requires $\mathcal{O}(q)$ *state* memory (in addition to storing the $q \times q$ matrix $\tilde{\mathbf{A}}$) and can be advanced using eq. (26). Each update requires $\mathcal{O}(q^2)$ operations and yields the updated discrete generalised Fourier coefficients $\mathbf{m}_t$.

For bases where we do not have an efficient sliding transformation (as, for example, for DLOPs), we must treat the basis transformation matrix $\mathbf{E}$ as a set of q FIR filters (cf. eq. 11). Filtering a signal $\mathbf{u}$ with q FIR filters requires holding the past N input samples in memory in addition to the $q \times N$ matrix $\mathbf{E}$. When done naively, convolution with the filters requires $\mathcal{O}(qN)$ operations in every timestep. Hence, updating the discrete LDN system is more efficient than repeated convolution if $q < N$.

Efficient zero-delay FIR filtering The above characterization is a bit misleading. The time complexity of $\mathcal{O}(qN)$ for online convolution of a signal with a set of q FIR filters only applies to the naive algorithm. In general, this operation can be performed using $\mathcal{O}(q \log(N))$ operations and $\mathcal{O}(N \log(N))$ memory (amortized; cf. Gardner, 1995). Note that the corresponding algorithm has a relatively large constant scaling factor of approximately 34 in the number of operations (cf. Section 5.2, p. 132 of Gardner, 1995).

This means that—much lower memory requirements aside—using eq. (26) to compute the discrete generalised Fourier coefficients $\mathbf{m}_t$ of the LDN basis is still attractive when compressing a large number of samples N into relatively few dimensions q. To be precise, using an LTI system to implement a sliding transformation is more efficient as long as $q < 34 \log_2(N)$. This relationship is depicted in Figure 13. As a rule of thumb, and assuming that memory is not a constraint, FIR filters should be used if $N \geq q > 300$.

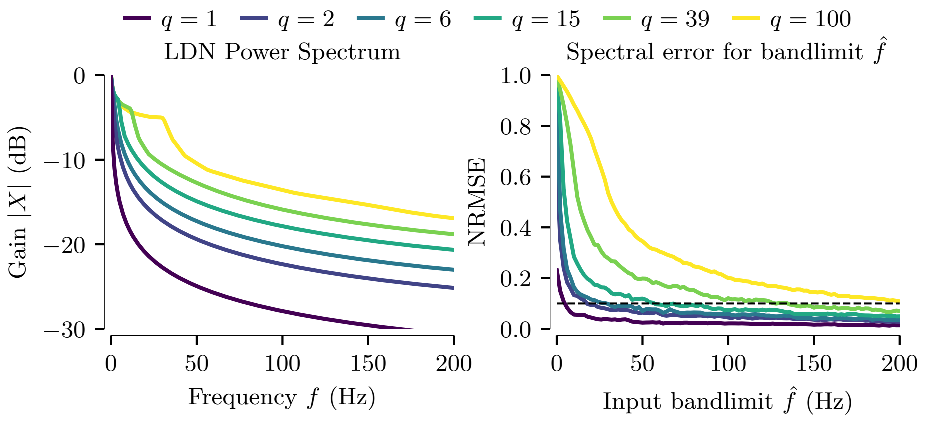

Figure 12: LDN impulse response power spectrum and low-pass characteristics. *Left:* Power spectrum for different q assuming $\theta = 1\,\mathrm{s}$; the LDN system acts as a low-pass. *Right:* Error $\mathbf{Hu} - \mathbf{H\hat{u}}$ for a white-noise signal $\mathbf{u}$ and a band-limited version $\mathbf{\hat{u}}$ thereof with band-limit $\hat{f}$. It must approximately hold $\hat{f} > 2q$ for the NRMSE to not surpass 0.1 (dashed line).

Fast Euler update Requiring $\mathcal{O}(q^2)$ operations per update step and $\mathcal{O}(q^2)$ of memory in total for the zero-order hold discrete LDN system may be a little disappointing. After all, the standard sliding transformations mentioned above only require $\mathcal{O}(q)$ operations per update steps in exchange for $\mathcal{O}(N)$ memory.

In fact, Voelker (2019, Figure 6.6) shows that the LDN can be advanced using only $\mathcal{O}(q)$ operations per step and $\mathcal{O}(q)$ *total* memory when using the Euler update from (cf. eq. 25). This low memory requirement can be seen as the *defining property of the LDN.* After all, the LDN was derived from first principles to represent a window of data using q state variables over time.

Limitations of the fast Euler update While the low memory requirements of the Euler method can be important, the time-complexity $\mathcal{O}(q)$ comes with a caveat. Remember from above that the Euler update is only feasible approximately if $N > 2.78q^2$. Consequently, and as we will explain in the following, both the $\mathcal{O}(q^2)$ zero-order hold algorithm and the $\mathcal{O}(q)$ Euler update have the same asymptotic time-complexity in practice under the condition that the input signal $u(t)$ is appropriately band-limited and sampled near the Nyqist limit.[5]

To see this, consider a q-dimensional LDN. As depicted in the left half of Figure 12, the LDN system acts as a low-pass filter. That is, higher frequencies in the input signal barely have an influence on the output of the system. The high frequencies can even be filtered out completely without changing the generalised LDN Fourier coefficients (the "LDN spectrum") much.

[5]Down-sampling to the band-limit introduces latency, which is not desirable in real-time control applications; the $\mathcal{O}(q)$ Euler update is optimal in this case.

This is depicted in the right half of Figure 12. Band-limiting a signal $u(t)$ to a maximum frequency $\hat{f} > 2q$ results in an NRMSE of at most 0.1 in the LDN spectrum, even if $u(t)$ is a white noise-signal. A discrete representation of this band-limited signal would require $N > 2\hat{f} = 4q$ samples per second according to Nyquist-Shannon (we discuss this in more detail in Section 6.1). The q^2 zero-order hold update algorithm then requires $4q^3$ operations per second, plus a low-pass filter, which can be cheaply implemented as a short FIR filter on $u(t)$.

The Euler update on the other hand requires at least $N = 2.78q^2$ samples to reach an NRMSE of 0.1, resulting in a total of $2.78q^3$ operations per second. While this is a little smaller than $4q^3$, remember that the estimate $N > 2\hat{f} = 4q$ was derived under very conservative circumstances (i.e., a white noise signal as an input); lower band-limits can likely be used in practice.

Sliding transformations and neural networks In the context of neural networks, we can use basis transformations in their FIR filter representation for training. This avoids recurrences and can increase throughput, particularly when training the network on GPUs.

During inference, a more efficient update rule can be used for the sliding transformation. A summary of the memory and run-time costs for the various transformations discussed in this report is given in Table 1.

As a side note, Gu et al. (2020) derive sliding transformations similar to the LDN LTI system for different polynomial bases and weightings. Our discussion of the LDN in terms of run-time and memory applies to these systems as well, although the low-pass filter characteristics may be different.

5 LTI Systems From Discrete Function Bases

In the previous section, we derived a basis transformation matrix $\mathbf{E}$ from the LDN system $\mathbf{A}$, $\mathbf{B}$. We also saw that the LDN system can be used to directly compute discrete generalised Fourier coefficients $\mathbf{m}_t$; all we need to do is to simply advance the LTI system for each input sample u_t. For small q this can be more efficient than repeated convolution with the past N input samples.

Of course, this raises the question whether we can reverse what we did above; in other words, derive an LTI system $\mathbf{A}$, $\mathbf{B}$ from arbitrary discrete function bases $\mathbf{E}$. This problem is known in the literature as a "system identification problem". See Verhaegen and Verdult (2007) for a thorough treatise; the solution presented in this report is as crude as it is simple.

5.1 Solving For $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$ Using Least Squares

In the last section, we constructed the unnormalised transformation matrix $\mathbf{E}$ as

This method is implemented in `reconstruct_lti`.

$$\big(\mathbf{E}^T\big)_k = \tilde{\mathbf{A}}^{N-k}\tilde{\mathbf{B}}\,, \quad \text{and correspondingly } \big(\mathbf{E}^T\big)_N = \tilde{\mathbf{B}}\,, \quad \big(\mathbf{E}^T\big)_i = \tilde{\mathbf{A}}\big(\mathbf{E}^T\big)_{i-1}\,,$$

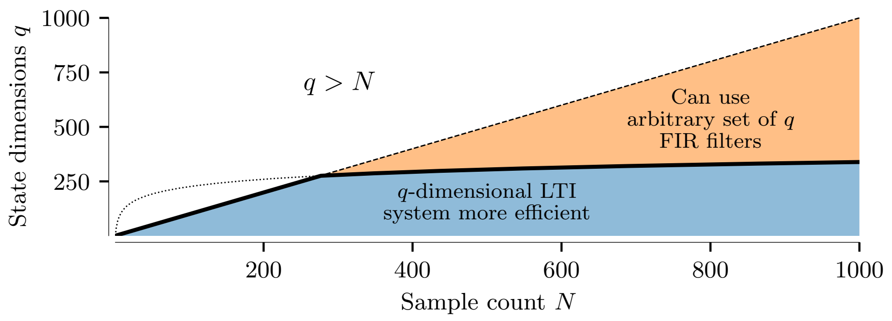

Figure 13: Combinations of state dimensions q, and sample counts N for which a suitable LTI system (such as the LDN) can compute discrete generalised Fourier coefficients $\mathbf{m}_t$ more efficiently compared to a FIR filter. Advancing an LTI system requires on the order of q^2 operations, whereas online evaluation of q FIR filters requires about $34\,q\log_2(N)$ operations per timestep (Gardner, 1995).

Table 1: Run-time and memory costs for performing discrete function basis transformations of order q and filter length N (generally we assume $q \leq N$; for the batch FFT, FCT and FHT $q = N$). "Batch" describes transforming a N samples into a different basis at once. "Sliding/Online" corresponds to computing the basis transformation for each incoming sample, where N is the length of the filter. All memory costs include constant matrices.

		Batch (N samples)		**Sliding/Online** (per sample)	
Basis	*Algorithm*	*Run-time*	*Memory*	*Run-time*	*Memory*
FIR	Naive	$\mathcal{O}(qN^2)$	$\mathcal{O}(qN)$	$\mathcal{O}(qN)$	$\mathcal{O}(qN)$
	FFT conv.	$\mathcal{O}(qN\log N)$	$\mathcal{O}(qN)$	/	/
	Gardner[1]	/	/	$\mathcal{O}(q\log N)$	$\mathcal{O}(qN\log N)$
Haar	FHT[2]	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(q)$	$\mathcal{O}(N)$
Fourier	FFT[3]	$\mathcal{O}(N\log N)$	$\mathcal{O}(N)$	/	/
	SDFT[4]	$\mathcal{O}(qN)$	$\mathcal{O}(N)$	$\mathcal{O}(q)$	$\mathcal{O}(N)$
Cosine	FCT[5]	$\mathcal{O}(N\log N)$	$\mathcal{O}(N)$	/	/
	SCT[6]	$\mathcal{O}(qN)$	$\mathcal{O}(N)$	$\mathcal{O}(q)$	$\mathcal{O}(N)$
LDN	ZOH LTI	$\mathcal{O}(q^2N)$	$\mathcal{O}(N+q^2)$	$\mathcal{O}(q^2)$	$\mathcal{O}(q^2)$
	Euler LTI[7]	$\mathcal{O}(qN)$	$\mathcal{O}(N)$	$\mathcal{O}(q)$	$\mathcal{O}(q)$

"FIR" corresponds to an arbitrary set of FIR filters. "ZOH LTI" refers to the zero-order-hold discrete LTI system (eq. 28). [1] Gardner, 1995; [2] Kaiser, 1998; [3] Cooley and Tukey, 1965; [4] Jacobsen and Lyons, 2003; [5] Makhoul, 1980; [6] Kober, 2004; [7] Voelker, 2019, Figure 6.6.

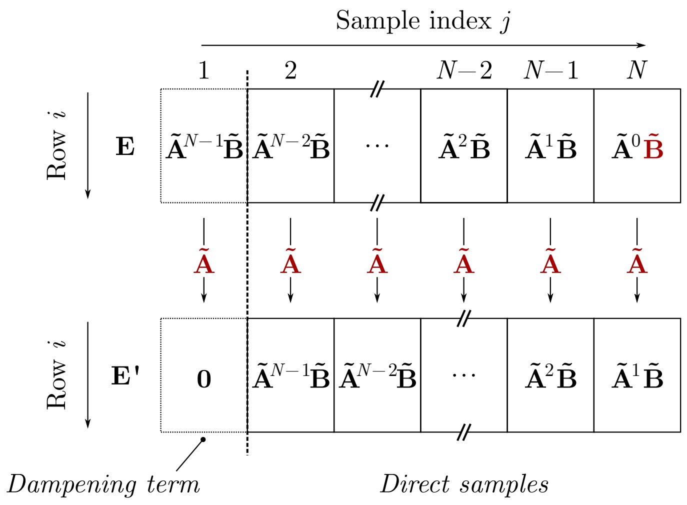
Sample index j
1
2
$N-2$
$N-1$
N
Row i
$\mathbf{E}$
$\tilde{\mathbf{A}}^{N-1}\tilde{\mathbf{B}}$
$\tilde{\mathbf{A}}^{N-2}\tilde{\mathbf{B}}$
$\cdots$
$\tilde{\mathbf{A}}^{2}\tilde{\mathbf{B}}$
$\tilde{\mathbf{A}}^{1}\tilde{\mathbf{B}}$
$\tilde{\mathbf{A}}^{0}\tilde{\mathbf{B}}$
$\tilde{\mathbf{A}}$
Row i
$\mathbf{E}'$
$\mathbf{0}$
$\tilde{\mathbf{A}}^{N-1}\tilde{\mathbf{B}}$
$\tilde{\mathbf{A}}^{N-2}\tilde{\mathbf{B}}$
$\cdots$
$\tilde{\mathbf{A}}^{2}\tilde{\mathbf{B}}$
$\tilde{\mathbf{A}}^{1}\tilde{\mathbf{B}}$
Dampening term
Direct samples

Figure 14: Constructing a discrete LTI system $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$ from any discrete basis transformation matrix $\mathbf{E}$. $\tilde{\mathbf{B}} \in \mathbb{R}^{q \times 1}$ is the last column of $\mathbf{E}$. $\tilde{\mathbf{A}}$ can be obtained by solving $\mathbf{E}'\tilde{\mathbf{A}} = \mathbf{E}$. Dampening encourages a finite impulse response.

where k is the kth column of $\mathbf{E}$ and $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$ are as defined in eq. (26). We can directly read $\tilde{\mathbf{B}}$ off $\mathbf{E}$, and estimate $\tilde{\mathbf{A}}$ by solving for a matrix that translates between individual columns using least squares. This is illustrated in Figure 14 (ignore dampening for now). Reverting discretisation yields an LTI system $\mathbf{A}, \mathbf{B}$:

$$\mathbf{A} = \frac{N}{\theta} \log(\tilde{\mathbf{A}}) \,, \qquad \mathbf{B} = \frac{1}{\theta}(\tilde{\mathbf{A}} - \mathbf{I})^{-1}\mathbf{A}\tilde{\mathbf{B}} \,. \tag{30}$$

One caveat with this approach is that the discrete basis transformation matrix $\mathbf{E}$ must be at least of size $q \times (q+1)$, the total number of degrees of freedom in $\tilde{\mathbf{A}} \in \mathbb{R}^{q \times q}$ and $\tilde{\mathbf{B}} \in \mathbb{R}^{q \times 1}$. Methods for solving for $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$ with fewer degrees of freedoms exist (see Verhaegen and Verdult, 2007, Chapter 7 onward).

Description of the resulting LTI systems The left half of Figure 15 depicts feedback matrices $\mathbf{A}$ and impulse responses for LTI systems constructed from the transformation matrices discussed so far. We now briefly discuss each system.

LDN system. Notably, the LDN system is—apart from some scaling factors due to regularisation—perfectly reconstructed. This should not surprise; our reconstruction method reverts the steps taken to construct $\mathbf{H}$ from $\mathbf{A}$, $\mathbf{B}$.

DLOP system. The DLOP LTI system perfectly generates the Legendre polynomials over the interval $[0, 1]$ as its impulse response. Unfortunately, being able to do so is rather pointless, since the impulse response quickly diverges for $t > 1$. Notice how the DLOP system $\mathbf{A}$ matrix is strictly positive. The positive coefficients in the LDN system matrix are very similar to the positive DLOP

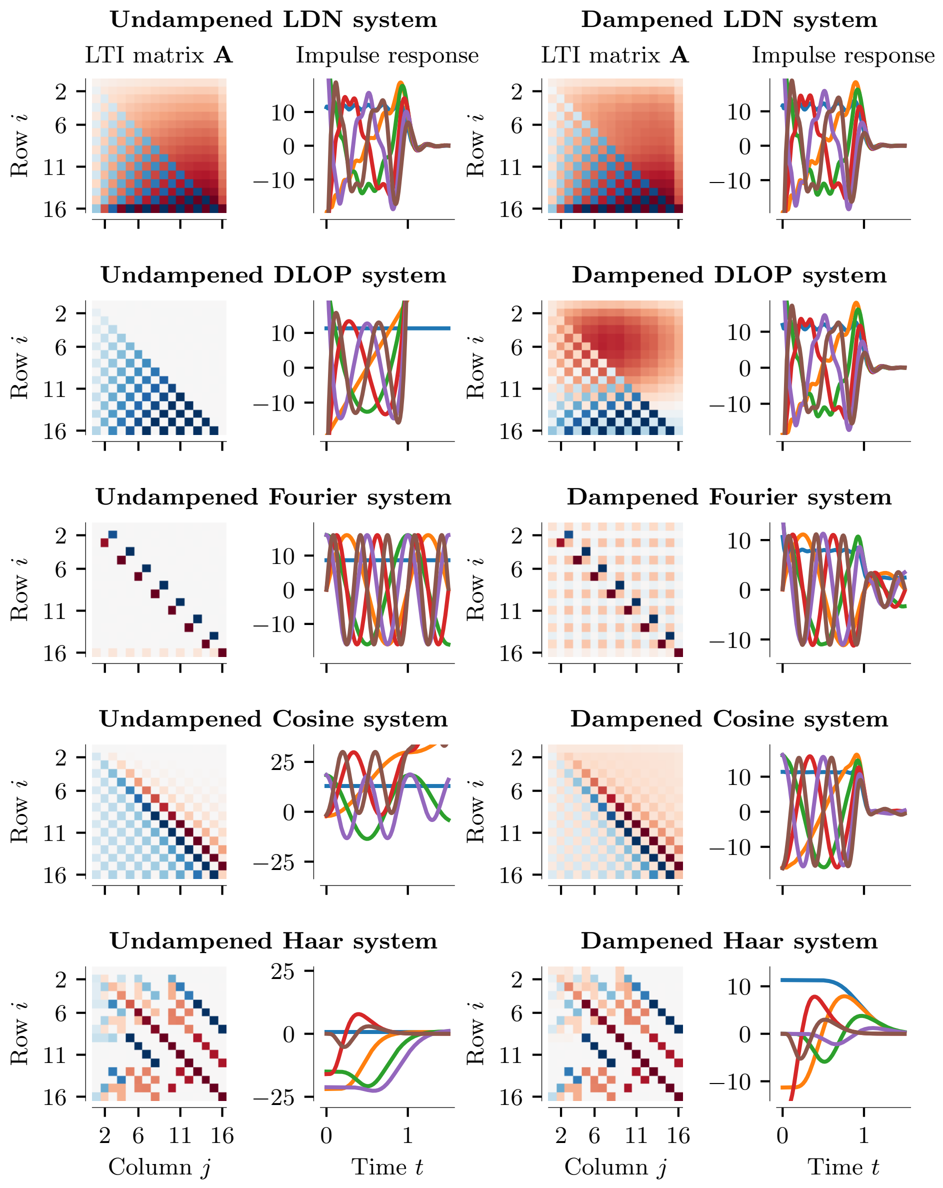

Figure 15: Constructing LTI systems from discrete function bases using least squares. Parameters are $q = 16$, $N = 128$, and $\theta = 1$. *Left half:* LTI systems reconstructed without dampening. *Right half:* LTI systems reconstructed with dampening. *Left:* Feedback matrix $\mathbf{A}$. Positive values are blue, negative values red. Colour map is rescaled such that 95% of the values are depicted without saturation. *Right:* Impulse response of the first six state dimensions.

system coefficients; hence, the LDN system can be thought of implementing the Legendre basis with some dampening through the added negative coefficients.

Fourier system. As one would expect, the Fourier system consists of $\frac{N}{2}$ independent oscillators with increasing frequency. Just like the DLOP system, the Fourier system has an infinite impulse response, but unlike the DLOP system, the system is stable.

Cosine system. Apart from some scaling and shifting issues, the cosine system reconstructs the cosine basis over the interval $[0, 1]$. For $t > 1$ the impulse response no longer resembles the cosine basis.

Haar system. We cannot expect a finite-dimensional LTI system to reproduce the discontinuities in the Haar basis. In contrast to the other systems, the Haar system has a finite impulse response for $q = 16$, but this is not guaranteed.

5.2 Dampening the Reconstructed System

Most of the reconstructed systems have an infinite impulse response. This is not desirable unless the system is never advanced beyond θ, or, in other words, only a fixed number of samples is processed.

In this report, we are more concerned with online processing, which implies a potentially unbounded number of samples and the inability to perform batch updates. To be suitable for online processing, the LTI system must not only optimally approximate a function basis, but also "optimally forget" information originating from more than θ seconds in the past.

In the following, we discuss two extensions to the above method, that encourage, but do not guarantee, a finite impulse response. The first method adds a "dampening" term to the least-squares equations. The second method explicitly erases information older than θ seconds from the state $\mathbf{m}$.

Dampening term in the least-squares system One way to encourage a finite impulse response is to add the equation $\tilde{\mathbf{A}}(\mathbf{E}^T)_1 = \mathbf{0}$ to the least-squares problem. As illustrated in Figure 14, multiplying the first column of the discrete function basis (which corresponds to $t = \theta$) with $\tilde{\mathbf{A}}$ should extinguish the impulse response. This dampening term should be weighted by $\frac{N-1}{q-1}$, maintaining a weight ratio of $1 : q - 1$ between the dampening term and the remaining samples.

This method is implemented in `reconstruct_lti` when passing `dampen = "lstsq"` as a parameter.

The least-squares system now contains exactly N samples, so, in theory, it could be used for matrices $\mathbf{E}$ of shape $q \times N$ with $N = q$. However, in practice, this often results in a singular $(\tilde{\mathbf{A}} - \mathbf{I})$ or non positive definite $\tilde{\mathbf{A}}$, for which the matrix logarithm cannot be computed.

This least-squares approach to system identification may not lead to optimal results. It may be beneficial to iteratively enforce the dampening condition using the actual impulse response of the reconstructed system.

Dampening by information erasure For our system to have a finite impulse response, we must ensure that information older than θ seconds (or N samples) is "forgotten". We can enforce this by *decoding* the input u_{t-N+1} from the current state $\mathbf{m}_t$ and computing the state vector $\mathbf{m}'_t$ *encoding* this input. Subtracting $\mathbf{m}'_t$ from $\mathbf{m}_t$ erases u_{t-N+1} from the state. Hence, this "old" sample can no longer influence the system state and the impulse response must be finite.

This method is implemented in `reconstruct_lti` *when passing* `dampen = "erasure"` *as a parameter.*

In practice, assuming that the discrete LTI system $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$ reconstructs the discrete function basis $\mathbf{E}$, the sample u_{t-N+1} can be decoded as follows

$$u_{t-N+1} = \bigl(\mathbf{E}^{+}\bigr)_1 \mathbf{m}_t \,, \qquad \text{where } \mathbf{E}^{+} = \bigl(\mathbf{E}^T\mathbf{E}\bigr)^{-1}\mathbf{E}^T \,,$$

and $\bigl(\mathbf{E}^{+}\bigr)_1$ denotes the first row of the pseudo-inverse. Re-encoding u_{t-N+1} under a zero-order hold assumption yields

$$\mathbf{m}'_t = \bigl(\mathbf{E}^T\bigr)_1 u_{t-N+1} = \bigl(\mathbf{E}^T\bigr)_1 \otimes \bigl(\mathbf{E}^{+}\bigr)_1 \mathbf{m}_t \,,$$

where $\otimes$ is the outer product and $\bigl(\mathbf{E}^T\bigr)_1$ is the first column of $\mathbf{E}$. Sequentially interleaving an "update" and an "erasure" step we obtain

$$\begin{aligned}
\mathbf{m}_t &\leftarrow \tilde{\mathbf{A}}\mathbf{m}_{t-1} + \tilde{\mathbf{B}}u_t \,, && \text{(Update)} \\
\mathbf{m}_t &\leftarrow \mathbf{m}_t - \mathbf{m}'_t = \bigl(\mathbf{I} - \bigl(\mathbf{E}^T\bigr)_1 \otimes \bigl(\mathbf{E}^{+}\bigr)_1\bigr)\mathbf{m}_t \,. && \text{(Erasure)} \\
&= \underbrace{\bigl(\mathbf{I} - \bigl(\mathbf{E}^T\bigr)_1 \otimes \bigl(\mathbf{E}^{+}\bigr)_1\bigr)\tilde{\mathbf{A}}}_{\tilde{\mathbf{A}}'}\, \mathbf{m}_t + \underbrace{\bigl(\mathbf{I} - \bigl(\mathbf{E}^T\bigr)_1 \otimes \bigl(\mathbf{E}^{+}\bigr)_1\bigr)\tilde{\mathbf{B}}}_{\tilde{\mathbf{B}}'}\, u_t \,.
\end{aligned}$$

Applying the inverse discretisation from eq. (30) to $\tilde{\mathbf{A}}'$, $\tilde{\mathbf{B}}'$ results in a dampened, continuous LTI system. Again, this is not an optimal solution, since the equations assume that $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ perfectly reconstruct the discrete function basis over $[0, \theta]$.

A continuous version of this method can be used to directly derive the LDN system from the Legendre polynomials (Stöckel, 2021).

Description of the dampened LTI systems Surprisingly, both dampening methods yield similar results—at least with respect to the impulse response of the resulting systems. In the following, we discuss the results obtained when using the information erasure method. The right half of Figure 15 depicts the impulse responses of the dampened LTI systems and the corresponding feedback matrices $\mathbf{A}$. As before, we will quickly comment on these for each discrete function basis.

Dampened LDN system. The LTI system and its impulse response remain almost unchanged. This includes the higher order dimensions (not depicted).

Dampened DLOP system. The dampened DLOP system has—apart from a different scaling of the individual systems—*almost* exactly the same impulse response as the LDN system. For all practical purposes, the LDN and dampened DLOP system seem equivalent.

Dampened Fourier system. Dampening the Fourier system leads to some reduction in magnitude for $t > \theta$. While the impulse response eventually does decay to zero, it does so very slowly.

Dampened Cosine system. Especially the depicted lower order terms of the dampened cosine system decay to zero quite quickly; however, this is not true for the higher order terms (not depicted), which behave more like the dampened Fourier system. Notice that the impulse response somewhat resembles the Legendre basis.

Dampened Haar system. The dampened Haar system behaves well; though it is not clear whether the impulse response resembles a sensible function basis.

5.3 Summary

The technique we presented in this section could be improved by parametrising $\mathbf{A}$, $\mathbf{B}$ in a way that enforces stability and results in well-conditioned systems. Nevertheless, our naive least-squares solution produces LTI systems that very well approximate a discrete function basis $E_n(k; N)$ over an interval $[0, \theta]$.

However, the true challenge lies in generating an LTI system that has a "finite impulse response", or, in other words, quickly converges to zero for $t > \theta$. While introducing a dampening term accomplishes this to some degree, most of the systems that end up with a finite impulse response (LDN, DLOP, partially the Cosine system) resemble the Legendre polynomials.

It is unclear whether LTI systems approximating bases other than the Legendre polynomials are really necessary. If so desired, similar approximations of other bases can be constructed by equipping the LDN system with a corresponding readout matrix $\mathbf{T} \in \mathbb{R}^{q \times q}$. Due to linearity, this generally yields results superior to what was shown here in terms of the finiteness of impulse response.

6 Low-pass Filtering Discrete Function Bases

If the number of discrete basis functions q is smaller than the number of samples N, then a discrete basis transformation $\mathbf{E} \in \mathbb{R}^{q \times N}$ is a *projection* mapping from a higher- onto a lower-dimensional space. Computing $\mathbf{m} = \mathbf{E}\mathbf{u}$ can be thought of as "packing" N samples $\mathbf{u}$ into a smaller q-dimensional vector $\mathbf{m}$. In general, projections are "lossy", that is, they decrease the amount of information present in $\mathbf{m}$ compared to $\mathbf{u}$.

In this section, we talk about the conditions under which information loss occurs, leading us to the Nyquist-Shannon sampling theorem discussed in Section 6.1, as well as a simple related lemma that can be applied to discrete function bases. Violating the conditions in the Nyquist-Shannon sampling theorem or the lemma results in a phenomenon called "aliasing". We discuss this in Section 6.2. We close with a technique for constructing discrete function bases that apply an anti-aliasing filter to $\mathbf{u}$ in Section 6.3. It is not immediately clear whether this is necessary or beneficial in the context of discrete function bases.

6.1 The Nyquist-Shannon Sampling Theorem

In most cases, projections such as $\mathbf{E}$ with $q < N$ inadvertently destroy information. That is, a signal $\mathbf{u}$ cannot be reconstructed from $\mathbf{Eu}$. An exception are signals $\mathbf{u}$ that are a linear combination of the q rows of the projection matrix $\mathbf{E}$, i.e., $\mathbf{u} = \mathbf{E}^T\mathbf{m}$, where $\mathbf{m} \in \mathbb{R}^q$. We express this formally in the lemma below.

Lemma 1. *Let* $\mathbf{E} \in \mathbb{R}^{q\times N}$ *with* $q \leq N$*,* $\mathbf{EE}^T = \mathbf{I}$*, and* $\mathbf{u} \in \mathbb{R}^N$*. It holds* $\mathbf{E}^T\mathbf{Eu} = \mathbf{u}$ *exactly if a unique* $\mathbf{m} \in \mathbb{R}^q$ *exists such that* $\mathbf{u} = \mathbf{E}^T\mathbf{m}$*.*

Proof. Consider the "$\Leftarrow$" direction. Let $\mathbf{m} \in \mathbb{R}^q$ and $\mathbf{u} = \mathbf{E}^T\mathbf{m}$. We need to show $\mathbf{E}^T\mathbf{Eu} = \mathbf{u}$. Expanding the left-hand side of the latter expression we get

$$\mathbf{E}^T\mathbf{Eu} = \mathbf{E}^T\mathbf{EE}^T\mathbf{m} = \mathbf{E}^T\mathbf{m} = \mathbf{u}\,. \checkmark$$

Now, consider the "$\Rightarrow$" direction. We have $\mathbf{E}^T\mathbf{Eu} = \mathbf{u}$ and must show that a unique $\mathbf{m}$ exists with $\mathbf{u} = \mathbf{E}^T\mathbf{m}$. Equating the two terms we get $\mathbf{E}^T\mathbf{Eu} = \mathbf{E}^T\mathbf{m}$, hence $\mathbf{Eu} = \mathbf{m}$. This is the only solution. Assume another solution $\mathbf{m}' \neq \mathbf{m}$ with $\mathbf{u} = \mathbf{E}^T\mathbf{m}'$ exists. Then $\mathbf{E}^T(\mathbf{m} - \mathbf{m}') = \mathbf{0}$. Hence, $\mathbf{0} \neq \mathbf{m}' - \mathbf{m} \in \ker(\mathbf{E}^T)$. Contradiction: $\ker(\mathbf{E}^T)$ is zero-dimensional and only contains $\mathbf{0}$. This is because $\mathbf{E}^T$ is of rank q (as $\mathbf{EE}^T = \mathbf{I}$) and $\dim(\ker(\mathbf{E}^T)) + \operatorname{rank}(\mathbf{E}^T) = q$. ↯ □

Essentially, this lemma states a condition under which a unique reconstruction of a signal $\mathbf{u}$ with respect to any basis transformation matrix $\mathbf{E}$ is possible. Put differently, the signal $\mathbf{u}$ must have been "bandlimited" to be constructed from at most q functions of the function basis it is being transformed into and $N \geq q$ samples are required. This is illustrated in Figure 16. For non-orthogonal matrices $\mathbf{E}$ of rank q, the Moor-Penrose pseudo inverse $\mathbf{E}^+ = (\mathbf{E}^T\mathbf{E})^{-1}\mathbf{E}^T$ can be used instead of $\mathbf{E}^T$.

The Nyquist-Shannon sampling theorem (Shannon, 1949) is similar to the above lemma in spirit. It bridges the continuous and discrete time domains and states a condition under which continuous signals can be perfectly represented by a set of samples $\mathbf{u} = (u_1, \ldots, u_N)$. Specifically, if a function over $[0, 1]$ only contains frequencies up to φ Hertz, then $N \geq 2\varphi + 1$ uniform samples are required to perfectly represent $f(t)$.

Theorem 1 (Nyquist-Shannon Sampling Theorem)**.** *Let $f(t)$ be a bandlimited function over $\mathcal{C}[0, 1]$; that is $\langle f, f_n\rangle = 0$ for all $n > \hat{q}$, where $(f_n)_{n\in\mathbb{N}}$ is the Fourier series as defined in eq.* (5)*. $f(t)$ is completely determined by N uniform sample points over* $[0, 1]$ *if $N > \hat{q}$ (cf. Shannon, 1949 for the proof).*

6.2 Aliasing

The above lemma and theorem state under which conditions signals can be represented and transformed without information loss. But what happens if we violate these conditions? If we discretise a signal $f(t)$ with fewer samples than necessary, or transform a signal $\mathbf{u}$ into a function space where it cannot be represented using q coefficients? Besides losing information, the answer to this

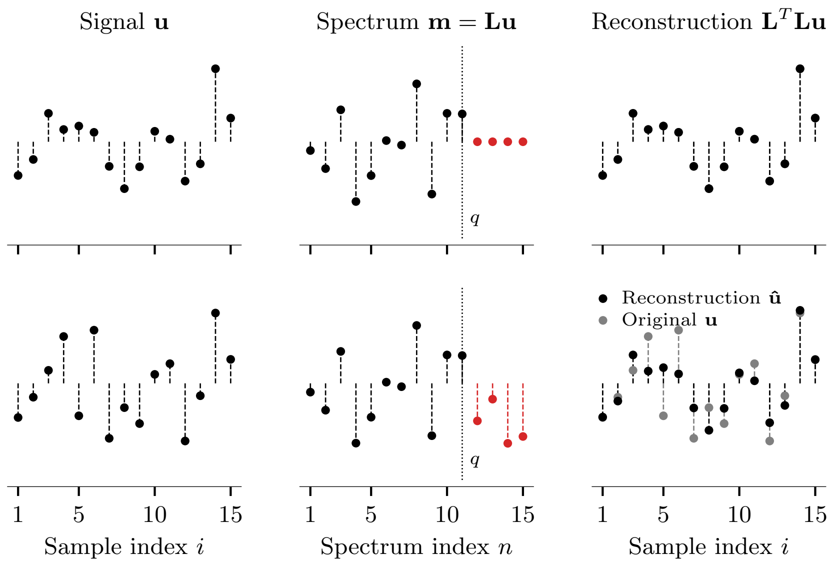

Figure 16: Illustration of Lemma 1. The discrete signal $\mathbf{u}$ has $N = 15$ samples. The signal can be losslessly transformed using $\mathbf{L} \in \mathbb{R}^{q \times N}$ with $q = 11$ if $\mathbf{u}$ can be constructed from the first q basis functions. *Top left:* Appropriately band-limited signal. *Top centre:* The Fourier coefficients m_n for $n \geq q$ (red) are all zero. *Top right:* The signal can be perfectly reconstructed. *Bottom:* The depicted signal is not properly bandlimited and cannot be losslessly reconstructed. Due to aliasing, it is reconstructed as if it was the signal in the top of the figure.

question is *aliasing*, though to different degrees of severity. Aliasing means that infinite "invalid" signals $f(t)$ or $\mathbf{u}$ are mapped onto the representation of a single "valid" signal.

Discrete case (violation of Lemma 1) In the discrete case, for a fixed $\mathbf{m} \in \mathbb{R}^q$, there are an infinite number of $\mathbf{u} \in \mathbb{R}^N$ with $\mathbf{m} = \mathbf{F}\mathbf{u}$ if $N > q$. Fortunately, assuming that the basis vectors are sorted by frequency, computing $\mathbf{F}\mathbf{u}$ merely discards higher frequency terms. We saw this in Figure 16, where the "invalid" signal is aliased onto the signal with the same first q spectral coefficients.

Sampling continuous functions (violation of Theorem 1) Violating the Nyquist-Shannon theorem causes distortions inside the frequency range representable by N samples. This is depicted in Figure 17. Not only are frequency coefficients outside the $N/2\,\text{Hz}$ range discarded when sampling a one-second signal $f(t)$ too sparsely, there are also changes ("distortions") in the magnitude of the frequency coefficients within that limit. That is, our incorrectly sampled function is aliased onto a function with (potentially) completely different fre-

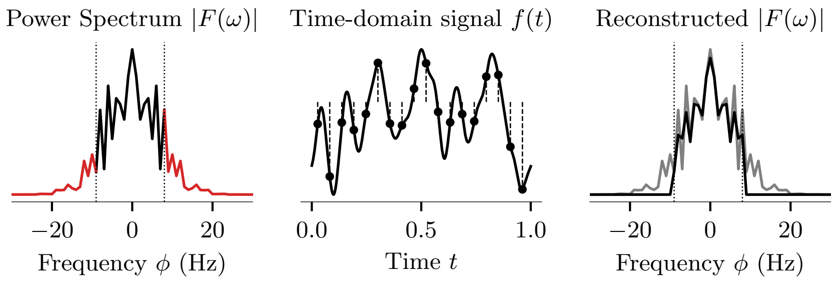

Figure 17: Violation of the Nyquist-Shannon theorem causes distortion. *Left:* Original power spectrum. Red segments should be zero for the N used. *Centre:* Original signal and samples. *Right:* Sampled power spectrum. Note the difference to the original spectrum (grey) within the valid bounds (dotted lines).

quency content. See Press et al. (2007, Section 12.1.1, pp. 605-606) for more information. This is significantly worse than what happens when we transform a discrete signal $\mathbf{u}$ into a discrete function space that does not span $\mathbf{u}$.

6.3 Anti-Aliasing Filters

To combat aliasing (i.e., "anti-aliasing"), we simply map each violating signal onto a valid signal that does not violate any of the above constraints. For example, continuous functions $f(t)$ can be low-pass filtered in the continuous domain to discard frequencies outside the $N/2\,\text{Hz}$ range. For discrete $\mathbf{u}$, we similarly discard components of $\mathbf{u}$ that cannot be represented in the target basis.

Put differently, "anti-aliasing" can be thought of as "controlled aliasing". The invalid signals are deterministically mapped ("aliased") onto a valid signal. As we saw, this is important when sampling continuous functions, as invalid signals cause distortions in the representable frequencies.

Band-limiting discrete signals Given a basis transformation matrix $\mathbf{E}$, a valid signal $\mathbf{u}'$ (in the sense of Lemma 1) is simply given as $\mathbf{u}' = \mathbf{E}^+\mathbf{E}\mathbf{u}$, where $\mathbf{E}^+ = (\mathbf{E}^T\mathbf{E})^{-1}\mathbf{E}^T$ is the Moore-Penrose pseudo inverse of $\mathbf{E}$. This filtering does not affect the generalised Fourier coefficients $\mathbf{m}$. We have

$$\mathbf{m} = \mathbf{E}\mathbf{u}' = \mathbf{E}\mathbf{E}^+\mathbf{E}\mathbf{u} = \mathbf{E}\mathbf{u}\,.$$

Filtering basis transformation matrices To spice things up a little, we can use two different discrete function bases; one for the transformation, and one for filtering. For example, let $\mathbf{E} \in \mathbb{R}^{q \times N}$ be an arbitrary discrete basis transformation matrix, and let $\mathbf{F} \in \mathbb{R}^{q' \times N}$ be the Fourier matrix. Projecting $\mathbf{u}$ onto the first q' coefficients of the Fourier basis, i.e., computing $\mathbf{u}' = \mathbf{F}^T\mathbf{F}\mathbf{u}$, and then applying the transformation $\mathbf{E}$ yields

$$\mathbf{m} = \mathbf{E}\mathbf{u}' = \mathbf{E}\mathbf{F}^T\mathbf{F}\mathbf{u} = (\mathbf{F}^T\mathbf{F}\mathbf{E}^T)^T\mathbf{u} = \mathbf{E}_\ell\mathbf{u}\,. \qquad (31)$$

This equation is implemented in `lowpass_filter_basis`*.*

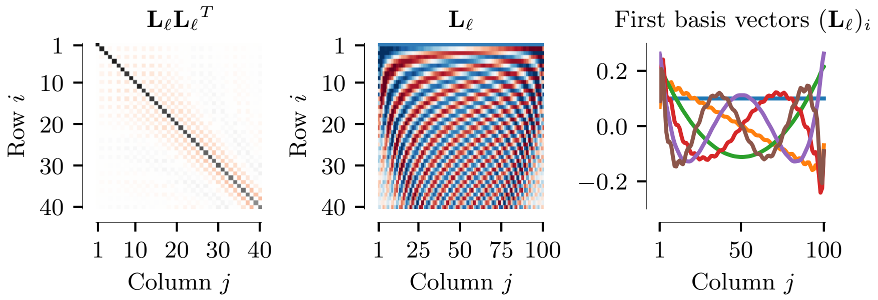

Figure 18: Examples of a low-pass filtered DLOP basis $\mathbf{L}_\ell$ for $q = q' = 40$ and $N = 100$. The filtered DLOP matrix $\mathbf{L}_\ell$ is no longer orthogonal, and instead exhibits ringing patterns similar to those found in the LDN basis.

Crucially, the term $\mathbf{E}_\ell^T = \mathbf{F}^T\mathbf{F}\mathbf{E}^T$ can be thought of as filtering the individual discrete basis functions in $\mathbf{E}$. That is, instead of filtering the incoming signal $\mathbf{u}$, we band-limit the basis functions with respect to the Fourier basis.

Figure 18 depicts a low-pass filtered versions of the DLOP function basis. The DLOP basis transformation matrix is no longer orthogonal after filtering and exhibits ringing artefacts similar to those seen in the LDN basis (cf. fig. 11). This may suggest that the LDN basis intrinsically contains an "optimal low-pass filter", but this needs further investigation. Judging from the LTI experiments in the last section, the ringing artefacts may instead be required to realize the finite impulse response (e.g., compare the impulse response of the undampened and dampened DLOP system in fig. 15 to that of the LDN system).

Connection to the Nyquist-Shannon theorem Of course, any other discrete basis transformation matrix may be used for filtering instead of $\mathbf{F}$ (using the pseudo-inverse $\mathbf{E}^+$ if $\mathbf{E}$ is not orthogonal). However, using $\mathbf{F}$ establishes a connection to the Nyquist-Shannon theorem. Any signal $\mathbf{u}'$ constructed from a Fourier-filtered basis transformation matrix $\mathbf{E}_\ell$ (i.e., $\mathbf{u}' = \mathbf{E}_\ell^T\mathbf{m}$) uniquely corresponds to a continuous function $f(t)$ as defined by Shannon (1949). This $f(t)$ is the same as the function we would obtain when increasing $N \to \infty$.

Filtering with the LDN system Using the LDN basis as a filter transformation can be beneficial when constructing LTI systems as discussed in Section 5. This ensures that the desired impulse response is a linear combination of the LDN system impulse response. The resulting LTI system is essentially a "rotated" version of the LDN system, guaranteeing a finite impulse response while still approximating the desired basis function reasonably well.

However, this approach is equivalent to applying a readout matrix $\mathbf{T}$ to the LTI system state, as discussed in Section 5.3. This is generally the better solution compared to using a different $\mathbf{A}$, $\mathbf{B}$, since the LDN system is already well conditioned.

7 Evaluating Discrete Function Bases

So far, we have discussed several discrete function bases, but we have not given any practical recommendation as for when to use which discrete function basis. This is because the notion of the "best" basis highly depends on the application.

In this section, we will first try to provide some very general recommendations as for when to use which function basis. We then perform a series of benchmarks. First, we evaluate how well delayed signals can be decoded from the representations formed by each function basis. Second, we test the discrete function bases as fixed temporal convolutions in a neural network context.

7.1 General Considerations

The *continuous* function bases discussed above can be divided into three categories: periodic, aperiodic, wavelets.

Periodic bases. A function basis $(f_n)_{n \in \mathbb{N}}$ over $[0, 1]$ is *periodic* if for every f_n the following function $\tilde{f}_n$ is infinitely continuously differentiable over $\mathbb{R}$:

$$\tilde{f}_n(x) = f_n\left(x - \lfloor x \rfloor\right) \,.$$

Intuitively, a function $f(x)$ fulfilling this requirement has no observable boundary between $x = 1$ and $x = 0$. Of all the bases we saw, only the Fourier basis (eq. 5) fulfils this requirement. Periodic bases are particularly suitable for representing periodic functions. While these bases can, of course, represent any aperiodic function $g \in L^2(0, 1)$, a *finite* generalised Fourier series of g, that is,

$$\hat{g} = \sum_{n=0}^{q-1} \langle g, f_n \rangle f_n \,,$$

will always be periodic (e.g., observe the "spikes" at the boundaries in fig. 1).

Aperiodic bases. If a basis is not periodic in the sense defined above, it is *aperiodic.* This includes the cosine (eq. 6), Legendre (eq. 9) and Haar basis (eq. 17). Analogously to the above, such bases are well suited to representing aperiodic functions.

Wavelet bases. Without defining this concept thoroughly, wavelet bases, such as the Haar basis (eq. 17), are characterized by being sparse. This is useful when approximating localized functions g, i.e., functions that are exactly zero for most x. Wavelet bases can also be thought of as intrinsically segmenting signals $\mathbf{u}$ into smaller "chunks". Hence, a small change at one point in time does not affect all generalised Fourier coefficients at once. One issue with wavelet bases is that increasing q by one (i.e., adding one basis function) does not decrease the approximation error $\hat{g}(x) - g(x)$ uniformly for all x.

Properties of various discrete function bases This list briefly characterises the *discrete* function bases we discussed in this report in terms of their computational properties and common applications where applicable.

Discrete Fourier basis. This periodic basis provides natural interpretations of signals in terms of "frequency" and "phase". Periodicity can be important when solving certain differential equations. Computing $\mathbf{Fu}$ for $\mathbf{u} \in \mathbb{R}^N$ and $\mathbf{F} \in \mathbb{R}^{N \times N}$ requires $\mathcal{O}(N \log(N))$ operations (Cooley and Tukey, 1965).

Discrete Cosine basis. This aperiodic basis tends to approximate the principal components of natural signals (such as sounds or images) and is thus extensively used in audio, video, and image compression. Furthermore, the derivative of these basis functions is zero at the boundaries, which is another common condition encountered when solving differential equations (Press et al., 2007, Section 12.4.2, p. 624). Computing $\mathbf{Cu}$ for $\mathbf{u} \in \mathbb{R}^N$ and $\mathbf{C} \in \mathbb{R}^{N \times N}$ requires $\mathcal{O}(N \log(N))$ operations (Makhoul, 1980).

DLOP basis. As discussed by Neuman and Schonbach (1974), this aperiodic basis is particularly useful when performing smooth polynomial interpolation. In contrast to the LDN basis, the DLOP basis is orthogonal. To our knowledge, there is no fast computation scheme; computing $\mathbf{Lu}$ for $\mathbf{u} \in \mathbb{R}^N$ and $\mathbf{L} \in \mathbb{R}^{q \times N}$ requires $\mathcal{O}(qN)$ operations.

LDN basis. This aperiodic basis is similar to the DLOP basis; computing $\mathbf{Hu}$ for $\mathbf{u} \in \mathbb{R}^N$ and $\mathbf{H} \in \mathbb{R}^{q \times N}$ requires $\mathcal{O}(qN)$ operations. Because of the underlying LTI system, the LDN basis has a potentially faster online, zero-delay update scheme, requiring $\mathcal{O}(q^2)$ operations per timestep. Refer to Section 4.3 for a thorough discussion.

Haar basis. As noted above, convolving a signal with this wavelet basis has a low time complexity. It is therefore extensively used in classic computer vision algorithms, including tasks such as face recognition (Viola and Jones, 2001). Computing $\mathbf{Wu}$ for $\mathbf{u} \in \mathbb{R}^N$ and $\mathbf{W} \in \mathbb{R}^{q \times N}$ requires $\mathcal{O}(N)$ operations. Online, zero-delay updates require $\mathcal{O}(q)$ operations per timestep (Kaiser, 1998). A downside is that this basis is not orthogonal for non-power-of-two N, and that q *should* be a power of two as well.

7.2 Delay Decoding Benchmark

An objective measure for the "quality" of a discrete function basis is the amount of information lost when encoding a signal $\mathbf{u}$. That is, we could simply compute

$$E = \|\mathbf{u} - \mathbf{E}^{+}\mathbf{E}\mathbf{u}\| = \|\mathbf{u} - (\mathbf{E}^T\mathbf{E})^{-1}\mathbf{E}^T\mathbf{E}\mathbf{u}\|$$

to evaluate a discrete function basis with an associated basis transformation matrix $\mathbf{E} \in \mathbb{R}^{q \times N}$. We can vary the number of discrete basis functions q and see in how far this affects the decoding error E. In fact, this is essentially what we will do, though a few details require further explanation.

Input signal selection One challenge with this approach is that the error measurement depends on the specific $\mathbf{u}$. Remember that each basis has a set of signals $\mathbf{u}$ that can be represented without error; as stated in Lemma 1, these are exactly the linear combinations of the basis vectors. If we are not careful, we could generate test signals $\mathbf{u}$ that fall into this category and thus skew the results favourably towards one particular basis.

We try to mitigate this in two ways. First, we average over a large number of randomly generated input signals. Second, the input signal itself is a low-pass filtered white noise signal. As such it contains frequencies up to the Nyquist frequency $N/2$, though with a small magnitude. We explicitly do not use hard band-limiting (eq. 31), as this would drastically bias the results toward the Fourier basis. Still, one shortcoming of our methodology is that we do not test a large corpus of different (natural) input signals. The neural network experiments in the next subsection aim to address this.

Sample decoder Instead of using the pseudo-inverse $\mathbf{E}^+$ as suggested above, we reconstruct individual input samples using a "learned" decoding vector $\mathbf{d}_\theta$. This emulates a simple machine learning context.

Let $\mathbf{u}_t = (u_{t-(N-1)}, \ldots, u_t) \in \mathbb{R}^N$ be an input signal. Given the discrete generalised Fourier coefficients $\mathbf{m}_t = \mathbf{E}\mathbf{u}_t$, we would like to determine how well a specific $u_{t-\theta}$ with $\theta \in \{0, \ldots, N-1\}$ can be reconstructed from $\mathbf{E}\mathbf{u}_t$. That is, given a fixed $\mathbf{d}_\theta$, we measure the error $E = \langle \mathbf{d}_\theta, \mathbf{E}\mathbf{u}_t \rangle - u_{t-\theta}$ over many $\mathbf{u}$ and t.

The decoder $\mathbf{d}_\theta$ is fixed for each pair of $\mathbf{E}$ and θ and is determined using regularised least squares over a large set of separate test signals.[6] In contrast to the corresponding row of $\mathbf{E}^+$, the decoder is fine-tuned to the class of signals $\mathbf{u}_t$ encountered in this experiment. Regularisation enforces a robust $\mathbf{d}_\theta$; the decoder produces a correct result even if noise is added to $\mathbf{m}$. As a side effect, the individual coefficients of $\mathbf{d}_\theta$ have a relatively small magnitude. The delay decoders $\mathbf{d}_\theta$ could be learned via stochastic gradient descent in a neural network.

Interpretation as "delay decoding" The technique described above can be seen as decoding delayed versions of the discrete input signal $\mathbf{u}_t$. If the generalised Fourier coefficients $\mathbf{m}_t$ are computed over time for each new input sample u_t, then $\mathbf{E}\mathbf{u}_t$ describes a set of FIR filters (cf. eq. 11), and $\langle d_\theta, \mathbf{E}\mathbf{u}_t \rangle$ reconstructs the sample from θ timesteps ago.

Figure 19 visualises this idea of "delay decoding" for an exemplary input signal $\mathbf{u}$ and the discrete Function bases discussed in this report (except for the naive discrete Legendre basis, which is very similar to the DLOP basis). Furthermore, the figure depicts the first six generalised Fourier coefficients $\mathbf{m}_t$. At each point in time, $\mathbf{m}_t$ encodes the last N samples of the input signal.

[6]For regularisation we pass `rcond` $= 10^{-4}$ to the Numpy `lstsq` method.

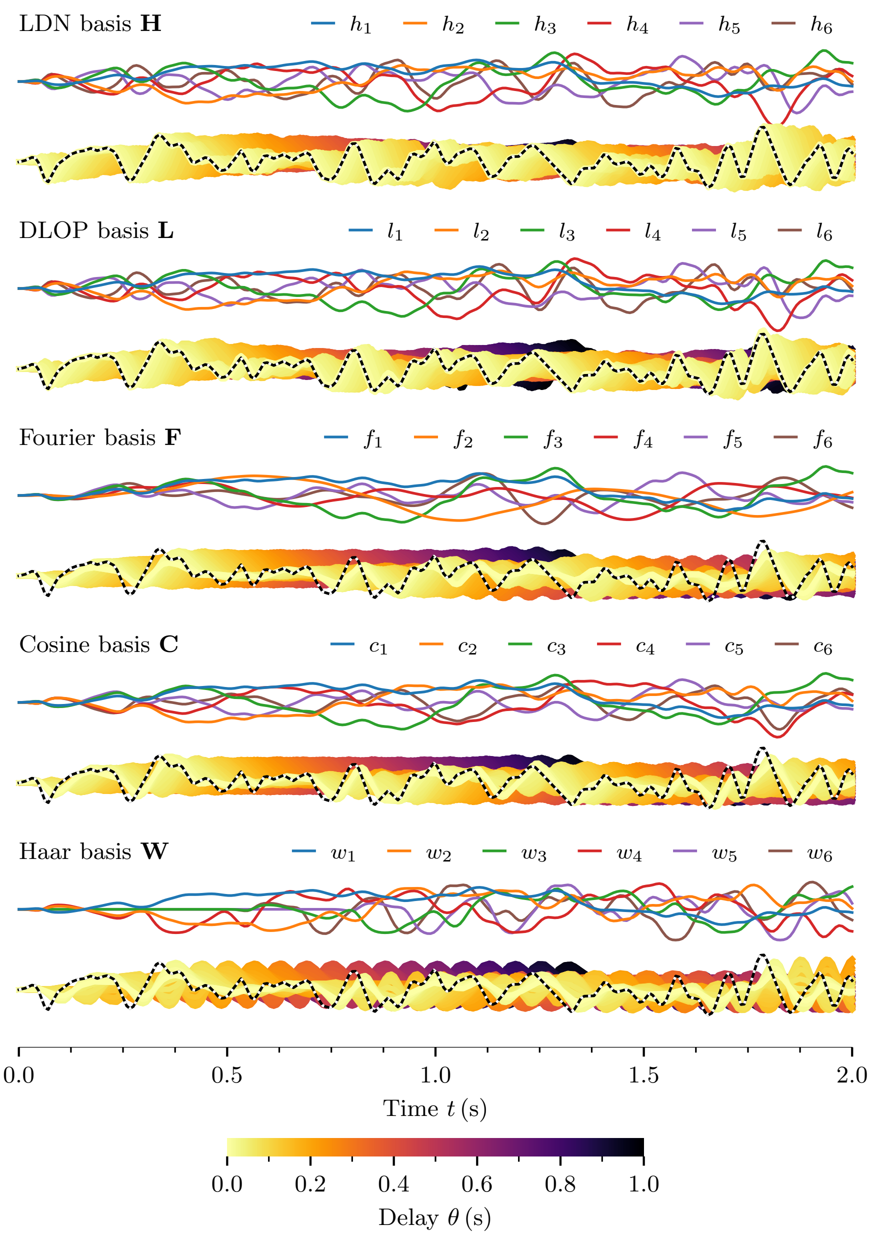

Figure 19: Approximating delays using discrete function basis representations ($q = 16$ and $N = 128$). Dotted black line is a sampled low-pass filterd white noise signal $\mathbf{u}$ with 256 samples. Light yellow to purple lines correspond to a delayed version of $\mathbf{u}$ given as $\langle \mathbf{d}_\theta, \mathbf{E}\mathbf{u}_t \rangle$, where $\mathbf{d}_\theta$ is a decoding vector, and $\mathbf{u}_t$ are the last N samples leading up to t. The first dimensions of $\mathbf{m} = \mathbf{E}\mathbf{u}_t$ are depicted in the top half of each diagram. $\theta = 1\,\mathrm{s}$ corresponds to $N = 128$ samples.

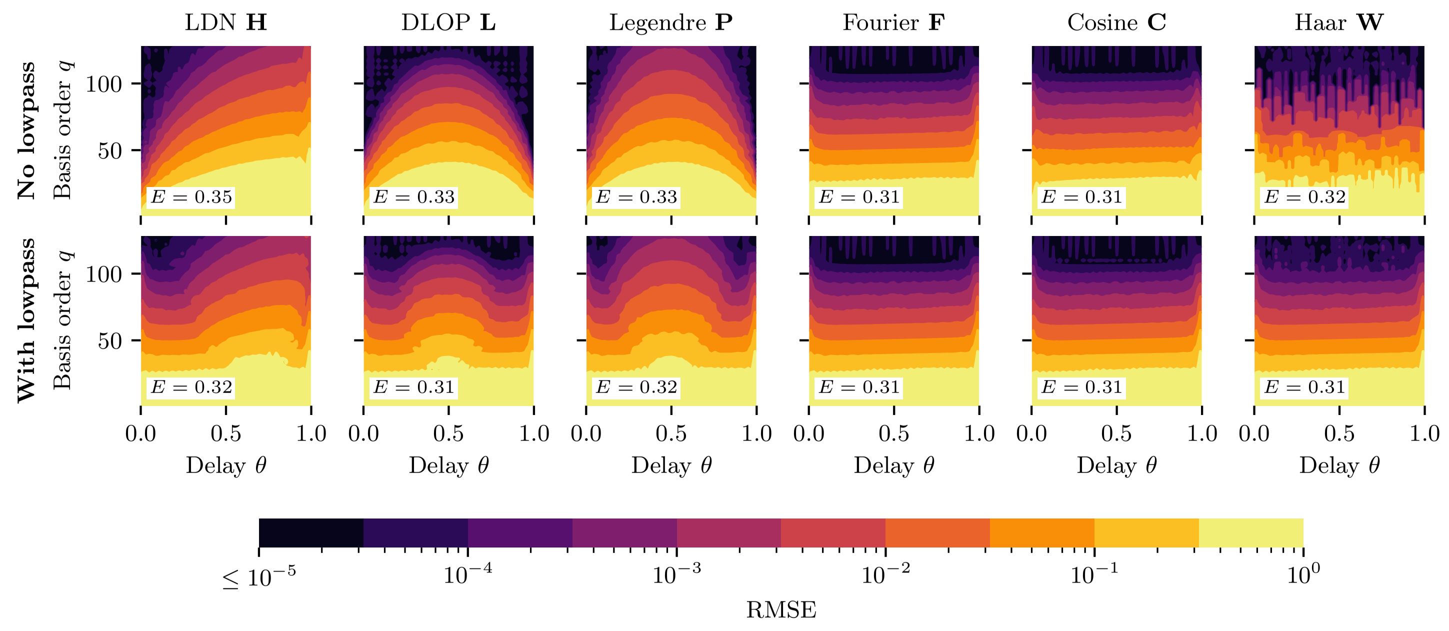

Figure 20: Comparison between individual discrete function bases when decoding a delayed signal. For all bases, $N = 128$ corresponding to 1 s. Each point is the RMSE over 1000 test signals (for 51 different qs and 51 different θs). Each test signal is generated from low-pass filtered white noise (low-pass frequency is 15 Hz). The error E in each diagram corresponds to the RMSE over all cells and trials. The LDN is optimal for decoding a delay of θ close to zero. The DLOP and Legendre basis are optimal for both θ close to zero and one. The Fourier and cosine basis are similarly good for all decoded delays.

Results Figure 20 depicts the results of the experiment described above for $N = 128$ and a signal length of 256. The figure depicts the overall average error E for the discrete bases functions discussed in this report, including band-pass filtered versions of each basis $\mathbf{E}_\ell$ (cf. eq. 31). Furthermore, contour plots break down the error for different delays θ and basis function counts q. Note that the displayed error scale is logarithmic. Even though some plots look drastically different, the absolute differences in error are not.

Discrete LDN basis. The LDN basis has the highest overall error with $E = 0.35$. Even though it was derived from an LTI system that optimally implements a delay of length $\theta = 1$, information is gradually lost as it propagates through the system. This is also visible in Figure 19, as well as the basis transformation matrix $\mathbf{H}$ in Figure 11, where the left half of the matrix is "fading out" for larger q. The lowest error is for a delay of length zero, which is of limited practical use.
Discrete DLOP basis. The DLOP basis has a slightly smaller overall error with $E = 0.33$. In contrast to the LDN basis, it can both reconstruct a delay of length $\theta = 0$ and $\theta = 1$ very well. The highest error is reached for $\theta = 0.5$. This can be explained by looking at the Legendre polynomials in Figure 2c. Half of the polynomials are zero at $x = 0.5$, and the other half has a relatively small magnitude for larger q, reducing the amount of information that can be decoded.

Legendre basis. The discretised Legendre basis reaches the same overall error as the DLOP basis with $E = 0.33$, and the overall shape of the contour plots is the same. The error is a little higher for larger q compared to the DLOP basis, which is likely a result of the DLOP basis being orthogonal.

Fourier and cosine basis. These two bases have the smallest overall error with $E = 0.31$. In both cases, the decoding error is uniform over all delays θ.

Haar basis. Surprisingly, the Haar basis has an error of only $E = 0.32$, which is slightly smaller than the overall error for the DLOP basis. However, as visible in Figure 19, only a fixed number of delays can be decoded well from the Haar basis; for $q = 16$, only 16 different delay values can be decoded. Furthermore, in contrast to all other bases, increasing q does not uniformly decrease the decoding error. This is because an individual basis function only provides local support.

Filtering. Filtering reduces the error for all bases to $E = 0.32$. This is not surprising, since, after filtering, each basis function is expressed in terms of the Fourier basis (cf. Section 6.3). The decoding vector $\mathbf{d}_\theta$ can partially "undo" the corresponding linear combination and thus treat the generalized Fourier coefficients as if they were with respect to the Fourier basis.

The most striking result is the effect filtering has on the Haar basis. After band-limiting the basis, or equivalently—see Section 6.3—the input signal, the error-over-delay plot for the Haar basis looks very similar to those of the Fourier and cosine basis. This suggests that the Haar basis is an excellent choice if the input signal $\mathbf{u}$ is band-limited in the Fourier domain.

7.3 Neural Network Experiments

It is unclear in how far the results of the above experiments generalise to a neural network context. To gain a better understanding of how basis transformations behave in neural networks, we perform a series of experiments on two toy datasets: permuted sequential MNIST (psMNIST), and the Mackey-Glass system.

Our experiments are mainly meant to answer to questions. First, we would like to know in how far the performance of the network depends on the selected basis, or, "temporal convolution". Second, we compare fixed temporal convolutions to networks with the same architecture, where the temporal convolution is initialized in the same way, but trained along with the other parameters. This latter approach is equivalent to "Temporal Convolutional Networks", an older idea that has lately been popularized by Bai, Kolter, and Koltun (2018).

There is an argument to be made that fixed convolutions can—under some circumstances—be a better choice compared to learned convolutions. Reduced run-time and memory requirements aside, fixed convolutions reduce the number of parameters in the network and may thus lead to faster convergence. At the same time, a fixed orthogonal basis spans a convenient space from which arbitrary functions can be linearly decoded.

This is not to say that convolutional neural networks cannot learn good function bases—they surely can. An example of this—albeit in image processing—are Gabor-like filters found in the first convolution layer in deep convolutional networks (cf. Krizhevsky, Sutskever, and Hinton, 2017). However, learning such clean basis functions from scratch requires large datasets and many epochs of training. Furthermore, judging from the examples provided in the Krizhevsky paper, learned convolutions tend to be somewhat redundant. That is, they do not achieve the same degree of orthogonality as a hand-picked basis.

7.3.1 psMNIST

This task has originally been proposed by Le, Jaitly, and Hinton (2015). The MNIST dataset (Lecun et al., 1998) contains images of hand-written digits between zero and nine. Each image consists of 28×28 greyscale pixels.

The idea of the psMNIST task is to treat each image as a sequence of $N = 784$ individual samples over time. Once the final sample has been processed, the system must correctly classify the digit. To eliminate spatial correlations in the sequence, a random but fixed permutation π is applied to the samples. This results in a signal of the form $\mathbf{u} = (u_{\pi(1)}, \ldots, u_{\pi(784)})$.

Comparability of psMNIST implementations The psMNIST task as described above is a little under-defined. When comparing different neural architectures, two additional constraints should be met. First, all architectures must use the same amount of *memory* (Voelker, Kajić, and Eliasmith, 2019).

Second, the resulting architecture should support *serial execution*, or, borrowing terminology from Chandar et al. (2019), have a good "forgetting ability".

```
N = 28 * 28; M = 346; H = mk_basis(q, N)
model = tf.keras.models.Sequential([
  tf.keras.layers.Reshape((N, 1)),              # (N, 1)
  TemporalBasisTrafo(H, units=1),               # (1, q)
  tf.keras.layers.Dropout(0.5),                 # (1, q)
  tf.keras.layers.Dense(M, activation="relu"), # (1, M)
  tf.keras.layers.Dense(10, use_bias=False),    # (1, 10)
  tf.keras.layers.Reshape((10,))                # (10)
])
```

Algorithm 1: Code used for the psMNIST experiment. Comments (`Nt`, `Nd`) indicate the output dimensions of each layer. `Nt` denotes the number of temporal samples; `Nd` is the number of non-temporal dimensions. `q` is between 1 and N.

In other words, the network must still produce correct classifications over time, even if multiple input signals are concatenated.

FIR filters intrinsically fulfil the serial execution constraint, but violate the memory constraint. Each filter requires access to all N samples in memory. This essentially reduces the task to a non-sequential MNIST task—despite a set of q FIR filters with $q < N$ forming an information bottleneck.

Hence, unless an LTI system with q state variables is used to implement the system, our results are not directly comparable to other psMNIST benchmarks. As we saw, reasonably good LTI systems can be constructed for most of the bases discussed here (cf. Section 6.3). However, only the LDN basis has a near-finite impulse response and supports serial execution without external state resets.

In the light of these constraints—and apart from the LDN basis results—our experiments are merely meant to compare different sets of basis functions, and not to contend with other approaches implementing psMNIST.

Methods Our model is specified in Algorithm 1. A single temporal basis transformation layer[7] applies a set of q FIR filters to N input samples; 346 ReLU neurons non-linearly processes the transformed input, followed by a linear read-out layer. The psMNIST training set is randomly split into 50 000 training and 10 000 validation samples. An Adam optimizer with default parameters is used over 100 epochs with a batch size of 100. The reported test errors are computed for the parameter set with the smallest validation error. For $q = 468$ the number of trainable parameters is $\sim$166k. Per default, the set of FIR filters are fixed and initialised with one of the discrete function bases. Another $\sim$350k parameters are added if the FIR filters are trained as well.

Results Results are depicted in Figure 21, Figure 22 and Table 2. For $q > 20$ there is no appreciable difference between the function bases, with the LDN

[7]The code for the `TemporalBasisTrafo` class can be found at `https://github.com/astoeckel/temporal_basis_transformation_network`.

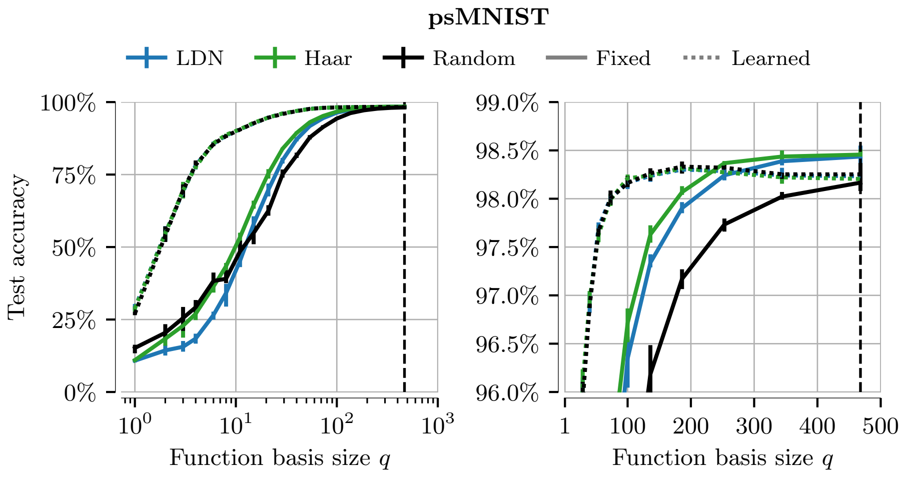

Figure 21: Mean psMNIST classification accuracy for different discrete temporal function bases. Data for the Haar basis are representative for the other fixed function bases (omitted for clarity). Dotted lines are errors obtained when learning the temporal convolution. Each data point is for $n = 11$ trials; error bars correspond to the first and third quartile. Dashed vertical line is at $q = 468$; detailed data for this q are given in Table 2 and Figure 22.

basis having slightly lower accuracies on average. When learning the temporal convolution, differences between the individual initializations disappear. Learning yields drastically better results for $q < 200$ with a peak at about $q = 190$ with an accuracy of about 98.4%. The accuracy is monotonically increasing with q when using fixed convolutions, reaching about 98.5% for $q = 468$. A random initialization reaches error rates of about 98.1%.

7.3.2 Mackey-Glass

The point of this task is to predict the time-course of the chaotic Mackey-Glass dynamical system for a certain number of samples. The Mackey-Glass system is

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = \frac{ax(t-\tau)}{1 + x(t-\tau)^{10}} - bx(t)\,,$$

where $a = 0.2$, $b = 1.2$, and $x(t) = 1.2 + \eta$ with $\eta \sim \mathcal{N}(\mu = 0, \sigma = 1)$ for $t < 0$. This system is chaotic for $\tau > 17$ and has been extensively used as a benchmark in time-series prediction (cf. Mendel, 2017, Section 4.3.1).

Dataset For our experiments, we choose $\tau = 30$. As a training dataset we generate 400 Mackey-Glass trajectories of length 10 000. The system as been discretised with a timestep of $\Delta t = 1$ using a fourth order Runge-Kutta integrator.

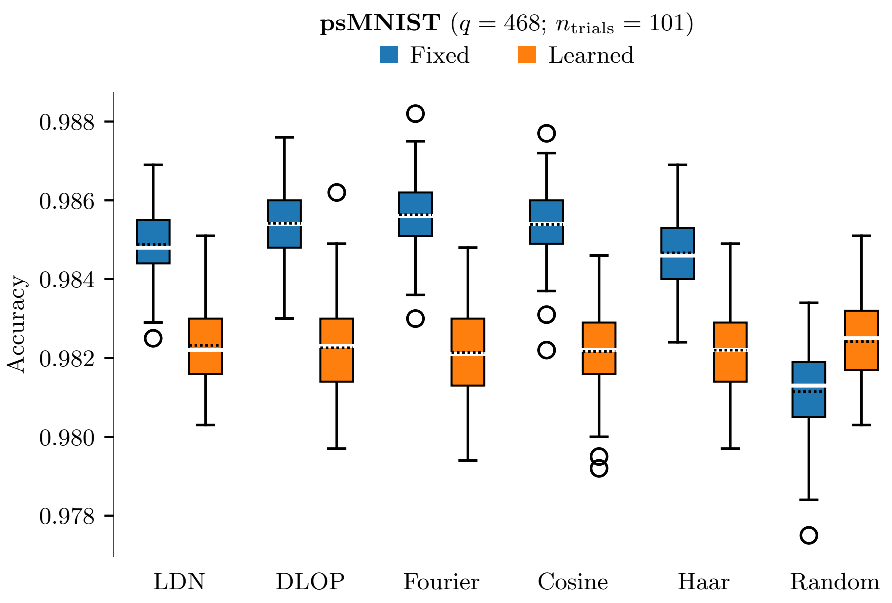

Figure 22: Classification accuracies for the psMNIST dataset using different discrete function bases as temporal convolutions with $q = 468$. Depicted are standard box-plots over $n = 101$ trials for each basis. Box corresponds to the first and third quartile; whiskers are the minimum/maximum after outlier rejection; outliers are depicted as circles. Thick white line is the median, dashed black line is the mean. Numerical values are given in Table 2.

Table 2: Test accuracies for the psMNIST experiment for $q = 468$. Data over $n = 101$ trials and 100 epochs. Q1 and Q3 are the 25- and 75-percentile, respectively. The best three results are highlighted in each column (darker colours are better).

Initial	🟦 **Fixed convolution**				🟧 **Learned convolution**			
basis	Mean	Median	Q1	Q3	Mean	Median	Q1	Q3
LDN	98.49	98.48	98.44	98.55	98.23	98.22	98.16	98.30
DLOP	98.54	98.54	98.48	98.60	98.23	98.23	98.14	98.30
Fourier	98.56	98.56	98.51	98.62	98.21	98.21	98.13	98.30
Cosine	98.54	98.54	98.49	98.60	98.22	98.22	98.16	98.29
Haar	98.47	98.46	98.40	98.53	98.22	98.22	98.14	98.29
Random	98.11	98.13	98.05	98.19	98.24	98.25	98.17	98.32

```
N_units0, N_units1, N_units2, N_units3 = 1, 10, 10, 10
N_wnd = N_wnd0 + N_wnd1 + N_wnd2 + N_wnd3 - 3
q0, q1, q2, q3 = 16, 8, 8, 4
H0, H1 = mk_basis(q0, N_wnd0), mk_basis(q1, N_wnd1)
H2, H3 = mk_basis(q2, N_wnd2), mk_basis(q3, N_wnd3)
model = tf.keras.models.Sequential([
  tf.keras.layers.Reshape((N_wnd, 1)),
  # (N_wnd0 + N_wnd1 + N_wnd2 + N_wnd3 - 3, 1)
  TemporalBasisTrafo(H0, n_units=N_units0, pad=False),
  # (N_wnd1 + N_wnd2 + N_wnd3 - 2, q0 * N_units0)
  tf.keras.layers.Dense(N_units1, activation="relu"),
  # (N_wnd1 + N_wnd2 + N_wnd3 - 2, N_units1)
  TemporalBasisTrafo(H1, n_units=N_units1, pad=False),
  # (N_wnd2 + N_wnd3 - 1, q1 * N_units1)
  tf.keras.layers.Dense(N_units2, activation="relu"),
  # (N_wnd2 + N_wnd3 - 1, N_units2)
  TemporalBasisTrafo(H2, n_units=N_units2, pad=False),
  # (N_wnd3, q2 * N_units2)
  tf.keras.layers.Dense(N_units3, activation="relu"),
  # (N_wnd3, N_units3)
  TemporalBasisTrafo(H3, n_units=N_units3, pad=False),
  # (1, q3 * N_units3)
  tf.keras.layers.Dense(N_pred, use_bias=False),
  # (1, N_pred)
  tf.keras.layers.Reshape((N_pred,))
  # (N_pred)
])
```

Algorithm 2: Code used for the Mackey-Glass experiment. Comments (`Nt`, `Nd`) indicate the output dimensions of the layer in the preceding line. `Nt` is the number of temporal samples; `Nd` is the number of non-temporal dimensions. Total number of trainable parameters (with fixed convolutions) is 2390.

All trajectories are different due to the stochasticity of the initial $x(t)$ for $t < 0$. We randomly extract 100 input sequences of length 33 followed by a target sequence of length 15 from each trajectory. This results in 40 000 training samples. The task is to predict the 15 target samples from the preceding 33.

We similarly generate 10 000 validation and 10 000 test samples. Training, validation, and test samples are all taken from separately generated trajectories.

Methods We use the neural network architecture described in Algorithm 2. This architecture is inspired by the architecture described by Voelker, Kajić, and Eliasmith (2019). To summarise, there are four cascading temporal basis transformation layers. The number of FIR filters q in each layer is equal to the number of input samples N, so no stage loses information. The first layer uses $q = N = 16$, the second and third $q = N = 8$, and the final layer $q = N = 4$, resulting in a total input window length of $16 + 8 + 8 + 4 - 3 = 33$.

The second to fourth layers consist of ten independent units using exactly

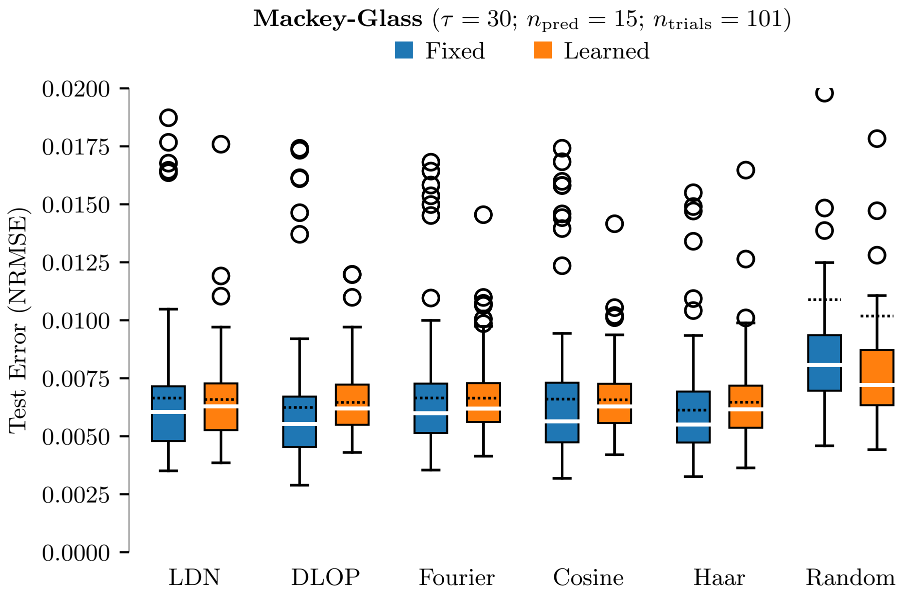

Figure 23: Prediction errors for the Mackey-Glass dataset using different discrete function bases as fixed temporal convolutions. See Figure 22 for more detail on the box-plots. Numerical values are given in Table 3.

the same set of convolutions. Layers are densely connected using ReLUs. Again, we compare fixed convolutions—including random initialization—to a version of the network where the convolutions are initialized in the same way, but then further refined during training. The learned convolutions are normalized, such that each learned basis function has norm one. The total number of trainable parameters is 2760; another 400 are added if the convolutions are trained.

We train the network for 100 epochs using a standard Adam optimizer. The final test error is computed for the parameters in the epoch with the smallest validation error. We measure the normalised RMSE, i.e., the RMSE divided by the RMS of the Mackey-Glass trajectories ($\approx$0.94).

Results Results are depicted in Figure 23 and Table 3. All fixed convolutions exhibit approximately the same performance. The random initialization has the highest error, and the Haar and DLOP filters result in the lowest error. Learning the convolution results in a slightly higher error overall, but closes the performance gap between the individual basis transformations.

Table 3: Test errors for the Mackey-Glass experiment. Data over $n = 101$ trials and 100 epochs. Q1 and Q3 are the 25- and 75-percentile, respectively. The best three results are highlighted in each column (darker colours are better).

Initial	🟦 **Fixed convolution**				🟧 **Learned convolution**			
basis	Mean	Median	Q1	Q3	Mean	Median	Q1	Q3
LDN	0.0067	0.0061	0.0048	0.0072	0.0066	0.0063	0.0053	0.0073
DLOP	0.0063	0.0055	0.0045	0.0067	0.0065	0.0062	0.0055	0.0072
Fourier	0.0067	0.0060	0.0051	0.0073	0.0067	0.0062	0.0056	0.0073
Cosine	0.0066	0.0057	0.0047	0.0073	0.0066	0.0063	0.0056	0.0073
Haar	0.0061	0.0055	0.0047	0.0069	0.0065	0.0062	0.0054	0.0072
Random	0.0109	0.0081	0.0070	0.0094	0.0102	0.0072	0.0063	0.0087

8 Discussion

We provided a whirlwind overview of discrete function bases with a particular focus on the LDN and DLOP bases. When used as FIR filters, these bases can compress temporal data into a stream of generalized Fourier coefficients.

Since the LDN basis was constructed from an LTI system with an almost finite impulse response, this convolution operation can be approximated "online" by recurrently advancing the discretised LTI system. This requires only $\mathcal{O}(q)$ *state* memory ($\mathcal{O}(q^2)$ in total), but, in general, $\mathcal{O}(q^2)$ operations per sample. This can be faster than online, zero-delay convolution using a set of FIR filters, which require at least $\mathcal{O}(q \log(N))$ operations per sample, but $\mathcal{O}(N \log(N))$ memory.

The LDN is similar in principle to the sliding transformations for the Haar, cosine, and Fourier basis. These transformations require $\mathcal{O}(q)$ update operations at the cost of $\mathcal{O}(N)$ memory. In contrast, memory and run-time requirements of the LDN can be reduced to $\mathcal{O}(q)$ when using the Euler update, although there are some potential caveats when using this approach (cf. Section 4.3).

While the DLOP basis performs better than the LDN basis, we are not aware of a fast update scheme. This basis should only be used if this is not a concern.

It is surprising to see that fixed, (almost) orthogonal temporal convolutions can yield better results than learned convolutions in neural networks. As seen in the psMNIST experiment, increasing the number of basis functions q can decrease the performance of the network when using learned convolutions. Doing the same for fixed convolutions seems to monotonically increase the performance of the system. This is somewhat intriguing, since fixed convolutions result in fewer trainable parameters, and, as discussed above, enable the use of efficient update schemes. In general, learned convolutions must resort to the $\mathcal{O}(q \log(N))$ Gardner (1995) algorithm. Future work is required to validate these results and to understand under which circumstances fixed convolutions can make sense.

Acknowledgements

This technical report was inspired by work done by Narsimha R. Chilkuri on feed-forward networks exploiting the LDN basis transformation. The original approach outlined in Section 4.2 is his idea, including eqs. (27) and (28). He also suggested to think about the LDN in terms of "optimally forgetting" signals outside the time-window, which is not necessary when using FIR filters. We furthermore thank Aaron R. Voelker and Chris Eliasmith for their feedback on this report.

References

Bai, Shaojie, J. Zico Kolter, and Vladlen Koltun (2018). *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling*. arXiv: 1803.01271 [cs.LG].

Baker Jr., George A. (2012). "Padé Approximant". In: *Scholarpedia* 7.6, p. 9756. DOI: 10.4249/scholarpedia.9756.

Chandar, Sarath et al. (July 2019). "Towards Non-Saturating Recurrent Units for Modelling Long-Term Dependencies". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01, pp. 3280–3287. DOI: 10.1609/aaai.v33i01.33013280. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4200.

Comon, Pierre (1994). "Independent Component Analysis, A New Concept?" In: *Signal Processing* 36.3, pp. 287–314. ISSN: 0165-1684. DOI: 10.1016/0165-1684(94)90029-9. URL: http://www.sciencedirect.com/science/article/pii/0165168494900299.

Cooley, James W. and John W. Tukey (1965). "An Algorithm for the Machine Calculation of Complex Fourier Series". In: *Mathematics of Computation* 19, pp. 297–301. DOI: 10.1090/S0025-5718-1965-0178586-1. URL: https://www.ams.org/journals/mcom/1965-19-090/S0025-5718-1965-0178586-1/.

DeRusso, Paul M. et al. (1998). *State Variables for Engineers*. Second Edition. New York, NY: John Wiley & Sons. 575 pp.

Gardner, William G. (1995). "Efficient Convolution without Input-Output Delay". In: *Journal of the Audio Engineering Society* 43.3, pp. 127–136. URL: http://www.aes.org/e-lib/browse.cfm?elib=7957.

Gu, Albert et al. (2020). *HiPPO: Recurrent Memory with Optimal Polynomial Projections*. arXiv: 2008.07669 [cs.LG].

Hefferon, Jim (2020). *Linear Algebra*. 525 pp. ISBN: 978-1-944325-03-9. URL: https://hefferon.net/linearalgebra/.

Jacobsen, E. and R. Lyons (Mar. 2003). "The Sliding DFT". In: *IEEE Signal Processing Magazine* 20.2, pp. 74–80. ISSN: 1558-0792. DOI: 10.1109/MSP.2003.1184347.

Kaiser, G. (Apr. 1998). "The Fast Haar Transform". In: *IEEE Potentials* 17.2, pp. 34–37. ISSN: 1558-1772. DOI: 10.1109/45.666645.

Kober, V. (2004). “Fast Algorithms for the Computation of Sliding Discrete Sinusoidal Transforms”. In: *IEEE Transactions on Signal Processing* 52.6, pp. 1704–1710. DOI: 10.1109/TSP.2004.827184.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (May 2017). “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Communications of the ACM* 60.6, pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386. URL: https://doi.org/10.1145/3065386.

Le, Quoc V., Navdeep Jaitly, and Geoffrey E. Hinton (2015). *A Simple Way to Initialize Recurrent Networks of Rectified Linear Units*. arXiv: 1504.00941 [cs.NE].

Lecun, Y. et al. (1998). “Gradient-Based Learning Applied to Document Recognition”. In: *Proceedings of the IEEE* 86.11, pp. 2278–2324. DOI: 10.1109/5.726791.

Makhoul, J. (1980). “A Fast Cosine Transform in One and Two Dimensions”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28.1, pp. 27–34. DOI: 10.1109/TASSP.1980.1163351.

Mendel, Jerry M. (2017). *Uncertain Rule-Based Fuzzy Systems: Introduction and New Directions*. 2nd Edition. Cham, Switzerland: Springer International Publishing AG. 701 pp. ISBN: 978-3-319-51369-0. DOI: 10.1007/978-3-319-51370-6.

Neuman, C. P. and D. I. Schonbach (1974). “Discrete (Legendre) Orthogonal Polynomials—a Survey”. In: *International Journal for Numerical Methods in Engineering* 8.4, pp. 743–770. DOI: 10.1002/nme.1620080406.

Press, William H. et al. (2007). *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. 3rd ed. New York, NY, USA: Cambridge University Press. ISBN: 0-521-88068-8 978-0-521-88068-8.

Shannon, C. E. (1949). “Communication in the Presence of Noise”. In: *Proceedings of the IRE* 37, pp. 10–21.

Stöckel, Andreas (2021). *Constructing Dampened LTI Systems Generating Polynomial Bases*. arXiv: 2103.00051. URL: https://arxiv.org/abs/2103.00051.

Verhaegen, Michel and Vincent Verdult (2007). *Filtering and System Identification: A Least Squares Approach*. Cambridge, United Kingdom: Cambridge University Press. 405 pp. ISBN: 978-0-521-87512-7. URL: https://www.cambridge.org/9780521875127.

Viola, P. and M. Jones (2001). “Rapid Object Detection Using a Boosted Cascade of Simple Features”. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 1, pp. I–I. DOI: 10.1109/CVPR.2001.990517.

Voelker, Aaron R. (2019). “Dynamical Systems in Spiking Neuromorphic Hardware”. PhD thesis. Waterloo, ON: University of Waterloo. URL: http://hdl.handle.net/10012/14625.

Voelker, Aaron R. and Chris Eliasmith (Mar. 2018). “Improving Spiking Dynamical Networks: Accurate Delays, Higher-Order Synapses, and Time Cells”. In: *Neural Computation* 30.3, pp. 569–609. DOI: 10.1162/neco_a_01046.

Voelker, Aaron R., Ivana Kajić, and Chris Eliasmith (2019). "Legendre Memory Units: Continuous-Time Representation in Recurrent Neural Networks". In: *Advances in Neural Information Processing Systems.*

Young, Nicholas (1988). *An Introduction to Hilbert Space.* Cambridge Mathematical Textbooks. Cambridge, United Kingdom: Cambridge University Press. 239 pp. ISBN: 978-0-521-33717-5. DOI: 10.1017/CBO9781139172011.

Revisions

(1) January 11, 2021. First internal version of this document.

(2) January 26, 2021. Added the ZOH vs. Euler error diagram Figure 9. Discussed sliding transforms in Section 4.3; expanded the discussion of the asymptotic complexity of the ZOH vs. Euler LDN update. Added Figure 12 and Table 1. Added discussion of dampening by "information erasure". Added Section 7.3. Using a reordered Haar transformation matrix; commented on the effects of filtering this better Haar matrix. Rewrote the abstract and discussion to take these changes into account.

(3) March 9, 2021. Publication to arXiv. Added reference to the "Constructing Dampened LTI Systems Generating Polynomial Bases" technical report.