

# Alternative Models

## Discretized State Space Machines

ELL8299 · ELL881 · AIL861



**Sourish Dasgupta**

Associate Professor, DAU, Gandhinagar  
<https://daiict.ac.in/faculty/sourish-dasgupta>

# State Space Machines – *Language as a Diffusive Field*

**Core idea.** Instead of updating memory in discrete jumps (as in RWKV or LSTM), a State Space Model (SSM) treats hidden meaning as a *continuously evolving field*:

$$\frac{dx}{dt} = Ax(t) + Bu(t).$$

- $x(t)$  — the latent semantic field (what the model “feels” at any instant)
- $Ax(t)$  — how that field drifts or decays on its own (internal physics)
- $Bu(t)$  — how the current token *nudges* or perturbs the field



# State Space Machines – *Language as a Diffusive Field*

**Model:**  $\dot{x}(t) = Ax(t) + Bu(t), \quad y(t) = Cx(t) + Du(t)$

## Assumptions used for the derivation

- $A, B, C, D$  are time-invariant (constant) matrices.
- $u(t)$  is piecewise continuous and bounded on finite intervals.
- Initial state  $x(t)$  is finite for finite  $t$ .

**Goal.** Express  $x(t + \Delta)$  in terms of  $x(t)$  and  $u(\cdot)$ , then obtain a per-token update later.



## Solving the Continuous SSM ODE ...

$$x(t + \Delta) = e^{A\Delta}x(t) + \int_0^{\Delta} e^{A\tau} B u(t + \Delta - \tau) d\tau.$$

*Meaning.*  $e^{A\Delta}x(t)$  = drift of existing memory; the integral = accumulated input effect over the interval.



## Solution Step 1:

Original ODE mixes  $x$  and  $\dot{x}$ :  $\dot{x} = Ax + Bu$  (hard to integrate directly).

$$\dot{x}(t) = Ax(t) + Bu(t), \quad M(t) = e^{-At}, \quad \dot{M}(t) = -AM(t).$$

$$\frac{d}{dt}(Mx) = \dot{M}x + M\dot{x} = (-AM)x + M(Ax + Bu) = \underbrace{(-AM + MA)}_{=0}x + MBu,$$

where  $M$

$$\frac{d}{dt}(e^{-At}x(t)) = e^{-At}Bu(t).$$

Now the left side is a *total derivative* — directly integrable.



## Solution Step 2: Integrating ...

$$e^{-A(t+\Delta)}x(t+\Delta) - e^{-At}x(t) = \int_t^{t+\Delta} e^{-As}B u(s) ds.$$

Left-multiply by  $e^{A(t+\Delta)}$  and use  $e^{A(t+\Delta)}e^{-At} = e^{A\Delta}$ :

$$x(t+\Delta) = e^{A\Delta}x(t) + \int_t^{t+\Delta} e^{A((t+\Delta)-s)}B u(s) ds.$$



## Solution Step 3: The Duhamel Form ...

Change variable  $\tau = (t + \Delta) - s \Rightarrow s = t + \Delta - \tau, ds = -d\tau$ :

$$\begin{aligned}\int_t^{t+\Delta} e^{A((t+\Delta)-s)} B u(s) ds &= \int_{\Delta}^0 e^{A\tau} B u(t + \Delta - \tau) (-d\tau); \\ &= \int_0^{\Delta} e^{A\tau} B u(t + \Delta - \tau) d\tau.\end{aligned}$$

**Duhamel's formula (continuous-time closed form):**

$$x(t + \Delta) = e^{A\Delta} x(t) + \int_0^{\Delta} e^{A\tau} B u(t + \Delta - \tau) d\tau.$$



# Interpreting the structure of $A$

Let  $A = V\Lambda V^{-1}$ , with  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ .

Quantity	Mathematical meaning	Linguistic / intuitive meaning
Eigenvalue $\lambda_i$	Rate and frequency of decay/oscillation for mode $i$ .	Memory lifetime and rhythm of a semantic feature.
Eigenvector $v_i$	Direction in latent space along which that mode acts.	What type of information (topic/phrase/connector) is being remembered.

**Table: Eigenstructure as memory timescales and semantic channels.**



# Interpreting the Eigenvectors

Let  $\lambda_j = \alpha_j + i\beta_j$ . Then:

$$e^{\lambda_j t} = e^{\alpha_j t} (\cos(\beta_j t) + i \sin(\beta_j t)).$$

**Together:** each eigenvalue encodes *how fast* and *how rhythmically* meaning evolves.

**Linguistic intuition.**

- $\alpha_j < 0 \rightarrow$  memory decays: “Mary” eventually fades.
- $\alpha_j = 0 \rightarrow$  memory persists: “John” stays in focus across the sentence.
- $\beta_j \neq 0 \rightarrow$  rhythm/recurrence: subject–verb–object or syntactic cycles reappear.



# Well $A$ is **time-invariant**. Then what evolves?

Object	Behavior over time (inference)	Why
Eigenvalues $\lambda_i$ of $A$	<b>Do not change.</b> Fixed rates & frequencies.	$A$ is time-invariant.
Mode factor $e^{\lambda_i t}$ or $\bar{\lambda}_i^k$	<b>Changes with <math>t</math> or <math>k</math>.</b>	Time appears in the exponential; this drives decay/oscillation.
Modal coefficients $\alpha_i(t), \beta_i(t)$	<b>Change with inputs.</b>	New inputs enter via $B$ and accumulate into each mode.
Output contribution $(Cq_i) \alpha_i(t) e^{\lambda_i t}$	<b>Changes.</b>	Both coefficient and exponential evolve; $C$ reweights at readout.
Selective SSM (e.g., Mamba)	<i>Effective <math>A_t</math> may vary</i> $\Rightarrow$ apparent mode shifts.	Input-dependent gating modulates dynamics.

**Table: State/mode evolution vs. eigenvalue constancy** in time-invariant SSMs.



# How the state evolves?

$$\dot{x}(t) = Ax(t) + Bu(t).$$

If  $A$  is diagonalizable,  $A = Q\Lambda Q^{-1}$  with  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ .

Define modal coordinates:

## Interpretation

- Change of basis separates memory into **independent modes**.
- Each eigenvalue  $\lambda_i$  defines a distinct time scale (decay/oscillation).
- Inputs enter via  $B$  and are projected into these modal directions through  $Q^{-1}B$ .



# Dynamics of each *Mode*

For mode  $i$ , extract the  $i$ th component:

## Meaning

- $e^{\lambda_i(t-t_0)}$  — how existing memory decays/oscillates.
- $b_i^\top u(\tau)$  — how new input excites mode  $i$ .
- Each mode acts as a **leaky resonator** driven by  $u(t)$ .



# Accumulation of the dynamics over time

Unrolling the recurrence:

$t-1$

**State reconstruction:**

$$x_t = Q z_t, \quad y_t = Cx_t = \sum_i (Cq_i) z_{i,t}.$$

Interp

- $e^{\lambda_i(t-t_0)}$  — fading echo from past tokens.
- $\bar{b}_i^\top u_k$  — how strongly token  $u_k$  excites mode  $i$ .
- Each eigenmode is a **leaky integrator**: older tokens contribute weaker signals.



# Modal Memory – Linguistic Interpretation

Mathematical object	Linguistic meaning
$q_i$ (eigenvector)	Semantic axis — “topic”, “phrase rhythm”, or “connector direction”.
$\lambda_i$ (eigenvalue)	How fast / how rhythmically that semantic trace fades or oscillates.
$b_i^\top u_t$	How much the current word injects energy into that semantic axis.
$z_{i,t}$	Current strength of that memory mode.
$e^{\lambda_i \Delta}$	Memory persistence (decay or recurrence rate).

**Table:** Each input word updates semantic “resonators” (modes) through  $B$  and  $\lambda_j$ .



# So why discretization is needed?

## Language Tokens are Discrete

Example: “John loves Mary who lives in New York City.”

Tokens: [John, loves, Mary, who, lives, in, New, York, City]

- Continuous-time SSM expects smooth input flow.
- Discretization samples the dynamics at each token arrival.
- Converts differential form into step-wise recurrence:

$$x_{t+1} = \bar{A}x_t + \bar{B}u_t$$

- $\bar{A} = e^{A\Delta t}$  preserves semantic drift and oscillation.



# Zero-Order Hold (ZOH) – The discretization trick

**Tokenized inputs.** On each interval  $[t, t + \Delta)$ , approximate  $u(\tau)$  as constant:

$$u(\tau) \approx u_t \quad (\text{ZOH}).$$

Insert into Duhamel's formula:

$$x(t + \Delta) = e^{A\Delta}x(t) + \left( \int_0^\Delta e^{A\tau} B d\tau \right) u_t.$$

**Define discrete parameters**

$$\bar{A} = e^{A\Delta}, \quad \bar{B} = \int_0^\Delta e^{A\tau} B d\tau.$$

**Discrete state update (one token step)**

$$x_{t+1} = \bar{A}x_t + \bar{B}u_t.$$

*Note:*  $B$  is constant  $\Rightarrow \left( \int_0^\Delta e^{A\tau} d\tau \right) B = \int_0^\Delta e^{A\tau} B d\tau$ .



# From Flow to Token – Discrete updates (under ZOH)

Discretize with step  $\Delta$ :

$$\bar{A} = e^{A\Delta}, \quad \bar{B} = \int_0^\Delta e^{A\tau} B d\tau.$$

In modal basis:

$$z_{t+1} = e^{\Lambda\Delta} z_t + Q^{-1} \bar{B} u_t.$$

Component-wise:

$$z_{i,t+1} = e^{\lambda_i\Delta} z_{i,t} + \bar{b}_i^\top u_t, \quad \bar{b}_i^\top = e_i^\top Q^{-1} \bar{B}.$$

Plain words

Each token injects new energy into all modes through  $\bar{b}_i^\top u_t$ , while the existing amplitude decays/oscillates by  $e^{\lambda_i\Delta}$ .



# Numerical Solution (via Matrix Taylor Series)

For constant  $A$ :

$$\bar{A} = e^{A\Delta} = I + A\Delta + \frac{A^2\Delta^2}{2!} + \frac{A^3\Delta^3}{3!} + \dots$$

$$\bar{B} = \int_0^\Delta e^{A\tau} B d\tau = \sum_{k=0}^{\infty} \frac{A^k B \Delta^{k+1}}{(k+1)!} = \Delta B + \frac{AB\Delta^2}{2!} + \frac{A^2B\Delta^3}{3!} + \dots$$

*Reading it:*  $\bar{A}$  mixes prior state across multiple time-scales;  $\bar{B}$  accumulates input influence within the step.



# Matrix Taylor Series – Interpretation

**Definition**  $e^{At} = \sum_{k=0}^{\infty} \frac{(At)^k}{k!}$

## Key properties (used next)

①  $\frac{d}{dt} e^{At} = A e^{At}$ , with  $e^{A \cdot 0} = I$ .

② (Semigroup)  $e^{A(t+s)} = e^{At} e^{As}$ .

③  $A$  commutes with  $e^{At}$  because  $e^{At}$  is a power series in  $A$ :  $A e^{At} = e^{At} A$ .

*Interpretation.*  $e^{At}$  advances the state under the homogeneous dynamics  $\dot{x} = Ax$ .



# What exactly is discrete evolution?

**But in language:** tokens arrive at discrete time steps.

$$t = 0, \Delta, 2\Delta, 3\Delta, \dots$$

Each token “samples” this continuous semantic flow.

**Question.** How does this sampling affect the evolution encoded by  $\lambda = a + ib$ ?

*We move from continuous evolution  $e^{\lambda t}$  to its discrete-step version  $e^{\lambda \Delta}$ .*



# Sampling does not affect the semantic dynamics

For each token step  $\Delta$ :

$$\bar{\lambda} = e^{\lambda\Delta} = e^{(a+ib)\Delta} = e^{a\Delta} (\cos(b\Delta) + i \sin(b\Delta)).$$

*eigenvalue's dynamics are preserved in sampled form.*

- $\cos(b\Delta) + i \sin(b\Delta)$  — how much the meaning *rotates* (semantic or syntactic shift).
- $\Delta$  — sampling interval between tokens (semantic drift per word).



# Why not be happy with the discretized recurrence model?

## Discrete update:

$$x_{t+1} = \bar{A}x_t + \bar{B}u_t, \quad y_t = Cx_t + Du_t.$$

(We focus here on the recurrent part  $Cx_t$ ; the  $Du_t$  term adds the current token's direct meaning.)

$$y_t = Cx_t.$$

## Observation.

- Each new token  $u_t$  updates memory through  $\bar{A}x_t$ , linking back to the previous state.
- But recurrence is inherently sequential — we must process one token at a time.
- This hides *how much influence* each earlier token still has.



# Translating *Sequential Memory* into *Convolutional Memory*

We can **unroll the recurrence**:

$$x_t = \sum_{i=0}^{t-1} \bar{A}^{t-1-i} \bar{B} u_i = \sum_{k=1}^t \bar{A}^{k-1} \bar{B} u_{t-k}; \quad y_t = Cx_t = \sum_{k=1}^t \underline{C\bar{A}^{k-1}\bar{B}} u_{t-k}.$$

**Interpretation.**

- Each earlier token  $u_{t-k}$  contributes an “echo” scaled by  $C\bar{A}^{k-1}\bar{B}$ .
- The farther back the token, the smaller its influence (decay through  $\bar{A}^{k-1}$ ).
- This weighted sum is a **convolution**:

$$y_t = \underline{(K * u)}_t, \quad \text{with } \underline{K_k = C\bar{A}^{k-1}\bar{B}}; \quad K = [C\bar{B}, C\bar{A}\bar{B}, C\bar{A}^2\bar{B}, \dots]$$

**Next:** we make this **kernel** explicit to see *how* the model remembers — and forgets.



# Why to translate into the **Kernel** form?

## Mathematically:

$$K_k = C\bar{A}^{k-1}\bar{B}$$

encodes how the effect of a token  $k$  steps ago decays or oscillates before reaching the present.

## Linguistically:

- $K_0$ : direct meaning of the current word (“City”).
- $K_1, K_2$ : short-term echoes (“New”, “York”).
- $K_3-K_6$ : longer semantic ripples (“Mary who lives in”).



# So should we be happy with the kernelized discretization?

From before, recall:

$$y_t = \sum_{k=0}^t K_k u_{t-k}, \quad \text{where } K_k = C\bar{A}^{k-1}\bar{B}.$$

This **kernel convolution** makes memory explicit — but computing it directly is slow:

Cost:  $\mathcal{O}(T^2)$  additions for a  $T$ -token sentence.



A signal/wave perspective: Every word is an “echo” (Time-Domain)

$$y_t = \sum_{i=0}^t K_{t-i} u_i$$

- $u_i$  — embedding (signal) of the  $i$ th word.
- $K_{t-i}$  — how much a word  $i$  steps ago still matters.
- $y_t$  — contextualized meaning at the current step.



A signal/wave “Fourier” perspective: Converting “echo” to “rhythm”

$$u[t] = \sum_{\omega} c_{\omega} e^{i\omega t}$$

- Each  $\omega$  is a **frequency** — a rhythm or tempo in meaning evolution.
- Each  $c_{\omega}$  says how strong that rhythm is.



# Making the quadratic Convolution *fast*

The Fast Fourier Transform (FFT) converts our convolution into a simple multiplication:

$$\underline{(K * u)(t)} \iff \text{IFFT}(\underline{\text{FFT}(K) \odot \text{FFT}(u)})$$

- In the **time domain**: summing echoes word by word (slow).
- In the **frequency domain**: multiplying rhythms (fast).



# The FFT operation – FFT of Input & Kernel

Given discrete sequences (kernel and input):

$$K = [K_0, K_1, \dots, K_{T-1}], \quad u = [u_0, u_1, \dots, u_{T-1}],$$

the convolution  $y = K * u$  can be computed efficiently using FFT in four steps:

**1 FFT of the input:**

$$\tilde{u}[\omega] = \sum_{t=0}^{T-1} u_t e^{-i2\pi\omega t/T}$$

Decomposes  $u$  into frequency components (semantic rhythms).

**2 FFT of the kernel:**

$$\tilde{K}[\omega] = \sum_{t=0}^{T-1} K_t e^{-i2\pi\omega t/T}$$

Encodes how each frequency is amplified or damped by the model.



# The FFT equivalence – Reconstruction of output

## ① Elementwise multiplication:

$$\tilde{y}[\omega] = \tilde{K}[\omega] \tilde{u}[\omega]$$

Performs frequency-wise filtering (apply rhythm-specific weights).

## ② Inverse FFT:

$$y_t = \frac{1}{T} \sum_{\omega} \tilde{y}[\omega] e^{i2\pi\omega t/T}$$

replaces  $\mathcal{O}(T^2)$  convolution with  $\mathcal{O}(T \log T)$  spectral filtering.



Continuing the running example: “*John loves Mary who lives in New York City.*”

FFT Step	Linguistic Interpretation
FFT( $u$ )	Decomposes the sentence into <i>semantic rhythms</i> : slow rhythms capture topics ( <i>John, Mary</i> ), fast rhythms capture short syntactic ripples ( <i>in, who</i> ).
FFT( $K$ )	Encodes the model’s sensitivity to each rhythm: which frequencies to amplify (keep in memory) or damp (forget).
Multiply ( $\odot$ )	Applies the model’s learned <i>frequency filter</i> : how much each semantic rhythm influences later tokens.
IFFT	Reconstructs the sentence from all filtered rhythms: the model

**Analogy:** Rather than replaying every echo word-by-word, FFT lets the model play the whole harmony of context at once.



# The FFT equivalence – Derivation

The discrete convolution:

$$y[n] = \sum_{m=0}^{N-1} K[m] u[(n - m) \pmod{N}].$$

**Take the DFT of both sides:**

$$Y[r] = \sum_{n=0}^{N-1} y[n] e^{-i2\pi rn/N}.$$



# The FFT equivalence – Derivation

Substitute the convolution definition:

$$\begin{aligned} Y[r] &= \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} K[m] u[(n - m) \pmod{N}] e^{-i2\pi rn/N} \\ &= \sum_{m=0}^{N-1} K[m] e^{-i2\pi rm/N} \sum_{p=0}^{N-1} u[p] e^{-i2\pi rp/N} = K[r] U[r]. \end{aligned}$$

**Hence:**

$$\boxed{\text{FFT}(y) = \text{FFT}(K) \odot \text{FFT}(u)}.$$

Each frequency bin  $r$  represents an independent rhythm in the sequence.



# The FFT equivalence – Derivation

The above derivation holds for **circular** convolution. To compute the true (linear) convolution  $y[n] = \sum_m K[m]u[n - m]$ :

- Zero-pad both sequences to length

$$N \geq L_K + L_u - 1.$$

- Apply the same FFT pipeline:

$$y = \text{IFFT}(\text{FFT}(K_{\text{padded}}) \odot \text{FFT}(u_{\text{padded}})).$$

- This avoids wrap-around and reproduces causal semantics.

**Intuition:** Zero-padding ensures no token “wraps around” in time — each token’s influence (its echo) fades forward but never peeks backward.



# Eigenvalues as Hidden Spectrum of Meaning

After discretization:

$$\bar{A} = e^{A\Delta} = Qe^{\Lambda\Delta}Q^{-1}, \quad K_k = C\bar{A}^k\bar{B} = CQe^{k\Lambda\Delta}Q^{-1}\bar{B}.$$

Each eigenvalue  $\lambda_i$  contributes a **frequency component**:

$$K_k^{(i)} \propto e^{\alpha_i k \Delta} e^{i\beta_i k \Delta}.$$



# Eigenvalues as Hidden Spectrum of Meaning

## Connection to FFT:

Concept	Math object	Linguistic meaning
Decay rate	$\text{Re}(\lambda_i)$	Memory persistence (how long “Mary” stays)
Oscillation	$\text{Im}(\lambda_i)$	Rhythmic recurrence (subject–verb cycles)
Semantic axis	$q_i$	Direction of meaning change (entity, relation, syntax)
Spectrum	$K(\omega)$	Observable rhythm pattern of the sentence

Table: Putting It All Together



# Questions?

