State Space Models: Nuts and Bolts

Abhishek Aich
NEC Laboratories, America

Abstract

We present a theoretical support of State Space Models, in particular focusing on Linear State Spae Layers, introduced in the paper '*Combining Recurrent, Convolutional, and Continuous-time Models with Linear State-Space Layers*' [1]. This report disentangles the underlying concept of the objective function of these models to provide a comprehensive overview. Suggestions/corrections are welcome.

1 Introduction

State space models (SSMs) [1] have been proposed as a possible alternative to (now) ubiquitous transformers. SSMs extended the idea proposed in [2] that capturing long range dependencies is equivalent to optimally storing and incorporating information from previous inputs in *memory*. Transformers[1] were known to have difficulty scaling to long sequence lengths, primarily due to the quadratic complexity of their self-attention mechanism [3]. Hence, authors of [2] proposed to treat *memory* as a problem of online function approximation where a function (representing past memory) $\boldsymbol{f}(t) : \mathbb{R}^+ \rightarrow \mathbb{R}$ is summarized by storing its optimal coefficients in terms of some basis functions. Following on [2] idea, [1] proposed *Linear State Space Layers* (LSSLs) for sequence modeling based on classical standard state space representations. Readers are recommended to go through this extremely well summarized video [4].

This report is dedicated to deciphering the minute details of LSSLs starting from the classical state space representations as the basics. Please do not hesitate to contact the author in case you find an error in the analysis.

2 The Classical (Continuous) "State Space Model"

In practice when we analyze any system, we start by developing a mathematical description of the system in the form of differential equations [5]. This is because differential equations provide a powerful and flexible framework for modeling, analyzing, and predicting the behavior of such systems.

A "state space" representation of a system is the mathematical model where the inputs, outputs and the system's internal components (that mathematically describe the system, also called as "states") [6] are related via *first-order differential equations*. Although there are other mathematical models [6, 7] for establishing the relationship between the aforementioned trio (e.g. using transfer functions), state space models leverage first-order differential equations for their conciseness (isolate each components or states and provide a representation) as well as the ease of efficient analysis (e.g. using linear algebra techniques for better system explainability). State-space models are one such method to convert a n-th order differential equation to *first-order matrix* differential equation [5] for their aforementioned useful properties.

Let's discuss the origins of the state space representation of a system. Denote an input and output (both function of time t) with $\boldsymbol{u}(t)$ and $\boldsymbol{y}(t)$, respectively. Consider a n-th order system described by

[1] this is in 2020.

a n-th order differential equation [6]

$$\frac{d^n \boldsymbol{y}(t)}{dt^n} + a_{n-1}\frac{d^{n-1}\boldsymbol{y}(t)}{dt^{n-1}} + \cdots a_1 \frac{d\boldsymbol{y}(t)}{dt} + a_0 \boldsymbol{y}(t) = b_0 \boldsymbol{u}(t) \tag{1}$$

Here, $a_i \forall i$ and b_0 represent the corresponding scalar coefficients. To understand how state-space models convert Eq. 1 into a batch of first-order differential equations, lets look at an example of a system defined by third-order differential equations (i.e. putting $n = 3$ in Eq. 1). We will drop the time function representation of all variables for simplicity.

$$\frac{d^3 \boldsymbol{y}}{dt^3} + a_2 \frac{d^2 \boldsymbol{y}}{dt^2} + a_1 \frac{d\boldsymbol{y}}{dt} + a_0 \boldsymbol{y} = b_0 \boldsymbol{u} \tag{2}$$

To convert this third-order differential equation to a set of first-order differential equations, we introduce the following three variables (all functions of time t) as follows.

$$\boldsymbol{h}_1 = \boldsymbol{y}, \boldsymbol{h}_2 = \frac{d\boldsymbol{y}}{dt}, \boldsymbol{h}_3 = \frac{d^2 \boldsymbol{y}}{dt^2} \tag{3}$$

For $\boldsymbol{h}_2$ and $\boldsymbol{h}_3$, we can also observe that

$$\boldsymbol{h}_2 = \frac{d\boldsymbol{h}_1}{dt}, \boldsymbol{h}_3 = \frac{d\boldsymbol{h}_2}{dt}, \quad \text{along with,} \frac{d\boldsymbol{h}_3}{dt} = \frac{d^3 \boldsymbol{y}}{dt^3}$$

Using above in Eq. 2, we get

$$\frac{d\boldsymbol{h}_3}{dt} + a_2 \boldsymbol{h}_3 + a_1 \boldsymbol{h}_2 + a_0 \boldsymbol{h}_1 = b_0 \boldsymbol{u} \tag{4}$$

We summarize the obtained first-order differential equations in Eq. 3, needed to describe Eq. 1, as follows.

$$\frac{d\boldsymbol{h}_1}{dt} = \boldsymbol{h}_2, \frac{d\boldsymbol{h}_2}{dt} = \boldsymbol{h}_3, \frac{d\boldsymbol{h}_3}{dt} = -(a_2 \boldsymbol{h}_3 + a_1 \boldsymbol{h}_2 + a_0 \boldsymbol{h}_1) + b_0 \boldsymbol{u} \tag{5}$$

Writing the above equations together gives us the following matrix form.

$$\begin{bmatrix} \frac{d\boldsymbol{h}_1}{dt} \\ \frac{d\boldsymbol{h}_2}{dt} \\ \frac{d\boldsymbol{h}_3}{dt} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \\ \boldsymbol{h}_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ b_0 \end{bmatrix} \boldsymbol{u} \tag{6}$$

Setting $\boldsymbol{h} = \begin{bmatrix} \frac{d\boldsymbol{h}_1}{dt} \\ \frac{d\boldsymbol{h}_2}{dt} \\ \frac{d\boldsymbol{h}_3}{dt} \end{bmatrix}$, $\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 \end{bmatrix}$, $\boldsymbol{B} = \begin{bmatrix} 0 \\ 0 \\ b_0 \end{bmatrix}$, we get the *state* equations. After computing the solution to this state equation, we can set $\boldsymbol{C} = [1 \quad 0 \quad 0]$ and arrive at the *output* equation.

$$\boldsymbol{y} = [1 \quad 0 \quad 0] \begin{bmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \\ \boldsymbol{h}_3 \end{bmatrix} \tag{7}$$

Hence, the final SSM representation is given as

$$\dot{\boldsymbol{h}} = \boldsymbol{A}\boldsymbol{h} + \boldsymbol{B}\boldsymbol{u} \tag{8a}$$

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{h} \tag{8b}$$

In general, for n-th order differential equation can be represented by ($\boldsymbol{D}$ turned out to be zero for above example, but it's usually not the case)

$$\begin{aligned} \dot{\boldsymbol{h}}(t) &= \boldsymbol{A}\boldsymbol{h}(t) + \boldsymbol{B}\boldsymbol{u}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{h}(t) + \boldsymbol{D}\boldsymbol{u}(t) \end{aligned} \tag{9}$$

In control system literature [7], $\boldsymbol{A}$ is the *state* matrix, $\boldsymbol{B}$ is the *input* matrix, $\boldsymbol{C}$ is the *output* matrix, and $\boldsymbol{D}$ is the *feedthrough* matrix. Further, we need to compute the solution of the state equation for using the output equation. We visualize the total representation in Fig. 1. Next, let's relate these fundamental SSM equations Eq. 9 to Linear State-Space Layers (LSSLs) introduced in [1]. For this, we will need to convert Eq. 9 to its discrete form.

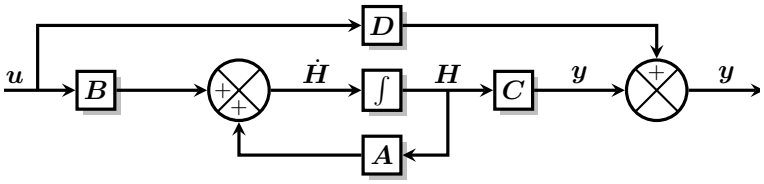

Figure 1: Classical SSMs

3 The Classical (Discrete) "State Space Model"

We aim to understand how [1] used SSMs (Eq. 9) as a black-box representation in a deep sequence model, where $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}$ are parameters learned by gradient descent. Similar to [8], we will assume $\boldsymbol{D} = 0$, interpreting it as an easily computable skip connection (can also be observed in Fig. 1).

To do this, we first convert the continuous-time equation group into discrete-time LSSLs (so that we can process discrete inputs such as image pixels). Essentially, we are estimating $\boldsymbol{y}(t)$ from $\boldsymbol{u}(t)$ using the system hidden state $\boldsymbol{h}(t)$. In particular, we leverage Generalized Bilinear Transform (GBT) [9]. GBT of Eq. 9 help us apply SSMs to operate on discrete input sequence $(\boldsymbol{u}_0, \boldsymbol{u}_1, \cdots)$ instead of continuous function $\boldsymbol{u}(t)$. The inputs $\boldsymbol{u}_k$ can be viewed as sampling the continuous signal $\boldsymbol{u}(t)$. We assume that the input $\boldsymbol{u}(t)$ is constant over the time interval Δ (also known as Zero-Order Hold assumption [10]), which is represented as follows.

$$\boldsymbol{u}_k = \boldsymbol{u}(k\Delta) \coloneqq \boldsymbol{u}[k] \quad \forall \quad k\Delta \leq t \leq (k+1)\Delta \tag{10}$$

Step size Δ represents the gap between time points (i.e. time-period *or* time-period-scale *or* time-scale [1]) from where $\boldsymbol{u}_k$ is sampled from $\boldsymbol{u}(t)$. This means that the length of dependencies (within the input sequence)that a (discrete) SSM can capture, is dependent on this value. This makes Δ an important factor in deep sequence modeling. So, lets first derive the discrete versions of Eq. 9 before computing the discrete versions of $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ (using GBT).

There are two ways [10] to convert Eq. 9 to discrete form. One is an *Approximate discretisation* (which uses the Euler's approximation) and the other one, which we will focus on, is the Exact discretisation. The exact discretisation requires us first to calculate the solution to Eq. 9, in particular, solving for $\dot{\boldsymbol{h}}(t) = A\boldsymbol{h}(t) + B\boldsymbol{u}(t)$. For clarity, we will replace the variable t with τ while computing the solution of this state equation (as t will appear again later).

It is know that for a scalar equation (where $\boldsymbol{h}(\tau) \in \mathbb{R}$ and a is some constant), $\dot{\boldsymbol{h}}(\tau) = a\boldsymbol{h}(\tau)$ has a solution of the form $\boldsymbol{h}(\tau) = e^{\boldsymbol{A}\tau}\boldsymbol{h}(0)$. Thus, in our case, where A is a matrix, we can start with the assumption that the solution is of the form $\boldsymbol{h}(\tau) = e^{\boldsymbol{A}\tau}\boldsymbol{h}(\tau)$ [10]. With this, let's multiply $e^{-A\tau}$ on both sides of $\dot{\boldsymbol{h}}(\tau) = A\boldsymbol{h}(\tau) + B\boldsymbol{u}(\tau)$ giving us the following.

$$e^{-A\tau}\dot{\boldsymbol{h}}(\tau) = e^{-A\tau}\big(A\boldsymbol{h}(\tau) + B\boldsymbol{u}(\tau)\big) \tag{11}$$

$$\implies e^{-A\tau}\dot{\boldsymbol{h}}(\tau) - e^{-A\tau}A\boldsymbol{h}(\tau) = e^{-A\tau}\boldsymbol{B}\boldsymbol{u}(\tau) \tag{12}$$

Noting that $\frac{d\left(e^{-A\tau}\boldsymbol{h}(\tau)\right)}{d\tau} = e^{-A\tau}\dot{\boldsymbol{h}}(\tau) - e^{-A\tau}A\boldsymbol{h}(\tau)$, we get

$$\frac{d\left(e^{-A\tau}\boldsymbol{h}(\tau)\right)}{d\tau} = e^{-A\tau}\boldsymbol{B}\boldsymbol{u}(\tau) \tag{13}$$

Integrating both sides of Eq. 13 between 0 to t gives us

$$e^{-A\tau}\boldsymbol{h}(\tau)\Big|_{\tau=0}^{\tau=t} = \int_{\tau=0}^{\tau=t} e^{-A\tau}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau \tag{14}$$

$$\implies e^{-At}\boldsymbol{h}(t) - e^{0}A\boldsymbol{h}(0) = \int_{\tau=0}^{\tau=t} e^{-A\tau}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau \tag{15}$$

Multiplying $e^{\boldsymbol{A}t}$ to both sides of Eq. 15 and noting $e^0 = \boldsymbol{I}$ (identity matrix), we get the solution of the state equation in Eq. 9 as follows

$$\boldsymbol{h}(t) = e^{\boldsymbol{A}t}\boldsymbol{h}(0) + \int_{\tau=0}^{\tau=t} e^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau \tag{16}$$

Inserting this solution Eq. 16 in the output equation of Eq. 9, we get the output prediction as follows.

$$\boldsymbol{y}(t) = \boldsymbol{C}\left(e^{\boldsymbol{A}t}\boldsymbol{h}(0) + \int_{\tau=0}^{\tau=t} e^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau\right) + \boldsymbol{D}\boldsymbol{u}(t) \tag{17}$$

Now that we have computed the solution in Eq. 16, we can now derive the discrete versions of Eq. 9. With discrete input defined as Eq. 10, we evaluate the state equation at the sampling instant $t = (k+1)\Delta$ as follows [10].

$$\boldsymbol{h}\big((k+1)\Delta\big) = e^{\boldsymbol{A}(k+1)\Delta}\boldsymbol{h}(0) + \int_{\tau=0}^{\tau=(k+1)\Delta} e^{\boldsymbol{A}((k+1)\Delta-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau \tag{18}$$

$$\implies \boldsymbol{h}\big((k+1)\Delta\big) = e^{\boldsymbol{A}\Delta}\left(e^{\boldsymbol{A}k\Delta}\boldsymbol{h}(0)\right) + \underbrace{\left(\int_{\tau=0}^{\tau=(k+1)\Delta} e^{\boldsymbol{A}((k+1)\Delta-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau\right)}_{\mathcal{I}} \tag{19}$$

Let's focus on the term $\mathcal{I}$. We can break this term in two intervals as follows.

$$\mathcal{I} = \underbrace{\int_{\tau=0}^{\tau=k\Delta} e^{\boldsymbol{A}((k+1)\Delta-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau}_{\mathcal{I}.1} + \underbrace{\int_{\tau=k\Delta}^{\tau=(k+1)\Delta} e^{\boldsymbol{A}((k+1)\Delta-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau}_{\mathcal{I}.2} \tag{20}$$

Taking $e^{\boldsymbol{A}\Delta}$ outside the integral in $\mathcal{I}.1$, and noting that $\boldsymbol{u}(\tau) = \boldsymbol{u}(k\Delta)$ in the interval $k\Delta \leq \tau \leq (k+1)\Delta$ for $\mathcal{I}.2$, we get

$$\mathcal{I} = e^{\boldsymbol{A}\Delta}\left(\int_{\tau=0}^{\tau=k\Delta} e^{\boldsymbol{A}(k\Delta-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau\right) + \left(\int_{\tau=k\Delta}^{\tau=(k+1)\Delta} e^{\boldsymbol{A}((k+1)\Delta-\tau)}\boldsymbol{B}d\tau\right)\boldsymbol{u}(k\Delta) \tag{21}$$

Putting Eq. 21 back in Eq. 19, we get

$$\begin{aligned}\boldsymbol{h}\big((k+1)\Delta\big) = e^{\boldsymbol{A}\Delta}\left(e^{\boldsymbol{A}k\Delta}\boldsymbol{h}(0)\right) + e^{\boldsymbol{A}\Delta}\left(\int_{\tau=0}^{\tau=k\Delta} e^{\boldsymbol{A}(k\Delta-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau\right) \\ + \left(\int_{\tau=k\Delta}^{\tau=(k+1)\Delta} e^{\boldsymbol{A}((k+1)\Delta-\tau)}\boldsymbol{B}d\tau\right)\boldsymbol{u}(k\Delta)\end{aligned} \tag{22}$$

$$\begin{aligned}\implies \boldsymbol{h}\big((k+1)\Delta\big) = e^{\boldsymbol{A}\Delta}\underbrace{\left(e^{\boldsymbol{A}k\Delta}\boldsymbol{h}(0) + \int_{\tau=0}^{\tau=k\Delta} e^{\boldsymbol{A}(k\Delta-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau\right)}_{\mathcal{II}} \\ + \underbrace{\left(\int_{\tau=k\Delta}^{\tau=(k+1)\Delta} e^{\boldsymbol{A}((k+1)\Delta-\tau)}\boldsymbol{B}d\tau\right)}_{\mathcal{III}}\boldsymbol{u}(k\Delta)\end{aligned} \tag{23}$$

The term $\mathcal{II}$ is clearly equivalent to $\boldsymbol{h}(k\Delta)$ if we set $t = k\Delta$ in Eq. 16. The term $\mathcal{III}$ can be further simplified by setting $\epsilon = (k+1)\Delta - \tau$. This also means $d\tau = -d\epsilon$.

$$\mathcal{III} = \int_{\epsilon=\Delta}^{\epsilon=0} e^{\boldsymbol{A}(\epsilon)}B(-d\epsilon) = \int_{\epsilon=0}^{\epsilon=\Delta} e^{\boldsymbol{A}\epsilon}\boldsymbol{B}d\epsilon \tag{24}$$

If we further simplify $\mathcal{III}$ to remove the integral, we can use the following steps [11]. Assuming $\boldsymbol{A}$ is invertible, we can write $\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix. Then, the term $\mathcal{III}$ can be written as follows.

$$\mathcal{III} = \boldsymbol{A}^{-1}\left(\int_{\epsilon=0}^{\epsilon=\Delta} \boldsymbol{A}e^{\boldsymbol{A}\epsilon}d\epsilon\right)\boldsymbol{B} \tag{25}$$

With $\frac{d(e^{\boldsymbol{A}\epsilon})}{d\epsilon} = \boldsymbol{A}e^{\boldsymbol{A}\epsilon}$,

$$\mathcal{III} = \boldsymbol{A}^{-1}\left(\int_{\epsilon=0}^{\epsilon=\Delta} \frac{d(e^{\boldsymbol{A}\epsilon})}{d\epsilon}d\epsilon\right)\boldsymbol{B} = \boldsymbol{A}^{-1}\big(e^{\boldsymbol{A}\epsilon}\big|_{\epsilon=0}^{\epsilon=\Delta}\big)\boldsymbol{B} = \boldsymbol{A}^{-1}\big(e^{\boldsymbol{A}\Delta} - \boldsymbol{I}\big)\boldsymbol{B} \tag{26}$$

Putting $\mathcal{I}$ and $\mathcal{II}$ back in Eq. 22, we obtain the discretized state space equation.

$$\boldsymbol{h}\big((k+1)\Delta\big) = \Big(e^{\boldsymbol{A}\Delta}\Big)\boldsymbol{h}\big(k\Delta\big) + \Big(\boldsymbol{A}^{-1}\big(e^{\boldsymbol{A}\Delta} - \boldsymbol{I}\big)\boldsymbol{B}\Big)\boldsymbol{u}(k\Delta) \tag{27}$$

Similar to computing the solution (Eq. 16) of continuous-time state equation in Eq. 9, we now compute the solution of Eq. 27 using the Iterative Method. Let $\overline{\boldsymbol{A}} = e^{\boldsymbol{A}\Delta}, \quad \overline{\boldsymbol{B}} = \boldsymbol{A}^{-1}\big(e^{\boldsymbol{A}\Delta} - \boldsymbol{I}\big)\boldsymbol{B}$. Start by setting $k = 0$ and $k = 1$ (drop Δ in the notation for brevity).

$$\boldsymbol{h}(1) = \overline{\boldsymbol{A}}\boldsymbol{h}(0) + \overline{\boldsymbol{B}}\boldsymbol{u}(0) \tag{28}$$

$$\boldsymbol{h}(2) = \overline{\boldsymbol{A}}\boldsymbol{h}(1) + \overline{\boldsymbol{B}}\boldsymbol{u}(1) \tag{29}$$

Let's use Eq. 28 in Eq. 29.

$$\begin{aligned}\boldsymbol{h}(2) &= \overline{\boldsymbol{A}}\big(\overline{\boldsymbol{A}}\boldsymbol{h}(0) + \overline{\boldsymbol{B}}\boldsymbol{u}(0)\big) + \overline{\boldsymbol{B}}\boldsymbol{u}(1) \\ &= \overline{\boldsymbol{A}}^2\boldsymbol{h}(0) + \overline{\boldsymbol{A}\boldsymbol{B}}\boldsymbol{u}(0) + \overline{\boldsymbol{B}}\boldsymbol{u}(1)\end{aligned} \tag{30}$$

We can keep performing the same steps for $k = 3$, which leads to the following.

$$\boldsymbol{h}(3) = \overline{\boldsymbol{A}}^3\boldsymbol{h}(0) + \overline{\boldsymbol{A}}^2\overline{\boldsymbol{B}}\boldsymbol{u}(0) + \overline{\boldsymbol{A}\boldsymbol{B}}\boldsymbol{u}(1) + \overline{\boldsymbol{B}}\boldsymbol{u}(2) \tag{31}$$

Clearly, we can generalize this form to the following [10].

$$\boldsymbol{h}(k) = \overline{\boldsymbol{A}}^k\boldsymbol{h}(0) + \sum_{n=0}^{k-1}\overline{\boldsymbol{A}}^{(k-1-n)}\overline{\boldsymbol{B}}\boldsymbol{u}(n) \tag{32}$$

This state solution now can be used with the discretized output equation. To obtain the output equation, we set $t = k\Delta$ in Eq. 17 and noting the occurrence of the term $\mathcal{I}$.

$$\boldsymbol{y}(k\Delta) = \boldsymbol{C}\underbrace{\left(e^{\boldsymbol{A}k\Delta}\boldsymbol{h}(0) + \int_{\tau=0}^{\tau=k\Delta} e^{\boldsymbol{A}(k\Delta-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau\right)}_{\mathcal{I}} + \boldsymbol{D}\boldsymbol{u}(k\Delta) \tag{33}$$

$$\implies \boldsymbol{y}(k\Delta) = \boldsymbol{C}\boldsymbol{h}(k\Delta) + \boldsymbol{D}\boldsymbol{u}(k\Delta) \tag{34}$$

To summarize, we have the following discrete SSM of Eq. 9.

$$\begin{aligned}\boldsymbol{h}_{k+1} &= \overline{\boldsymbol{A}}\boldsymbol{h}_k + \overline{\boldsymbol{B}}\boldsymbol{u}_k \\ \boldsymbol{y}_k &= \overline{\boldsymbol{C}}\boldsymbol{h}_k + \overline{\boldsymbol{D}}\boldsymbol{u}_k\end{aligned} \tag{35}$$

where,

$$\overline{\boldsymbol{A}} = e^{\boldsymbol{A}\Delta}, \quad \overline{\boldsymbol{B}} = \boldsymbol{A}^{-1}\big(e^{\boldsymbol{A}\Delta} - \boldsymbol{I}\big)\boldsymbol{B}, \quad \overline{\boldsymbol{C}} = \boldsymbol{C}, \quad \overline{\boldsymbol{D}} = \boldsymbol{D} \tag{36}$$

This discrete SSM in Eq. 35 is exactly the same as `Eq. (4)` in Mamba [12]. With the parameter transformation from $(\Delta, \boldsymbol{A}, \boldsymbol{B})$ to $(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}})$, the LSSL based sequence model can be represented in following two ways: either as a linear recurrence or as a global convolution.

4 State Space Sequence Models

Now, that we have arrived at the discrete form of SSMs in Eq. 35 that can handle discrete input sequences $\boldsymbol{u}_k$, lets formalize the notations and nomenclature before we discuss SSM's application for deep sequence modeling, following [1].

Given a fixed state space representations $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$, an LSSL is the sequence-to-sequence mapping defined by discretizing the linear state-space model Eq. 9 (which we derived in Eq. 35). Eq. 35 can be directly interpreted as a Recurrent Neural Network (RNN) representation. In particular, Eq. 35 maps the sequence $\boldsymbol{u}_k$ to $\boldsymbol{y}_k$ using the hidden state in $\boldsymbol{h}_k$. However to process the (new) sequence $\boldsymbol{u}_{k+1}$, we need to recompute this hidden state $\boldsymbol{h}_{k+1}$ recursively using the transition matrix $\overline{\boldsymbol{A}}$. Hence, we will leverage the *convolutional representation*.

4.1 The Convolutional Representation

Let's go back to the solution of the discrete state equation in Eq. 32.

In Eq. 32, assume $\boldsymbol{h}(0) = 0$ and write $\boldsymbol{w}[k-1-n] = \overline{\boldsymbol{A}}^{(k-1-n)}\overline{\boldsymbol{B}}$ as follows.

$$\boldsymbol{h}[k] = \sum_{n=0}^{k-1} \boldsymbol{w}[k-1-n]\boldsymbol{u}[n] \tag{37}$$

We can note that Eq. 37 represents the discrete convolution operation between the interval $0 \leq n \leq k-1$. Insert Eq. 37 in output equation of Eq. 35 to get

$$\begin{aligned} \boldsymbol{y}[k] &= \boldsymbol{C}\left(\sum_{n=0}^{k-1} \boldsymbol{w}[k-1-n]\boldsymbol{u}[n]\right) + \boldsymbol{D}\boldsymbol{u}[k] \\ &= \overline{\boldsymbol{K}} * \boldsymbol{u}[k] + \boldsymbol{D}\boldsymbol{u}[k] \end{aligned} \tag{38}$$

With $\overline{\boldsymbol{K}}$, the convolution operation can be computed comparatively efficiently than the RNN form [1].

5 Conclusion

We provided an overview of Linear State Space Layers (LSSLs), which are the core components of State Space Models (SSMs). We traced their development from classical continuous-time models to discrete-time formulations, demonstrating their application in deep sequence modeling based on solid mathematical principles. Finally, we discussed their efficient convolutional representation, which makes LSSLs well-suited for modern deep sequence modeling.

References

[1] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in neural information processing systems*, 34:572–585, 2021.

[2] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33: 1474–1487, 2020.

[3] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020.

[4] Samuel Albanie. Mamba - a replacement for transformers?, 2023. URL https://youtu.be/ouF-H35atOY?si=mRJfTM_-sTBBdDny. Accessed: September 8, 2024.

[5] Erik Cheever. Representing linear physical systems with differential equations. Online, 2022. URL https://lpsa.swarthmore.edu/Representations/SysRepDiffEq.html. Accessed: 2024-05-30.

[6] Ramkrishna Pasumarthy and Viswanath Talasila. Introduction to state space systems, 2017. URL https://youtu.be/xajgSUci9zs?feature=shared. Accessed: 2024-05-29.

[7] Richard Bishop and Robert H C Dorf. *Modern control systems*. Pearson, 2011.

[8] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

[9] Guofeng Zhang, Tongwen Chen, and Xiang Chen. Performance recovery in digital implementation of analogue systems. *SIAM journal on control and optimization*, 45(6):2207–2223, 2007.

[10] Julio H. Braslavsky. Lecture notes in control system design and management (elec4410), 2006. URL https://www.eng.newcastle.edu.au/~jhb519/teaching/elec4410/lectures/Lec11.pdf.

[11] Zhi Jane Li. Lecture notes in synergy of human and robotic systems (rbe59), 2017. URL https://users.wpi.edu/~zli11/teaching/rbe595_2017/LectureSlide_PDF/discretization.pdf.

[12] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[13] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[14] Doron Levy. Lecture notes in introduction to numerical analysis i (amsc 466), 2016. URL https://math.umd.edu/~dlevy/classes/amsc466/notes.pdf.

[15] Jeffrey Wong. Lecture notes in applied computational analysis (math 563), 2020. URL https://services.math.duke.edu/~jtwong/math563-2020/lectures/Lec4-orth.pdf.