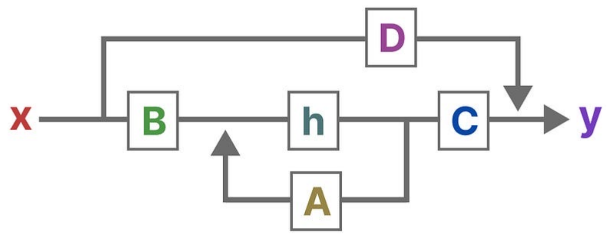
Mamba: Linear-Time Sequence Modeling with Selective State Spaces

S4 (hard mode)

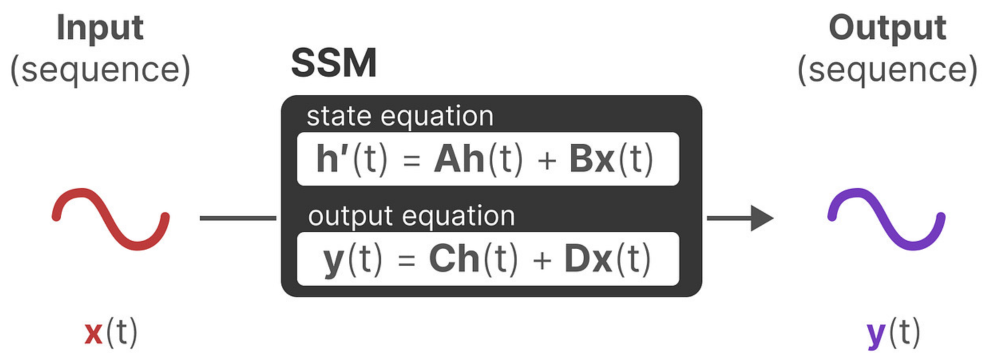
Chapter

- 1. Review Easy Mode
- 2. SSM Truncated Generation Functions
- 3. Diagonal Cases and DPLR
- 4. Code Explanation for S4

Review S4 (Contributions)



State Space Model

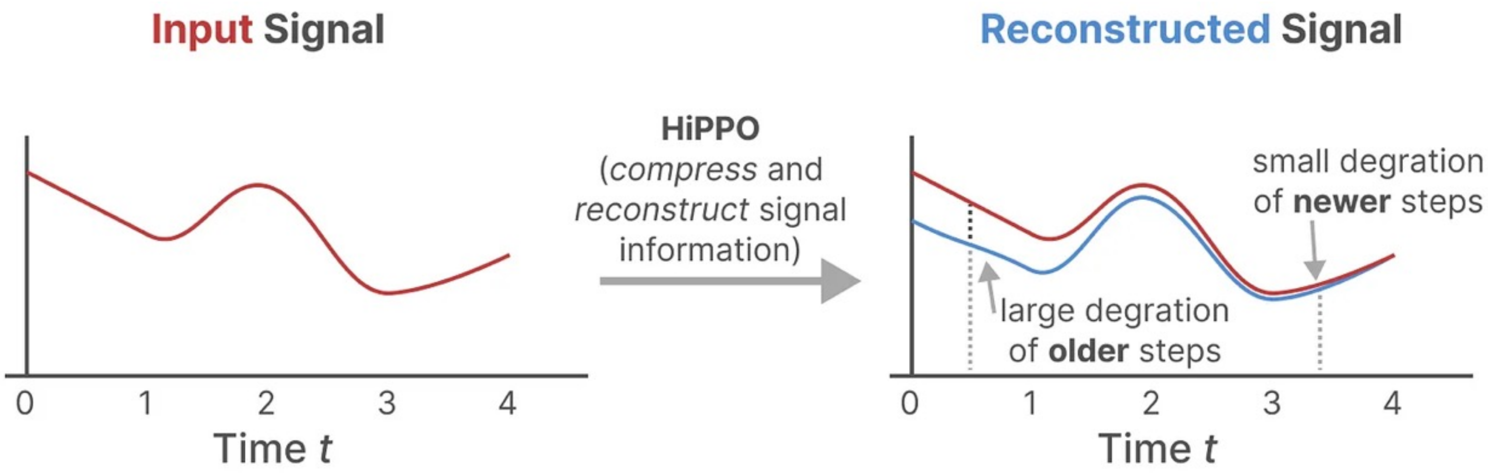


Train: (A, B, C, D)

(1) Cannot Remember Long Info

(2) Not Parallel (not efficient)

Review S4 (Cannot Remember Long Info)



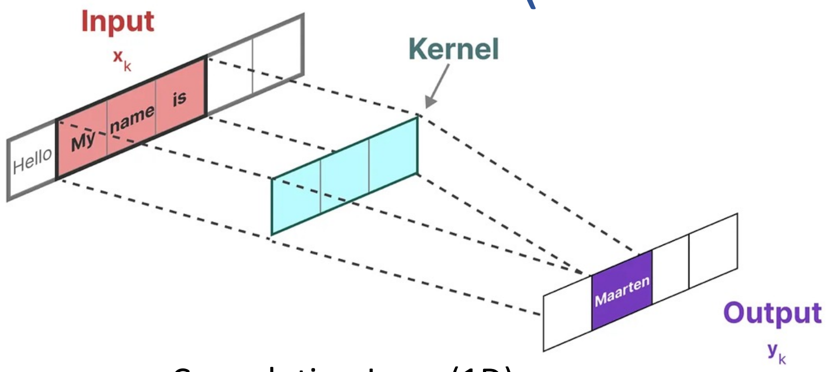
Mathematically, it does so by tracking the coefficients of a Legendre polynomial which allows it to approximate all of the previous history.

$$\textbf{(HiPPO Matrix)} \qquad \boldsymbol{A}_{nk} = -\begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases}.$$

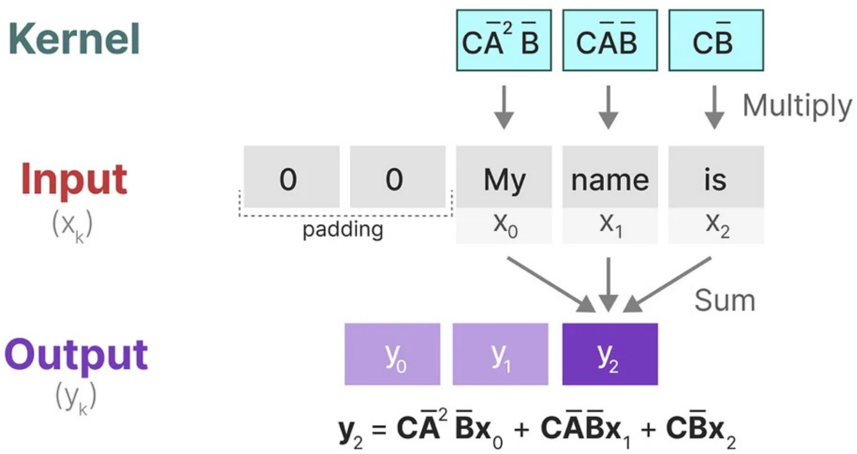
https://hazyresearch.stanford.edu/blog/2020-12-05-hippo

https://proceedings.neurips.cc/paper/2019/hash/952285b9b7e7a1be5aa7849f32ffff05-Abstract.html

Review S4 (Not Parallel)



Convolution Layer(1D)



$$y = \bar{K} * u$$

Convolutional Representation

time	hidden	prediction
0	$\bar{B}x_0$	$C\bar{B}x_0$
1	$\bar{A}h_0 + \bar{B}x_1$	$C\bar{A}\bar{B}x_0 + C\bar{B}x_1$
2	$\bar{A}(\bar{A}h_0 + \bar{B}x_1) + \bar{B}x_2$	$C\bar{A}^2\bar{B}x_0 + C\bar{A}\bar{B}x_1 + C\bar{B}x_2$
…	…	…

If conv size is fixed, like conv size = 3,
Conv kernel should be: $\bar{K} = (C\bar{A}^{L-1}B, \dots, C\bar{A}^2\bar{B}, C\bar{A}\bar{B}, C\bar{B})$

Difficulties: L times for computing $y = \bar{K} * x$
Solve: truncated generating function (convert the power of matrix to the inverse of matrix, like

$$y = fun_1(A) * fun_2(A^{-1}) * x$$

Supplement (Convolution and FFT)

$y = \overline{K} * u$ Time domain

(FFT)

$y = FFT(\overline{K})FFT(\overline{u})$ Freq domain

(IFFT)

$y = \overline{K} * u$ Time domain

```python
def causal_convolution(u, K, nofft=False):
    if nofft:
        # 不使用FFT
        return np.dot(K[::-1], np.transpose(u))
    else:
        # 使用FFT
        assert K.shape[0] == u.shape[0]
        ud = np.fft.rfft(np.pad(u, (0, K.shape[0])))
        Kd = np.fft.rfft(np.pad(K, (0, u.shape[0])))
        out = ud * Kd
        return np.fft.irfft(out)[u.shape[0] - 1]
```

Computational Challenge of S4:

$$x_0 = \overline{\boldsymbol{B}} u_0 \qquad x_1 = \overline{\boldsymbol{A}}\overline{\boldsymbol{B}} u_0 + \overline{\boldsymbol{B}} u_1 \qquad x_2 = \overline{\boldsymbol{A}}^2 \overline{\boldsymbol{B}} u_0 + \overline{\boldsymbol{A}}\overline{\boldsymbol{B}} u_1 + \overline{\boldsymbol{B}} u_2 \qquad \dots$$

$$y_0 = \overline{\boldsymbol{C}}\overline{\boldsymbol{B}} u_0 \qquad y_1 = \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}\overline{\boldsymbol{B}} u_0 + \overline{\boldsymbol{C}}\overline{\boldsymbol{B}} u_1 \qquad y_2 = \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^2 \overline{\boldsymbol{B}} u_0 + \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}\overline{\boldsymbol{B}} u_1 + \overline{\boldsymbol{C}}\overline{\boldsymbol{B}} u_2 \qquad \dots$$

$$y_k = \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^k \overline{\boldsymbol{B}} u_0 + \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^{k-1} \overline{\boldsymbol{B}} u_1 + \dots + \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}\overline{\boldsymbol{B}} u_{k-1} + \overline{\boldsymbol{C}}\overline{\boldsymbol{B}} u_k$$

(Naïve and Unstable)

Pipeline for Computational Challenge

Define:

(1)Normal Plus Low Rank (NPLR): Matrix A (Hippo) can be separated a Normal Matrix Plus low rank matrix.

(2)Diagonal Plus Low Rank (DPLR): Matrix A (Hippo) can be separated a Diagnal Matrix Plus low rank matrix.

(3)Unitarily Equivalent: SSM(A, B, C) ~ SSM($\Lambda - PQ^*$, B, C)

(4)Cauchy Kernel : https://baike.baidu.com/item/柯西核/22933987

Pipeline：

（1） Use Truncated Generation Function for calculating Kernel

（2） the diagonal matrix case is equivalent to the computation of a Cauchy Kernel

（3） Employ DPLR and Woodbury Identity to correct the result

SSM Generating Functions

Generation function: $f(x) = \sum a_i k_i(x)$; Specifically, in S4,

$$f(x) = \sum a_i x^i, i \geq 0$$ *(Simliar to Z-transform)*

$$K(x) = \sum \bar{C}\bar{A}^i\bar{B}x^{L-i}$$

Origin version:
Use FFT to get $y_1(x_0)y_2(x_0)$ quickly.
E.g., we have two polynomial function $y_1(x)$ and $y_2(x)$, and we want to get $y_1(x_0)y_2(x_0)$

Naïve method:
Step 1, calculate $y_1(x)$*$y_2(x)$
Step 2, get $\mathrm{x} = \mathrm{x}_0$, and calculate $y_1(x_0)$* $y_2(x_0)$
Time complexity: O(n^2)

SSM Generating Functions

FFT method:

$$Y = y_1(x) * y_2(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{2n} x^{2n}$$

$$\begin{bmatrix} Y[0] \\ Y[1] \\ . \\ . \\ . \\ Y[2n] \end{bmatrix} = M(x) \begin{bmatrix} a_0 \\ a_1 \\ . \\ . \\ . \\ a_{2n} \end{bmatrix}$$

How to Choose M(x)
Intuition 1: (From FFT)

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i\frac{2\pi}{N}kn}$$

Similari to

$$Y_k = \sum_{n=0}^{N-1} a_n e^{-i\frac{2\pi}{N}kn}$$

Also, $e^{-i\frac{2\pi}{N}k}$ $(k \leq 0)$ is the root of a unit circle in complex domain.[这里也可以看成是一个负数边上的z-transform]
Intuition 2: ***Cooley-Tukey算法***

SSM Generating Functions

According to FFT, we can get:

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ \vdots \\ X_{N-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{N-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{N-1} & \alpha^{2(N-1)} & \cdots & \alpha^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{bmatrix}$$

X means value Y and x means coefficient a.
Obviously, the Matrix M here is a **Vandermonde matrix**.

In Generating Functions problems, we should have

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ \vdots \\ X_{N-1} \end{bmatrix}$$

and

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{N-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{N-1} & \alpha^{2(N-1)} & \cdots & \alpha^{(N-1)(N-1)} \end{bmatrix}$$

Like $\mathrm{Y} = \mathrm{M}X \rightarrow M^{-1}Y = X$

SSM Generating Functions

This problem equalt to two subproblem:
(1) How to get Y

$$\hat{\mathcal{K}}_L(z) = \sum_{i=0}^{L-1} \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^i\overline{\boldsymbol{B}}z^i = \overline{\boldsymbol{C}}(\boldsymbol{I} - \overline{\boldsymbol{A}}^L z^L)(\boldsymbol{I} - \overline{\boldsymbol{A}}z)^{-1}\overline{\boldsymbol{B}} = \widetilde{\boldsymbol{C}}(\boldsymbol{I} - \overline{\boldsymbol{A}}z)^{-1}\overline{\boldsymbol{B}}$$

Where $\mathrm{z}^L = 1$ because z is a root of a unit circle.

(2) How to calculate the inverse of a **Vandermonde matrix**.

$$M^{-1} = \frac{M^*}{L}$$

Reference: https://www.cnblogs.com/gzy-cjoier/p/9741950.html

Have A Rest

What we have already known:

$$\hat{\mathcal{K}}_L(z) = \sum_{i=0}^{L-1} \overline{\boldsymbol{C}}\,\overline{\boldsymbol{A}}^i \overline{\boldsymbol{B}} z^i = \overline{\boldsymbol{C}}(\boldsymbol{I} - \overline{\boldsymbol{A}}^L z^L)(\boldsymbol{I} - \overline{\boldsymbol{A}} z)^{-1} \overline{\boldsymbol{B}} = \widetilde{\boldsymbol{C}}(\boldsymbol{I} - \overline{\boldsymbol{A}} z)^{-1} \overline{\boldsymbol{B}}$$

However, we can still simplify the inverse process:

$$(I - \bar{A}\,z)^{-1}$$

For the next two steps, we will simplify

$$\bar{A} = \Lambda + PQ^*$$

Then, we can employ Woodbury identity:

$$(\boldsymbol{\Lambda} + \boldsymbol{P}\boldsymbol{Q}^*)^{-1} = \boldsymbol{\Lambda}^{-1} - \boldsymbol{\Lambda}^{-1}\boldsymbol{P}(1 + \boldsymbol{Q}^*\boldsymbol{\Lambda}^{-1}\boldsymbol{P})^{-1}\boldsymbol{Q}^*\boldsymbol{\Lambda}^{-1}$$

to make this calculation easier and efficient.

Chapter

- 1. Review Easy Mode
- 2. SSM Truncated Generation Functions
- 3. Diagonal Cases and DPLR
- 4. Code Explanation for S4

Definition Explanation （Unitarily Equivalent）

Unitarily Equivalent :

(Lemma 3.1 in S4 Paper): conjugation is an equivalence relation on

$$SSMs(A,B,C) \sim (V^{-1}AV, V^{-1}B, CV)$$

Wirte out the two SSMs with state denoted by x and $\tilde{x}$ respectively:

$$\begin{cases} x' = Ax + Bu \\ y = Cx' \end{cases} \qquad \begin{cases} \tilde{x}' = V^{-1}AV\tilde{x} + V^{-1}Bu \\ y = CV\tilde{x}' \end{cases}$$

If we set $x = \mathrm{V}\tilde{x}$ and after multiplying the right side SSM by V, the two SSMs can become identical. Therefore these compute the exact same operator u -> y, but with a change of basis by V in the state x

Explanation

$$\begin{aligned} y &= Cx' \\ &= C(Ax + Bu) \\ &= CVV^{-1}(AV\tilde{x} + Bu) \\ &= CV(V^{-1}AV\tilde{x} + V^{-1}Bu) \end{aligned} \quad \Longrightarrow \quad \begin{cases} \tilde{x}' = V^{-1}AV\tilde{x} + V^{-1}Bu \\ y = CV\tilde{x}' \end{cases}$$

Definition Explanation（NPLR）

Normal Plus Low Rank:

For a Hippo Matrix A (Hippo-LegS):

$$A_{nk} = -\begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & if\ n > k \\ n+1 & if\ n = k\ , \\ 0 & if\ n < k \end{cases} \qquad B_n = (2n+1)^{\frac{1}{2}}$$

Normal Matrix (Complex):

For a normal matrix A, it satisfies

$$AA^* = A^*A,$$

where A^* is the conjugate transpose of A.

Why Normal Matrix:

A normal matrix A satisfy:

$$A = U\Lambda U^*,$$

Where Λ is a diagnoal matrix.

Definition Explanation （NPLR）

Convert Matrix A (Hippo-LegS) to NPLR matrix

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

(1) Let A plus $\frac{1}{2}(2n+1)^{\frac{1}{2}}(2k+1)^{\frac{1}{2}}(we\ define\ it\ as\ M_{nk})$ and we can get B =

$$-\begin{cases} \frac{1}{2}(2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k \\ \frac{1}{2} & \text{if } n = k \\ -\frac{1}{2}(2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n < k \end{cases}$$

Definition Explanation （NPLR -> DPLR）

(2) Let $\mathrm{B} = -\frac{1}{2}\mathrm{I} + \mathrm{S}$, where S is a skew − symmetric matrix

$$-\begin{cases} \frac{1}{2}(2n+1)^{\frac{1}{2}}(2k+1)^{\frac{1}{2}} \; if\ n > k \\ 0, \quad if\ n = k \\ -\frac{1}{2}(2n+1)^{\frac{1}{2}}(2k+1)^{\frac{1}{2}}, \quad if\ n < k \end{cases}$$

An example of a skew-symmetric matrix

$$\begin{bmatrix} 0 & 1 & -2 \\ -1 & 0 & -4 \\ 2 & 4 & 0 \end{bmatrix}$$

(3) Summary and Conclusion:

Step 1: $A_{nk} + M_{nk} = -\frac{1}{2}I + S$ (tips 1: A Normal Matrix plus c Identity Matrix is still a Normal Matrix)

Step 2: $A_{nk} = M_{nk} + Normal$

Step 3: $A_{nk} = -PQ^T + Normal$

Where $P = -(2n+1)^{\frac{1}{2}}, Q = (2k+1)^{\frac{1}{2}}; n, k \in [1, N]$

Step 4: $Normal = V\Lambda V^*$

Step 5: $A_{nk} = V\Lambda V^* - PQ^T$

Step 6: $A_{nk} = V(\Lambda - (V^*P)(V^*Q)^*)V^* \sim \Lambda - (V^*P)(V^*Q)^*$

$$SSMs(A, B, C) \sim (V^{-1}AV, V^{-1}B, CV)$$

Diagonal Cases （Conclusion）

$$\hat{\boldsymbol{K}}(z) = \frac{2}{1+z}\left[\tilde{\boldsymbol{C}}^*\boldsymbol{R}(z)\boldsymbol{B} - \tilde{\boldsymbol{C}}^*\boldsymbol{R}(z)\boldsymbol{P}\left(1+\boldsymbol{Q}^*\boldsymbol{R}(z)\boldsymbol{P}\right)^{-1}\boldsymbol{Q}^*\boldsymbol{R}(z)\boldsymbol{B}\right]$$

$$\tilde{\boldsymbol{C}} = (\boldsymbol{I} - \overline{\boldsymbol{A}}^L)^*\boldsymbol{C}$$

$$\boldsymbol{R}(z;\boldsymbol{\Lambda}) = \left(\frac{2}{\Delta}\frac{1-z}{1+z} - \boldsymbol{\Lambda}\right)^{-1}.$$

Woodbury Identity

$$(\boldsymbol{\Lambda} + \boldsymbol{P}\boldsymbol{Q}^*)^{-1} = \boldsymbol{\Lambda}^{-1} - \boldsymbol{\Lambda}^{-1}\boldsymbol{P}(1+\boldsymbol{Q}^*\boldsymbol{\Lambda}^{-1}\boldsymbol{P})^{-1}\boldsymbol{Q}^*\boldsymbol{\Lambda}^{-1}$$

$$\begin{aligned}
\tilde{\boldsymbol{C}}^*\left(\boldsymbol{I} - \overline{\boldsymbol{A}}z\right)^{-1}\overline{\boldsymbol{B}} &= \frac{2}{1+z}\tilde{\boldsymbol{C}}^*\left(\frac{2}{\Delta}\frac{1-z}{1+z} - \boldsymbol{A}\right)^{-1}\boldsymbol{B} \\
&= \frac{2}{1+z}\tilde{\boldsymbol{C}}^*\left(\frac{2}{\Delta}\frac{1-z}{1+z} - \boldsymbol{\Lambda} + \boldsymbol{P}\boldsymbol{Q}^*\right)^{-1}\boldsymbol{B} \\
&= \frac{2}{1+z}\left[\tilde{\boldsymbol{C}}^*\boldsymbol{R}(z)\boldsymbol{B} - \tilde{\boldsymbol{C}}^*\boldsymbol{R}(z)\boldsymbol{P}\left(1+\boldsymbol{Q}^*\boldsymbol{R}(z)\boldsymbol{P}\right)^{-1}\boldsymbol{Q}^*\boldsymbol{R}(z)\boldsymbol{B}\right].
\end{aligned}$$

Diagonal Cases

Cachy-Kernel:

$$K_R(\tilde{C}, B) = \sum_i \frac{\widetilde{C_i}^* B_i}{R(z;\Lambda)}$$

$$\boldsymbol{R}(z;\boldsymbol{\Lambda}) = \left(\frac{2}{\Delta}\frac{1-z}{1+z} - \boldsymbol{\Lambda}\right)^{-1}.$$

$$\begin{aligned}
\tilde{\boldsymbol{C}}^*\left(\boldsymbol{I} - \overline{\boldsymbol{A}}z\right)^{-1}\overline{\boldsymbol{B}} &= \frac{2}{1+z}\tilde{\boldsymbol{C}}^*\left(\frac{2}{\Delta}\frac{1-z}{1+z} - \boldsymbol{A}\right)^{-1}\boldsymbol{B} \\
&= \frac{2}{1+z}\tilde{\boldsymbol{C}}^*\left(\frac{2}{\Delta}\frac{1-z}{1+z} - \boldsymbol{\Lambda} + \boldsymbol{P}\boldsymbol{Q}^*\right)^{-1}\boldsymbol{B} \\
&= \frac{2}{1+z}\left[\underbrace{\tilde{\boldsymbol{C}}^*\boldsymbol{R}(z)\boldsymbol{B}}_{K_R(\tilde{C},B)} - \underbrace{\tilde{\boldsymbol{C}}^*\boldsymbol{R}(z)\boldsymbol{P}}_{K_R(\tilde{C},P)}\left(1 + \underbrace{\boldsymbol{Q}^*\boldsymbol{R}(z)\boldsymbol{P}}_{K_R(Q^*,P)}\right)^{-1}\underbrace{\boldsymbol{Q}^*\boldsymbol{R}(z)\boldsymbol{B}}_{K_R(Q^*,B)}\right].
\end{aligned}$$

Pipeline of S4

Algorithm 1 S4 CONVOLUTION KERNEL (SKETCH)

Input: S4 parameters $\boldsymbol{\Lambda}, \boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{B}, \boldsymbol{C} \in \mathbb{C}^N$ and step size Δ

Output: SSM convolution kernel $\overline{\boldsymbol{K}} = \mathcal{K}_L(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \overline{\boldsymbol{C}})$ for $\boldsymbol{A} = \boldsymbol{\Lambda} - \boldsymbol{P}\boldsymbol{Q}^*$ (equation (5))

1: $\widetilde{\boldsymbol{C}} \leftarrow \left(\boldsymbol{I} - \overline{\boldsymbol{A}}^L\right)^* \overline{\boldsymbol{C}}$ ▷ Truncate SSM generating function (SSMGF) to length L

2: $\begin{bmatrix} k_{00}(\omega) & k_{01}(\omega) \\ k_{10}(\omega) & k_{11}(\omega) \end{bmatrix} \leftarrow \left[\widetilde{\boldsymbol{C}}\ \boldsymbol{Q}\right]^* \left(\frac{2}{\Delta}\frac{1-\omega}{1+\omega} - \boldsymbol{\Lambda}\right)^{-1} [\boldsymbol{B}\ \boldsymbol{P}]$ ▷ Black-box Cauchy kernel

3: $\hat{\boldsymbol{K}}(\omega) \leftarrow \frac{2}{1+\omega}\left[k_{00}(\omega) - k_{01}(\omega)(1 + k_{11}(\omega))^{-1}k_{10}(\omega)\right]$ ▷ Woodbury Identity

4: $\hat{\boldsymbol{K}} = \{\hat{\boldsymbol{K}}(\omega) : \omega = \exp(2\pi i \frac{k}{L})\}$ ▷ Evaluate SSMGF at all roots of unity $\omega \in \Omega_L$

5: $\overline{\boldsymbol{K}} \leftarrow \mathsf{iFFT}(\hat{\boldsymbol{K}})$ ▷ Inverse Fourier Transform

$$
\begin{aligned}
\tilde{\boldsymbol{C}}^* \left(\boldsymbol{I} - \overline{\boldsymbol{A}}z\right)^{-1} \overline{\boldsymbol{B}} &= \frac{2}{1+z}\tilde{\boldsymbol{C}}^* \left(\frac{2}{\Delta}\frac{1-z}{1+z} - \boldsymbol{A}\right)^{-1} \boldsymbol{B} \\
&= \frac{2}{1+z}\tilde{\boldsymbol{C}}^* \left(\frac{2}{\Delta}\frac{1-z}{1+z} - \boldsymbol{\Lambda} + \boldsymbol{P}\boldsymbol{Q}^*\right)^{-1} \boldsymbol{B} \\
&= \frac{2}{1+z}\left[\tilde{\boldsymbol{C}}^*\boldsymbol{R}(z)\boldsymbol{B} - \tilde{\boldsymbol{C}}^*\boldsymbol{R}(z)\boldsymbol{P}\left(1 + \boldsymbol{Q}^*\boldsymbol{R}(z)\boldsymbol{P}\right)^{-1}\boldsymbol{Q}^*\boldsymbol{R}(z)\boldsymbol{B}\right].
\end{aligned}
$$

$K_R(\tilde{C}, B)$ $K_R(\tilde{C}, P)$ $K_R(Q^*, P)$ $K_R(Q^*, B)$

Code and Experiment

1. $\bar{A} = \Lambda + PQ^* \rightarrow \Lambda - PP^* (P = Q)$
2.

$$\boldsymbol{A} = -\begin{bmatrix} \frac{1}{2}+\beta & & & & \cdots \\ 1 & \frac{1}{2}+\beta & & & \\ 1 & 1 & \frac{1}{2}+\beta & & \\ 1 & 1 & 1 & \frac{1}{2}+\beta & \\ \vdots & & & & \ddots \end{bmatrix} = -\beta I - \begin{bmatrix} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \\ \frac{1}{2} & & -\frac{1}{2} & -\frac{1}{2} & \cdots \\ \frac{1}{2} & \frac{1}{2} & & -\frac{1}{2} & \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & & \\ \vdots & & & & \ddots \end{bmatrix} - \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots \\ 1 & 1 & 1 & 1 & \\ 1 & 1 & 1 & 1 & \\ 1 & 1 & 1 & 1 & \\ \vdots & & & & \ddots \end{bmatrix}.$$

The first term is skew-symmetric, which is unitarily similar to a (complex) diagonal matrix with pure imaginary eigenvalues (i.e., real part 0). The second matrix can be factored as pp^* for $p = 2^{-1/2}\begin{bmatrix}1 & \cdots & 1\end{bmatrix}^*$. Thus the whole matrix A is unitarily similar to a matrix $\Lambda - pp^*$ where the eigenvalues of Λ have real part between $-\frac{1}{2}$ and 0.

Ref: https://arxiv.org/abs/2202.09729

Code and Experiment

3. Hermitian Matrix: $A = A^*$
(1) $A = V^{-1}\Lambda\mathrm{V}$, for V, it composes a set of orthogonal basis
Ref: https://en.wikipedia.org/wiki/Hermitian_matrix

Lemma: for a Hermitian matrix A, if it satisfy:

$$\begin{cases} A = (B + kI) \\ A = V^{-1}\Lambda\mathrm{V} \end{cases} \rightarrow \mathrm{B} = V^{-1}\Lambda'\mathrm{V}$$

Proof:

$$\mathrm{V}(B + kI)\mathrm{V}^{-1} = \mathrm{VBV}^{-1} + kI = \Lambda$$
$$\mathrm{VBV}^{-1} = \Lambda' \rightarrow \mathrm{B} = V^{-1}\Lambda'\mathrm{V}$$

Where

$$\Lambda' = \Lambda - kI$$