Structured State Space Models

Uladzislau Yorsh

Charles University in Prague, 2024

CHARLES
UNIVERSITY

Table of Contents

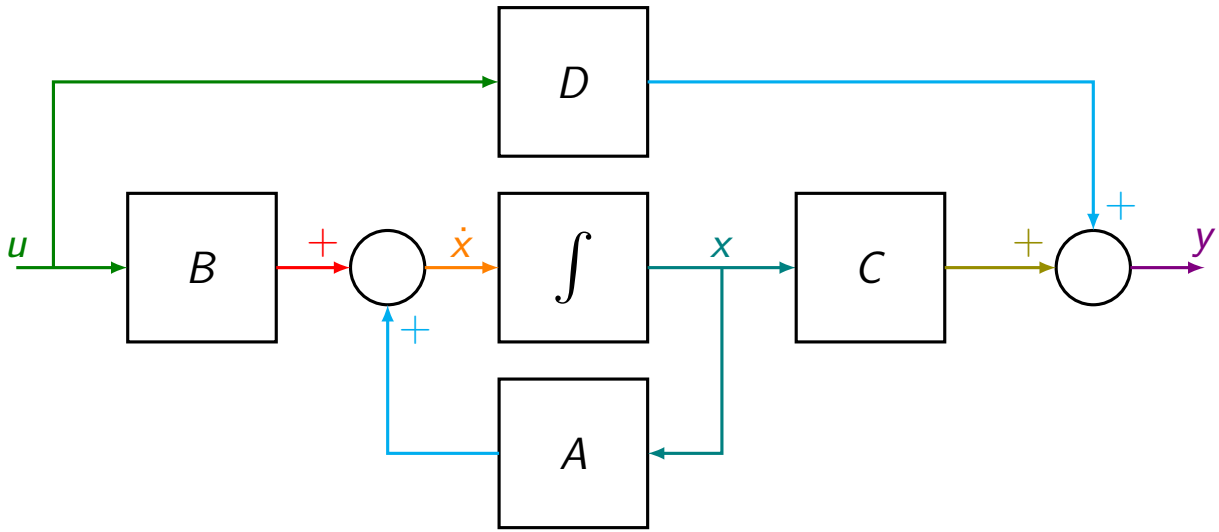
What are Linear State Space Models?

- Linear State Space Models (SSMs) are classical models in engineering, signal processing, finance, neuroscience, etc.
- Operate with finite-dimensional **state** vectors $x(t)$ and **input** (or **control**) vectors $u(t)$ (the discrete version is defined analogously):

$$\dot{x} = A(t)x(t) + B(t)u(t)$$
$$y = C(t)x(t) + D(t)u(t)$$

where $\dot{x}, x(t) \in \mathbb{C}^d$, $u(t) \in \mathbb{C}^n$, $y \in \mathbb{C}^m$ and A, B, C, D are conformable.

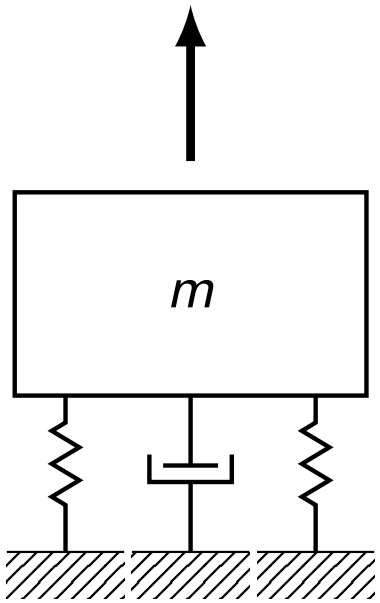
What are Linear State Space Models?



By integrating $\dot{x}$ over time we obtain a trajectory of x. In practice most of SSMs describe Linear Time Invariant (LTI) systems with fixed matrices.

SSMs in Engineering: from linear ODE

- SSMs are an essence of the modern control theory.
- Linear ODE systems or rational transfer functions allow to immediately write a corresponding SSM.
- First example: damped mass-spring system.



$$my'' + cy' + ky = r(t)$$

define:

$$x = \begin{bmatrix} y & y' \end{bmatrix}^T \qquad u = r$$

$$A = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad D = \mathbf{0}$$

then

$$\dot{x} = Ax(t) + Bu(t), y(t) = Cx$$

SSMs in Engineering: from rational transfer function

- Sometimes a transfer function can be given as a system description.
- Rational transfer functions correspond to linear ODEs in terms of input and output derivatives.

$$H(s) = \frac{N(s)}{D(s)} = \frac{b_{n-1}s^{n-1} + b_{n-2}s^{n-2} + \cdots + b_0}{s^n + a_{n-1}s^{n-1} + \cdots + a_0}$$

$$A = \begin{bmatrix} -a_{n-1} & -a_{n-2} & \dots & -a_0 \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} b_{n-1} & b_{n-2} & \dots & b_0 \end{bmatrix} \qquad D = \mathbf{0}$$

SSMs in Neuroscience

- Instead of real-valued signals neuroscientists deal with spike trains.
- Decoding them is implemented via filtering, thus neuroscience extensively uses signal processing toolkit.
- For example, lots of biological systems are integrators of some kind (e.g. eye positioning).
- SSMs are a convenient tool to describe these processes.

Let's train a SSM!

- The simple definition suggests to simply plug randomly initialized A, B, C, D into a gradient learner.
- Indeed we can do so, but the results will not be miraculous...
- Flattened permuted MNIST results:

Method	Val. acc. (%)
-LegS	**98.34**
-LagT	98.15
-LegT $\theta = 200$	98.0
-LegT $\theta = 20$	91.75
-Rand	69.93

Interlude 1: Laplace Transform

Laplace transform is the most essential integral transform in engineering.

- Defined as $\mathcal{L}[f] = \int_0^\infty f(t)e^{-st}dt = F(s)$, invertible for many functions.
- Linear, $\mathcal{L}[f * g] = \mathcal{L}[f]\mathcal{L}[g]$, $\mathcal{L}[f'] = sF(s) - f(0)$.
- The latter shows that **linear ODEs** in time domain correspond to **polynomials** in Laplace domain.
- The ratio of Laplace transforms of output and input is called **transfer function**.

Interlude 1: Laplace Transform

Theorem: a transfer function can be converted into a SSM iff it can be written as a *proper* (numerator degree $\leq$ denominator degree) rational function.

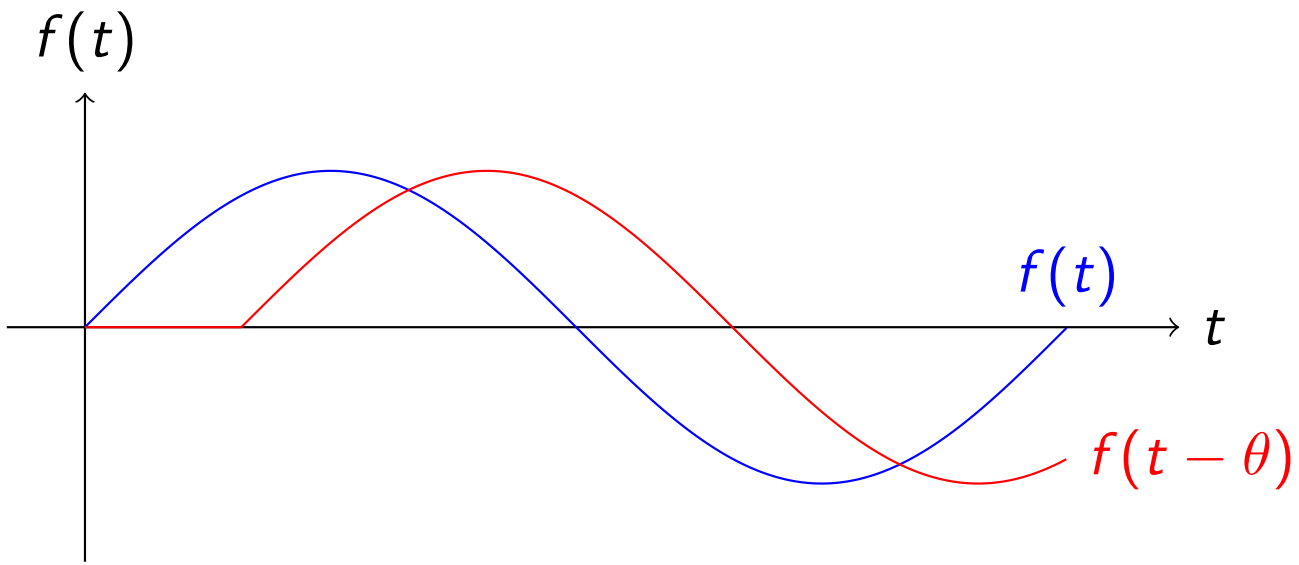
- Intuitively, to compute higher order derivatives we need more information.
- Higher order of the numerator of the improper function thus corresponds to a larger necessary amount of information than provided by input.
- It makes the system non-causal, i.e. the output at time t depends on the input at time $\tau > t$.

Key takeaways so far

- Many dynamical systems can be described by LTIs $\dot{x} = Ax(t) + Bu(t), y(t) = Cx(t) + Du(t)$.
- SSM equations can be obtained from transfer functions, but they need to be rational and proper.
- Transfer function is a ratio of an output and input Laplace transforms.
- Laplace transform is intimately connected with ODEs describing the system.
- Good deep learning models cannot be naïvely built with randomly initialized SSMs :(

Legendre Memory Units

- Consider a task of implementing a **continuous delay** $h(t) = f(t - \theta)$ by θ units.
- The task is a prototype for studying dynamical properties of neural computers in neuroscience.
- Has no solution in terms of a finite polynomial basis.



Legendre Memory Units

- The corresponding transfer function is $\frac{\mathcal{L}[f(t-\theta)]}{\mathcal{L}[f(t)]} = e^{-\theta s}$.
- It is not rational, so it needs an approximation.
- The authors leverage Padé approximants for $e^{-\theta s}$ and perform an extensive trickery to make it numerically stable.
- Even more trickery they use to derive the decoding for delays $\theta' < \theta$ from the model for θ.

We defer to the outstanding Aaron Voelker's PhD thesis for a detailed derivation in References.

Legendre Memory Units

- They arrive to the following SSM:

$$\theta \dot{m}(t) = Am(t) + Bu(t)$$

where

$$A \in \mathbb{R}^{n \times n}, A_{ij} = (2i+1) \begin{cases} -1 & i < j \\ (-1)^{i-j+1} & i \geq j \end{cases}$$

$$B \in \mathbb{R}^{n \times 1}, B_i = (2i+1)(-1)^i \qquad i, j \in 0, \ldots, n-1$$

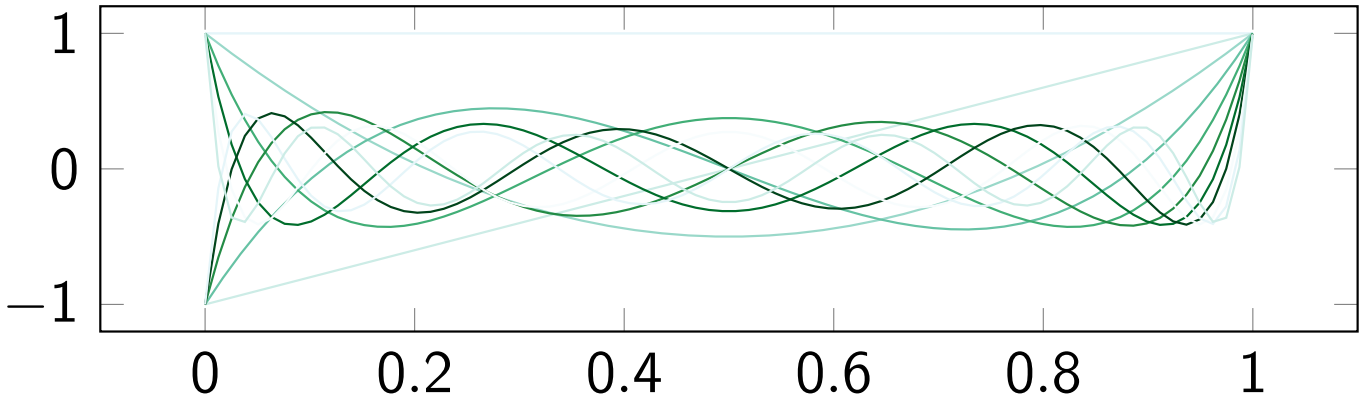
Legendre Memory Units

- An intriguing result is that the state represents the sliding window as coefficients in the basis of the first n **Legendre polynomials**:

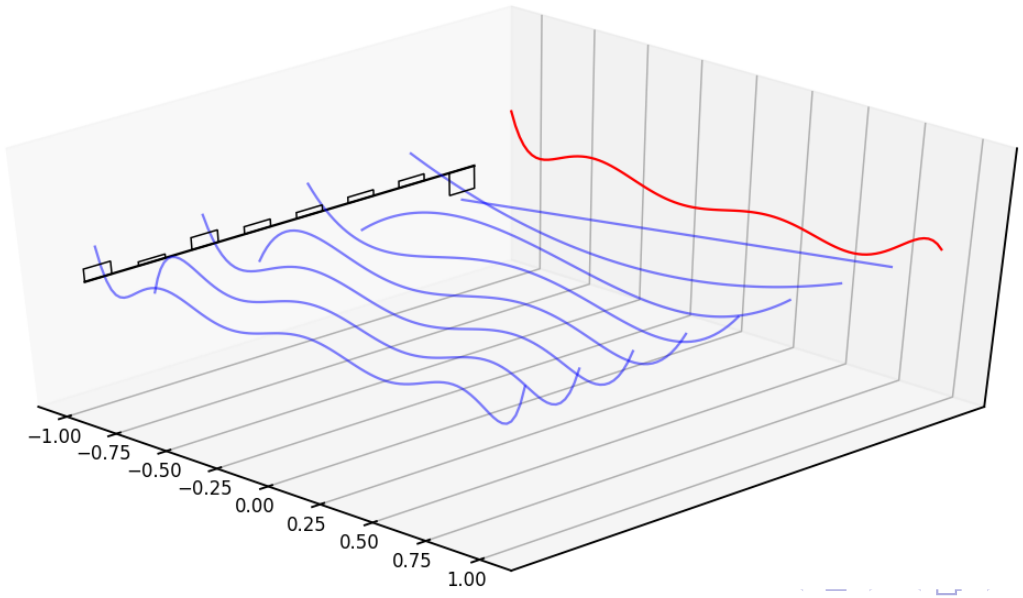
$$f(t-\theta') \approx \Sigma_{k=0}^{n-1} \quad \tilde{\mathcal{P}}_k \left(\frac{\theta'}{\theta} \right) m_k(t)$$

  where $\tilde{\mathcal{P}}_k$ is the kth shifted Legendre polynomial.
- Legendre polynomials are **orthogonal**, thus the resulting representation is **unique** and **optimal** (in L^2 sense).

Legendre Memory Units



First 11 shifted Legendre polynomials and an example function decomposition in the Legendre basis.



Interlude 2: Discretization

- So far we've been talking about *continuous* time models.
- They provide a powerful framework for analysis, but virtually all the data we are working with are discrete.
- Thus for putting them into practice we need to discretize the obtained equations.

Interlude 2: Discretization

The simplest method is the forward Euler:

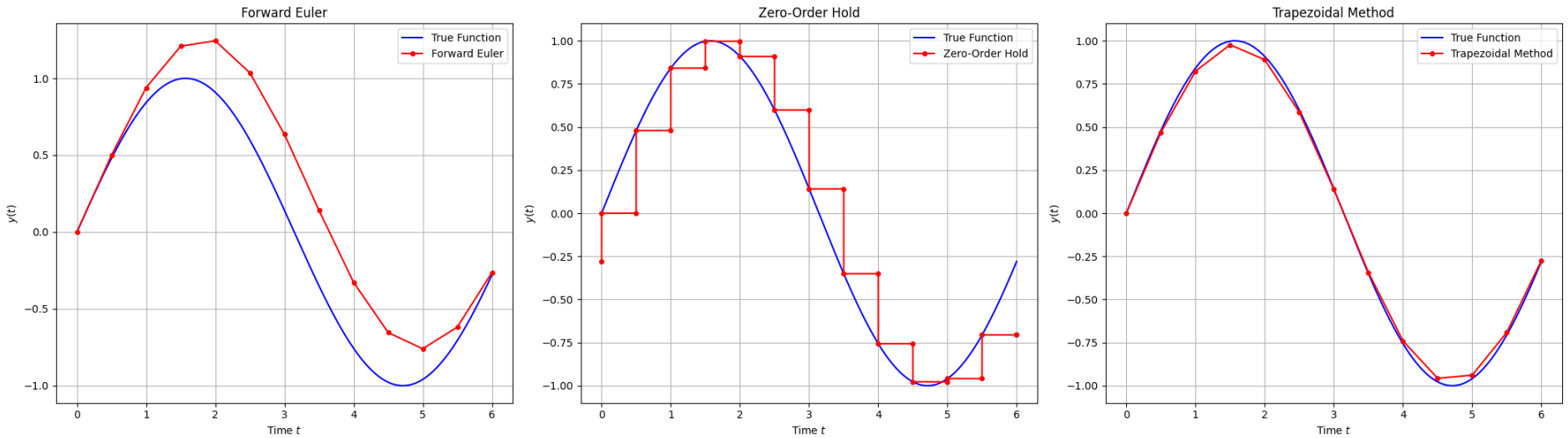
$$x_{t+1} = x_k + \Delta(Ax_t + Bu_t) = (I + \Delta A)x_t + \Delta B u_t = \bar{A}x_t + \bar{B}u_t$$

However, it's unstable and its value is mostly illustrative. The commonly used methods are:

$$\begin{aligned} \text{ZOH}: \quad & \bar{A} = e^{\Delta A}, && \bar{B} = (\Delta A)^{-1}(e^{\Delta A} - I)\Delta B \\ \text{Trapezoid}: \quad & \bar{A} = (I - \Delta A/2)^{-1}(I + \Delta A/2), && \bar{B} = (I - \Delta A/2)^{-1}\Delta B \end{aligned}$$

where Δ is some scalar time step.

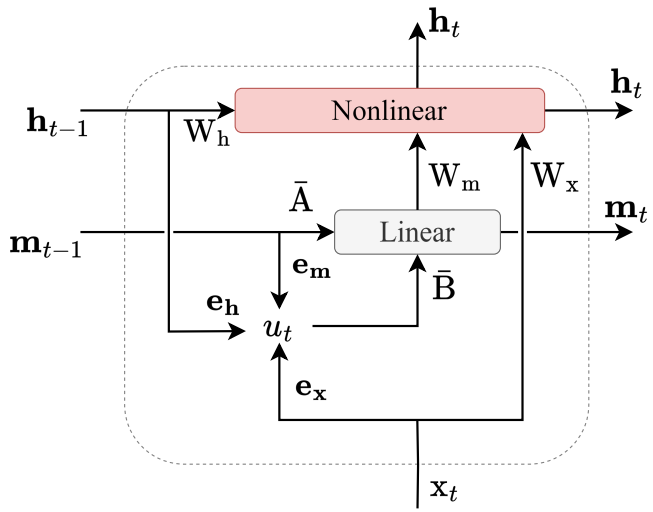
Interlude 2: Discretization



Discretization methods illustrated.

Legendre Memory Units: A Neural Network Module

- The LMU layer resembles the LSTM cell.
- m_t acts as a cell state and follows the derived SSM.



The cell had shown unprecedented results, modeling sequences up to 1B long in some setups, and can be implemented on neuromorphic HW.

Key takeaways so far (2)

- Imposing a structure on transition matrix A and input projection B can provide SSM with special abilities and interpretation.
- We have seen an example of such model derived from a task of implementing a continuous delay.
- SSM can be used as a part of a more general neural network module.

HiPPO: High-Order Polynomial Projection Operator

The following two sections are adapted from the Albert Gu PhD thesis.

Assume a more general problem setup:

- Consider the task of online function approximation of $f : \mathbb{R}^+ \to \mathbb{R}$
- We evaluate the approximation w.r.t some measure μ.
- The measure places a weight on different parts of the past.
- The measure is also time-dependent $\mu = \mu^{(t)}$ supported on $(-\infty, t]$.

HiPPO: High-Order Polynomial Projection Operator

- A natural basis for the task is a set of polynomials (OPs) orthogonal w.r.t. the measure:

$$\mathcal{P}_i, \mathcal{P}_k : \langle \mathcal{P}_i, \mathcal{P}_k \rangle_{\mu^{(t)}} = \int_a^b \mathcal{P}_i \mathcal{P}_k \, d\mu^{(t)} = \delta_{ik}$$

$$\mu^{(t)} > 0; a, b \in \mathbb{R} \cup \{-\infty, \infty\}$$

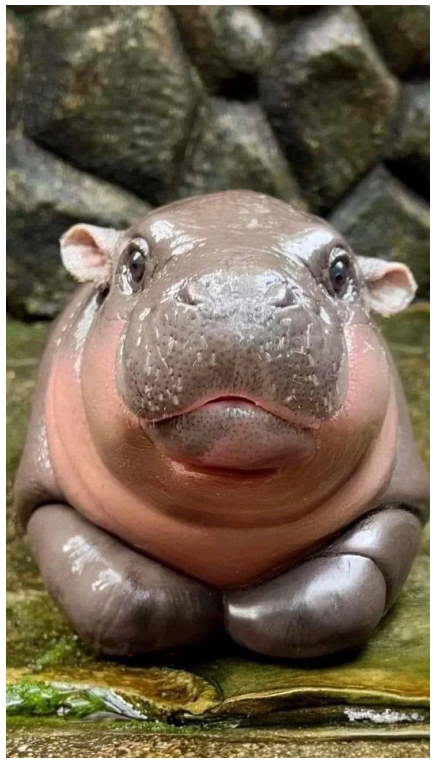
- Then the coefficients of the basis are simply $c_n^{(t)} = \langle f, \mathcal{P}_k \rangle_{\mu^{(t)}}$.
- Furthermore, differentiating by t through integral will often lead to c_n evolving in time: $\frac{d}{dt} c_n =$ function of $[c_k(t)_{k \in [N]}]$

HiPPO: High-Order Polynomial Projection Operator

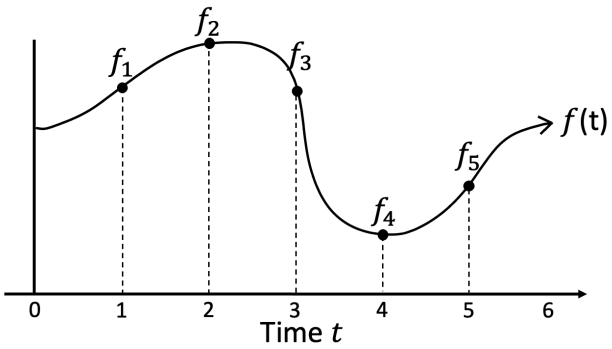
Given a finite basis $\mathcal{G}$ of the first N OPs, HiPPO is a composition of the two operators:

- proj_t: projects f to a polynomial in the span of $\mathcal{G}$.
- coef_t: returns the coefficients of the polynomial in terms of $\mathcal{G}$.

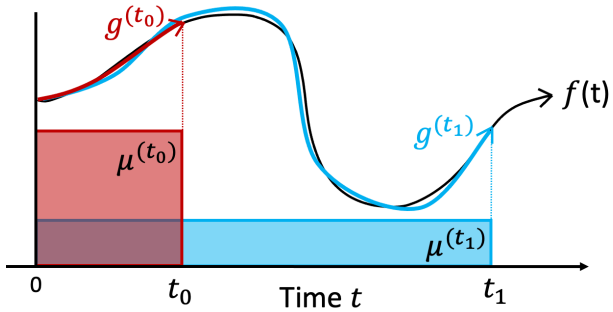
It can be shown, that $c(t) = [c_0(t), \ldots, c_n(t)]^T$ follow an ODE $c'(t) = A(t)c(t) + B(t)f(t)$.



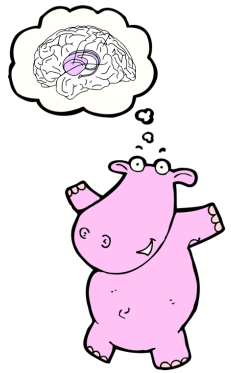
HiPPO: High-Order Polynomial Projection Operator



$\xrightarrow{\text{proj}_t}$



$\downarrow \text{coef}_t$



$$c(t_0) = \begin{bmatrix} 0.1 \\ -1.1 \\ 3.7 \\ 2.5 \end{bmatrix} \qquad c(t_1) = \begin{bmatrix} 1.5 \\ 2.9 \\ -0.3 \\ 2.0 \end{bmatrix}$$

Continuous-time HiPPO ODE

$$\frac{d}{dt}c(t) = A(t)c(t) + B(t)f(t)$$

$\xleftarrow{\text{discretize}}$

Discrete-time HiPPO Recurrence

$$c_{k+1} = A_k c_k + B_k f_k$$

Example HiPPO Instances

Different measures lead to models with different capabilities:

- LegT: uniform measure on $[t-\theta, t]$. Leads to the LMU up to rescaling.
- LegS: uniform measure on $[0, t]$ with state equations:

$$\bar{A}_{nk} = \begin{cases} -(2n+1)^{1/2}(2k+1)^{1/2}, & n > k \\ n+1, & n = k \\ 0, & n < k \end{cases}$$

$$\bar{B}_n = (2n+1)^{1/2}$$

LegS discretization is scale-equivariant, which allows to get rid of the Δ. It also has gradients bounded from both sides and can be discretized in $\mathcal{O}(N)$ steps by any method unlike LegT.

Example HiPPO Instances: LegT Results

Method	Error (MSE)	Speed (elements / s
LSTM	0.25	35,000
LMU (naive)	0.05	41,000
HIPPO-LegS	0.02	470,000

Table 4.1: Function approximation error after 1 million time steps, with 256 hidden units.

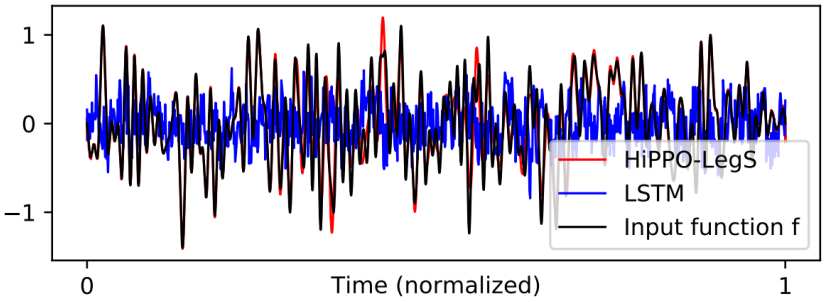
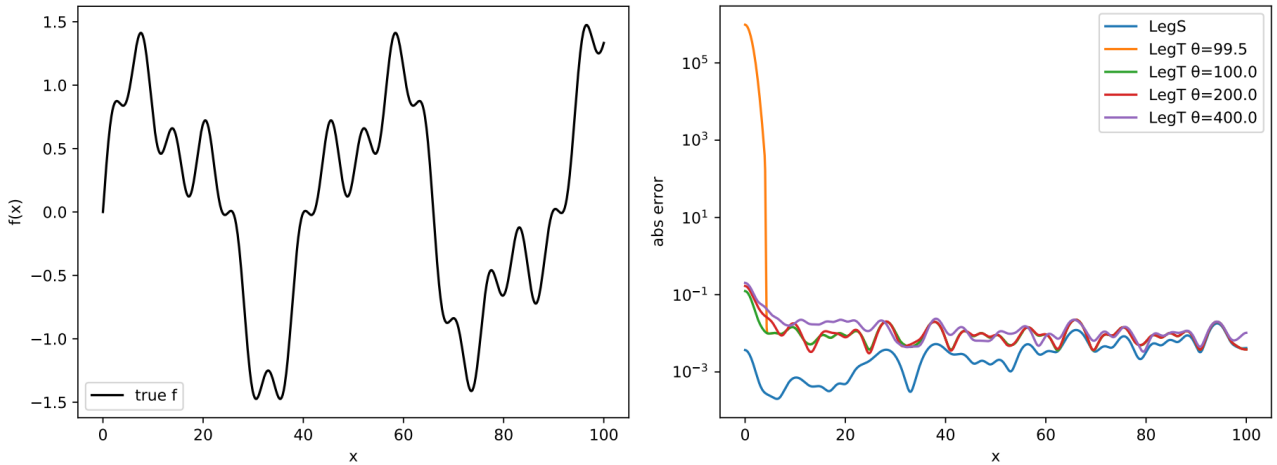


Figure 4.2: Input function and its reconstructions.



(a) True function $f(x)$

(b) Error for different θ's in LegT

Figure 4.3: Function approximation comparison between LegT and LegS. LegT error is sensitive to the choice of window length θ, especially if θ is smaller than the length of the true function.

HiPPO Generalization

The HiPPO framework can be generalized from OPs to any set of orthonormal functions. An example model is FouT derived around truncated Fourier basis:

- Due to Fourier basis approximates delay better than LegT.

$$A_{nk} = \begin{cases} -2, & n = k = 0 \\ -2\sqrt{2}, & n = 0, k \text{ is odd} \\ -2\sqrt{2}, & k = 0, n \text{ is odd} \\ -4, & n, k \text{ odd} \\ 2\pi k, & n - k = 1, k \text{ odd} \\ -2\pi n, & k - n = 1, n \text{ odd} \\ 0, & \text{otherwise} \end{cases}$$

$$B_n = \begin{cases} 2, & n = 0 \\ 2\sqrt{2}, & n \text{ odd} \\ 0, & \text{otherwise} \end{cases}$$

Key takeaways so far (3)

- We consider a task of online function approximation
- We can weight our past by different positive measures $\mu^{(t)}$
- HiPPO is a framework how to obtain state space matrices corresponding to these measures
- It has a strong foundation on lingebraic and approximation theory
- But tells nothing how to compute them fast :(

The Threefold View on SSMs

Assume we are given some A, B, trainable C and $D = 0$ (e.g. HiPPO).

- What kind of model do we have for discrete inputs?

SSMs are **all** of:

1. Continuous-time model
2. Recurrent model
3. Convolutional model

The Threefold View on SSMs

SSM is a continous model by definition. What are other interpretations?

- **Recurrent** view reveals some intricate connections to RNNs:
  - Gated recurrence is a backwards-Euler discretization of $\dot{x} = -x + u$.
  - Stacked SSMs with p.w. nonlinearities approximate $\dot{x} = -x + f(t, x)$.
- **Convolutional** view provides us with a faster way to compute output through a discrete convolution:

$$y = u * \bar{K}, \bar{K} = (C\bar{B}, C\bar{A}\bar{B}, \dots, C\bar{A}^k\bar{B}, \dots)$$

This can be done with FFT, but computing the kernel is a massive computational challenge on its own.

Structured State Spaces

- So far we have only addressed the modeling challenge.
- As we see, SSMs pose a computational challenge as well. What can we try?
  - Conjugates are equivalent: $(A, B, C) \sim (V^{-1}AV, V^{-1}B, CV)$

  Two main techniques:
  - Diagonalization
  - Normal Plus Low Rank (DPLR)

Structured State Spaces: Diagonalization

- ✓Easy powering, fast recurrent steps.
- ✓Diagonalizable matrices are dense in complex.
- ✗For HiPPO entries of V grow exponentially in N :(

Structured State Spaces: Normal Plus Low Rank

Normal matrix is diagonalizable by unitary V.

- ✓Main HiPPO matrices can be decomposed in such way.
- ✓Much faster to compute than original SSM.
- ✗Restrictive class.
- ✗Requires lingebraic computational trickery for implementation.

Structured State Spaces: Taking best of two worlds

- It appears, that by removing the "low rank" portion from DPLR from LegS can yield a good approximation (named S4-LegS).
  - Empirically it's slightly worse than LegS
- For other HiPPOs it doesn't work, nonetheless there exist other diagonal approximations.
- However, due to its properties S4-LegS can be considered as a "default" or "original" SSM for application.

Key takeaways so far (4)

- HiPPO and Structured State Space Models (S4) are two different approaches to two different problems.
- Their intersection is mostly diagonal approximations of HiPPO or diagonal-plus-low-rank HiPPO decompositions.
- Diagonal or DPLR structure allows to compute both recurrent and convolutional passes extremely fast.

General Setup

- Consider diagonal and DPLR S4s
- $A, B, C \in \mathbb{C}^{N/2}$ are initialized from HiPPO, C is trained
- Δ initialized in $(0.001, 0.1)$, can be different per channel
- Stack blocks with residual connections and apply per-block LayerNorms or BatchNorms.
- Concatenate forward and backward passes for bidirectional representation (e.g. in classification)

Speech

Table 7.3: (**Speech Commands classification.**) Test accuracy on 10- or 35- way keyword spotting. (*MFCC*) Training on MFCC-processed features (length 161). (*16k*) Training examples are 1-second audio waveforms sampled at 16000Hz, i.e. a 1 channel sequence of length 16000. (*8k*) Sampling rate change: 0-shot testing at 8000Hz where examples are constructed by naive decimation. ✗ denotes not applicable or computationally infeasible on single GPU.

(a) (**SC10 subset.**) Transformer, NODE, RNN, CNN, and SSM models.

	MFCC	16K Hz	8K Hz
Transformer	90.75	✗	✗
Performer	80.85	30.77	30.68
ODE-RNN	65.9	✗	✗
NRDE	89.8	16.49	15.12
UniCORNN	90.64	11.02	11.07
ExpRNN	82.13	11.6	10.8
CKConv	**95.3**	71.66	<u>65.96</u>
WaveGAN-D	✗	<u>96.25</u>	✗
S4	<u>93.96</u>	**98.32**	**96.30**

(b) (**Full dataset.**) S4 (DPLR and Diag) variants, and a collection of CNN baselines.

Model	Param.	16000Hz	8000Hz
S4-LegS	307K	96.08 (0.15)	91.32 (0.17)
S4-FouT	307K	95.27 (0.20)	91.59 (0.23)
S4D-LegS	306K	95.83 (0.14)	91.08 (0.16)
S4D-Inv	306K	<u>96.18</u> (0.27)	**91.80** (0.24)
S4D-Lin	306K	**96.25** (0.03)	<u>91.58</u> (0.33)
InceptionNet	481K	61.24 (0.69)	05.18 (0.07)
ResNet-18	216K	77.86 (0.24)	08.74 (0.57)
XResNet-50	904K	83.01 (0.48)	07.72 (0.39)
ConvNet	26.2M	95.51 (0.18)	07.26 (0.79)

Time Series Regression

Table 7.2: (**BIDMC Vital signs prediction.**) RMSE (std.) for predicting respiratory rate (RR), heart rate (HR), and blood oxygen (SpO2). *(Top)* S4. *(Middle)* Deep learning baselines. *(Bottom)* Other popular machine learning methods for time series. Citations indicate reported numbers from prior work.

Model	HR	RR	SpO2
S4	**0.332** (0.013)	**0.247** (0.062)	<u>0.090</u> (0.006)
S4-FouT	<u>0.339</u> (0.020)	0.301 (0.030)	**0.068** (0.003)
S4D-LegS	0.367 (0.001)	0.248 (0.036)	0.102 (0.001)
S4D-Inv	0.373 (0.024)	0.254 (0.022)	0.110 (0.001)
S4D-Lin	0.379 (0.006)	<u>0.226</u> (0.008)	0.114 (0.003)
UnICORNN [178]	1.39	1.06	0.869
coRNN [178]	1.81	1.45	-
CKConv	2.05	1.214	1.051
NRDE [140]	2.97	1.49	1.29
LSTM [178]	10.7	2.28	-
Transformer	12.2	2.61	3.02
XGBoost [199]	4.72	1.67	1.52
Random Forest [199]	5.69	1.85	1.74
Ridge Regress. [199]	17.3	3.86	4.16

Long Range Arena

Table 7.4: (**Long Range Arena.**) Test accuracy on full suite of LRA tasks. Hyperparameter in Appendix G.1.2 ✗ denotes failure to learn better than random guessing, following convention from Tay et al. [202]. (*Top*) Original Transformer variants in LRA [202]. (*Middle*) Other model reported in the literature. (*Bottom*) S4 variants.

Model	ListOps	Text	Retrieval	Image	Pathfinder	Path-X	Avg
Random	10.00	50.00	50.00	10.00	50.00	50.00	36.67
Transformer	36.37	64.27	57.46	42.44	71.40	✗	53.66
Local Attention	15.82	52.98	53.39	41.46	66.63	✗	46.71
Sparse Trans.	17.07	63.58	59.59	44.24	71.71	✗	51.03
Longformer	35.63	62.85	56.89	42.22	69.71	✗	52.88
Linformer	35.70	53.94	52.27	38.56	76.34	✗	51.14
Reformer	37.27	56.10	53.40	38.07	68.50	✗	50.56
Sinkhorn Trans.	33.67	61.20	53.83	41.23	67.45	✗	51.23
Synthesizer	36.99	61.68	54.67	41.61	69.45	✗	52.40
BigBird	36.05	64.02	59.29	40.83	74.87	✗	54.17
Linear Trans.	16.13	65.90	53.09	42.34	75.30	✗	50.46
Performer	18.01	65.40	53.82	42.77	77.05	✗	51.18
FNet	35.33	65.11	59.61	38.67	77.80	✗	54.42
Nyströmformer	37.15	65.52	79.56	41.58	70.94	✗	57.46
Luna-256	37.25	64.57	79.29	47.38	77.72	✗	59.37
S4D-Rand	60.08	86.93	90.39	83.80	91.12	95.70	84.67
S4D-Lin	60.76	87.36	90.66	89.21	95.52	97.26	86.80
S4D-Inv	60.36	87.47	90.62	88.57	94.98	**97.37**	86.56
S4D-LegS	60.99	87.34	90.65	89.04	95.35	97.33	86.78
S4-Rand	60.46	87.94	90.87	84.59	91.57	95.59	85.17
S4-FouT	**61.21**	86.72	90.47	89.18	95.68	✗	78.88
S4(-LegS)	60.86	**88.73**	**91.14**	**89.49**	**95.76**	97.32	**87.22**

Long Range Arena: PathX Task

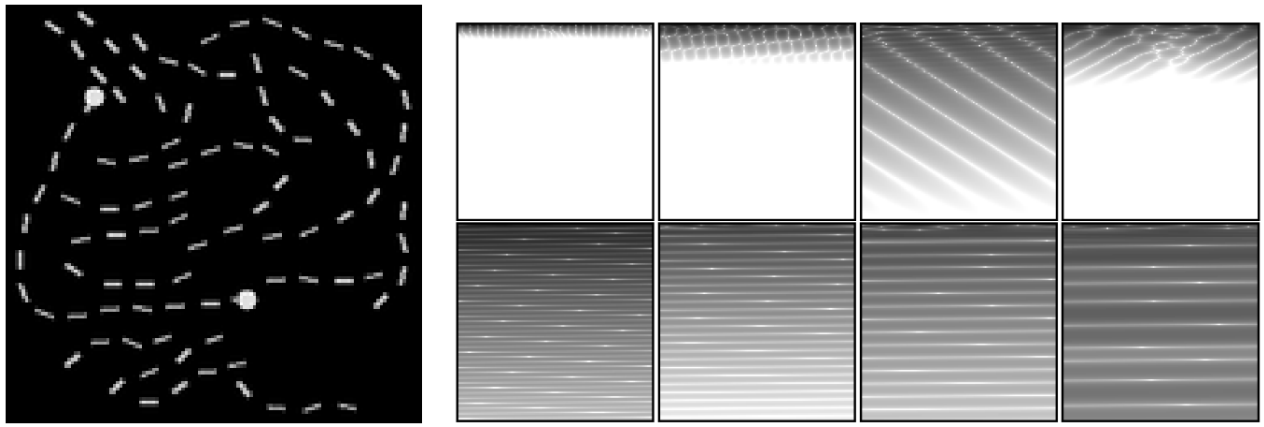


Figure 7.1: Visualizations of a trained S4 model on LRA Path-X. SSM convolution kernels $\overline{\boldsymbol{K}} \in \mathbb{R}^{16384}$ are reshaped into a 128×128 image. (*Left*) Example from the Path-X task, which involves deducing if the markers are connected by a path (*Top*) Filters from the first layer (*Bottom*) Filters from the last layer.

WikiText-103 Language Modeling

Table 7.7: (**WikiText-103 language modeling.**) S4 approaches the performance of Transformers with much faster generation. (*Top*) Transformer baseline which our implementation is based on, with attention replaced by S4. (*Bottom*) Attention-free models (RNNs and CNNs).

Model	Params	Test ppl.	Tokens / sec. (throughput)
Transformer [8]	247M	**20.51**	0.8K (1×)
GLU CNN [38]	229M	37.2	-
AWD-QRNN [137]	151M	33.0	-
LSTM + Cache + Hebbian + MbPA [169]	-	29.2	-
TrellisNet [12]	180M	29.19	-
Dynamic Convolution [227]	255M	25.0	-
TaLK Convolution [126]	240M	23.3	-
S4	249M	**20.95**	**48K** (**60**×)

Computational Complexity

Table 7.8: SSSMs: S4 (with Algorithm 1) is asymptotically more efficient than a naive SSM.

	Training Step (ms)			Memory Alloc. (MB)		
Dim.	128	256	512	128	256	512
SSSM	9.32	20.6	140.7	222.1	1685	13140
S4	4.77	3.07	4.75	5.3	12.6	33.5
Ratio	1.9×	6.7×	**29.6**×	42.0×	133×	**392**×

Table 7.9: Benchmarks vs. Transformers

	Length 1024		Length 4096	
	Speed	Mem.	Speed	Mem.
Transformer	1×	1×	1×	1×
Performer	1.23×	0.43×	3.79×	0.086×
Linear Trans.	**1.58**×	**0.37**×	**5.35**×	**0.067**×
S4	**1.58**×	0.43×	5.19×	0.091×

Discussion

- S4s are a promising sequence processing paradigm, excelling at signal processing
- The language domain is quite specific because of the language nature being far from physical process
- Transformer still excel because of their selectivity
- This is an ongoing research with promising results, e.g. Mamba