# Unleash the Full Potential of State Space Models with Complex States

March 10, 2025

**Abstract**

The state space models (SSM) are powerful and gaining popularity in the machine learning community. The state transition matrix, or simply the state matrix, is the key component of an SSM that accounts for its long-term memories. However, it is often designed as certain ad-hoc constant matrices that facilitate computations. This note suggests that in order to fully unleash the potential of SSMs, we need to learn that state matrix along with other ones as well. This practice is scalable with respect to the state size only when the SSM is rewritten as a modal decomposition, i.e., eigenvalue decomposition (EVD), form and its states are saved as complex vectors, even when the original SSM only involves real variables. The root cause for this is that the EVD is always feasible only in the domain of complex numbers even for real matrices.

## 1 Introduction

The state space model (SSM) or representation is one of the key concepts in control science and signal processing for analyzing the properties of a linear system, say its controllability, observability, stability and behaviors of its different modals. In recent years, the SSMs are also gaining popularity in machine learning due to their representation powerfulness and computational efficiency [1]. However, these works tend to use certain ad-hoc constant state transition matrices, say the HiPPO matrix proposed in [1]. Although they have been empirically proven to work well for a lot of real world problems, there is no theoretical guarantee that they can learn any form of SSMs or these manually designed state matrices are universally economic for different types of problems. Furthermore, their implementations can be quite involved and vastly different when switching the state matrices. This note revisits the SSMs and proposes to learn the state matrices directly. To achieve this, we need to consider the modal decomposition or eigenvalue decomposition (EVD) form of an SSM, and complex state vectors arise naturally in the decompositions, even for problems only involving real numbers.

## 2 The EVD and Convolution Forms of SSMs

A typical discrete SSM can be defined as

$$\begin{aligned} x_t &= Ax_{t-1} + Bu_t \\ y_t &= Cx_t + Du_t \end{aligned} \tag{1}$$

where $t \geq 1$ is the discrete time index, $x_0$ is the system initial state vector, $u_t$, $x_t$ and $y_t$ are the input or control vector, state vector, and output vector, respectively, $A$, $B$, $C$ and $D$ are called the state or system matrix, the input matrix, the output matrix, and the feedthrough or feedforward matrix, respectively. For an SSM with state vector size $N$ and sequence length $L$, a naive implementation of (1) leads to the complexity of $\mathcal{O}(LN^2)$, which is too high for large $L$ and $N$. Also, the recursive form of (1) is not friendly for today's hardware that favors operations can be parallelized.

### 2.1 The Modal Decomposition Form of SSM

The first technique to scale up the SSM computations is to convert the SSM form in (1) into an equivalent EVD or modal decomposition form such that the matrix-vector product operation, $Ax_t$, reduces to element-wise vector product. Let the EVD of $A$ be

$$A = V\Lambda V^{-1} \tag{2}$$

where the columns of $V$ contain the eigenvectors, and the diagonals of $\Lambda$ have the eigenvalues. Note that both $V$ and $\Lambda$ can be complex matrices even if $A$ is a real matrix. We further assume that $\Lambda$ is a diagonal matrix, and dismiss the degenerated cases where $\Lambda$ is almost diagonal but may have a few Jordan blocks [2]. After substituting (2) into (1), we can rewrite (1) as

$$\begin{aligned}(V^{-1}x_t) &= \Lambda(V^{-1}x_{t-1}) + (V^{-1}B)u_t \\ y_t &= (CV)(V^{-1}x_t) + Du_t \end{aligned} \tag{3}$$

Hence, we can simplify (1) into to the following modal decomposition form,

$$\begin{aligned}\bar{x}_t &= \Lambda\bar{x}_{t-1} + \bar{B}u_t \\ y_t &= \mathrm{Re}\left(\bar{C}\bar{x}_t\right) + Du_t \end{aligned} \tag{4}$$

where $\bar{x}_t = V^{-1}x_t$ is the complex state vector, $\bar{B}$ is the complex input matrix, $\bar{C}$ is the complex output matrix, and $\mathrm{Re}(\cdot)$ takes the real part of a complex vector to keep our outputs always be real. From (4), it is clear that the new state matrix is simply a diagonal matrix (or almost diagonal in the worst case when $\Lambda$ has a few Jordan blocks).

## 2.2 The Convolution Form of SSM

The second technique to scale up SSM is fast convolution. Note that an SSM is equivalent to a filter. Hence, we can rewrite it as a convolution. Let us expand (4) up to $\ell$ steps to have the following details

$$\begin{aligned}\bar{x}_1 =& \Lambda\bar{x}_0 + \bar{B}u_1 \\ \bar{x}_2 =& \Lambda^2\bar{x}_0 + \Lambda\bar{B}u_1 + \bar{B}u_2 \\ \bar{x}_3 =& \Lambda^3\bar{x}_0 + \Lambda^2\bar{B}u_1 + \Lambda\bar{B}u_2 + \bar{B}u_3 \\ &\vdots \\ \bar{x}_\ell =& \Lambda^\ell\bar{x}_0 + \Lambda^{\ell-1}\bar{B}u_1 + \ldots + \Lambda\bar{B}u_{\ell-1} + \bar{B}u_\ell \end{aligned} \tag{5}$$

With the notation of convolution, we can rewrite (5) compactly as

$$\bar{x}_{1:\ell} = \Lambda^{1:\ell}\bar{x}_0 + \mathrm{Conv}(\Lambda^{0:\ell-1}, \bar{B}u_{1:\ell}) \tag{6}$$

where $\bar{x}_{1:\ell}$ denotes the sequence $\{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_\ell\}$, $\Lambda^{0:\ell-1}$ denotes the sequence $\{I, \Lambda, \Lambda^2, \ldots, \Lambda^{\ell-1}\}$, $\bar{u}_{1:\ell}$ denotes the sequence $\{\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_\ell\}$, and $\mathrm{Conv}(\cdot, \cdot)$ denotes the convolution of two sequences. The convolution operation in (6) can be accelerated with the well-known Fast Fourier transform (FFT).

## 2.3 Summary of the Proposed SSM

Eq. (4) gives the recursive view of the proposed SSM. All the four matrices in (4), $\Lambda$, $\bar{B}$, $\bar{C}$ and $D$ are learnable parameters. Note that unlike the standard SSM form in (1), now only $D$ is real, and the other three matrices are complex ones even for problems only involving real numbers. The state matrix $\Lambda$ is a diagonal complex matrix. Generally, there is no need to consider the degenerated cases where $\Lambda$ is almost diagonal, but has a few Jordan blocks.

Combining (4) and (6), we can compute the states and outputs via fast convolution as

$$\begin{aligned}\bar{x}_{1:\ell} &= \Lambda^{1:\ell}\bar{x}_0 + \mathrm{Conv}(\Lambda^{0:\ell-1}, \bar{B}u_{1:\ell}) \\ y_{1:\ell} &= \mathrm{Re}\left(\bar{C}\bar{x}_{1:\ell}\right) + Du_{1:\ell} \end{aligned} \tag{7}$$

| Train loss \ Test classification error rate | Adam optimizer | PSGD optimizer |
|---|---|---|
| Real state SSM | 0.859\31.5% | 0.596\23.1% |
| Complex state SSM | 0.039\2.59% | 0.024\3.2% |

Table 1: Median train losses and test classification error rates under different settings.

## 3 Numerical Results

To verify the above theory, we consider the famous MNIST hand written digit recognition task. The SSM can only read one pixel value per step, and needs to give a class label estimation at the last step. Hence, it needs to keep long term memories in order to perform well in this task. It have been previously shown that a real diagonal or random state matrix initialization cannot perform well for such tasks [1]. If our theory is correct, an SSM with complex state and complex diagonal state matrix should be able to solve this problem as expected.

Table 1 summarizes the median cross entropy train losses and test classification error rates over five runs of the two types of SSMs with two optimizers and state vector size 128. Except for the popular Adam optimizer, another one considered is the preconditioned SGD (PSGD) [3]. From Table 1, we see that the SSMs with real states perform significantly worse than the ones with complex states. The difficulty of training a recurrent model cannot explain this vast performance difference. Actually, the PSGD optimizer already greatly outperforms Adam in both the train and test metrics for the real state SSMs. Still, it cannot close the performance gap between these two types of SSMs. Further increasing the state size of real state SSMs can only provides marginal helps. On the other hand, the complex state SSMs are already on par with a tiny fully connected feedforward network that has about 128 neurons. Thus, they must have been successfully memorized the long range contexts. For further experimental details, please check up our code `https://github.com/lixilinx/Complex-state-SSM`.

## 4 Conclusions

This note proposes to learn the state matrix of a state space model (SSM) in the domain of complex numbers even for problems only involving real numbers. The rationale is that the eigenvalue decomposition (EVD) of a real state matrix generally may not exist in the domain of real numbers. Thus, it is necessary to work in the domain of complex numbers with the modal decomposition or EVD form of an SSM. As a result, the state vectors must be complex. We call this form of SSM the complex state SSM. Numerical results have confirmed that it indeed can solve the problems that require long range memories and fail SSMs with real state vectors and real diagonal state matrices.

## References

[1] Albert Gu, Karan Goel, and Christopher Re, “Efficiently modeling long sequences with structured state spaces,” arXiv: `https://arxiv.org/pdf/2111.00396`, 2022.

[2] G. H. Golub and C. V. Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

[3] X. L. Li, “Preconditioned stochastic gradient descent,” *IEEE Trans. Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 454–1466, 2018.