Linear Dynamical Systems as a Core Computational Primitive

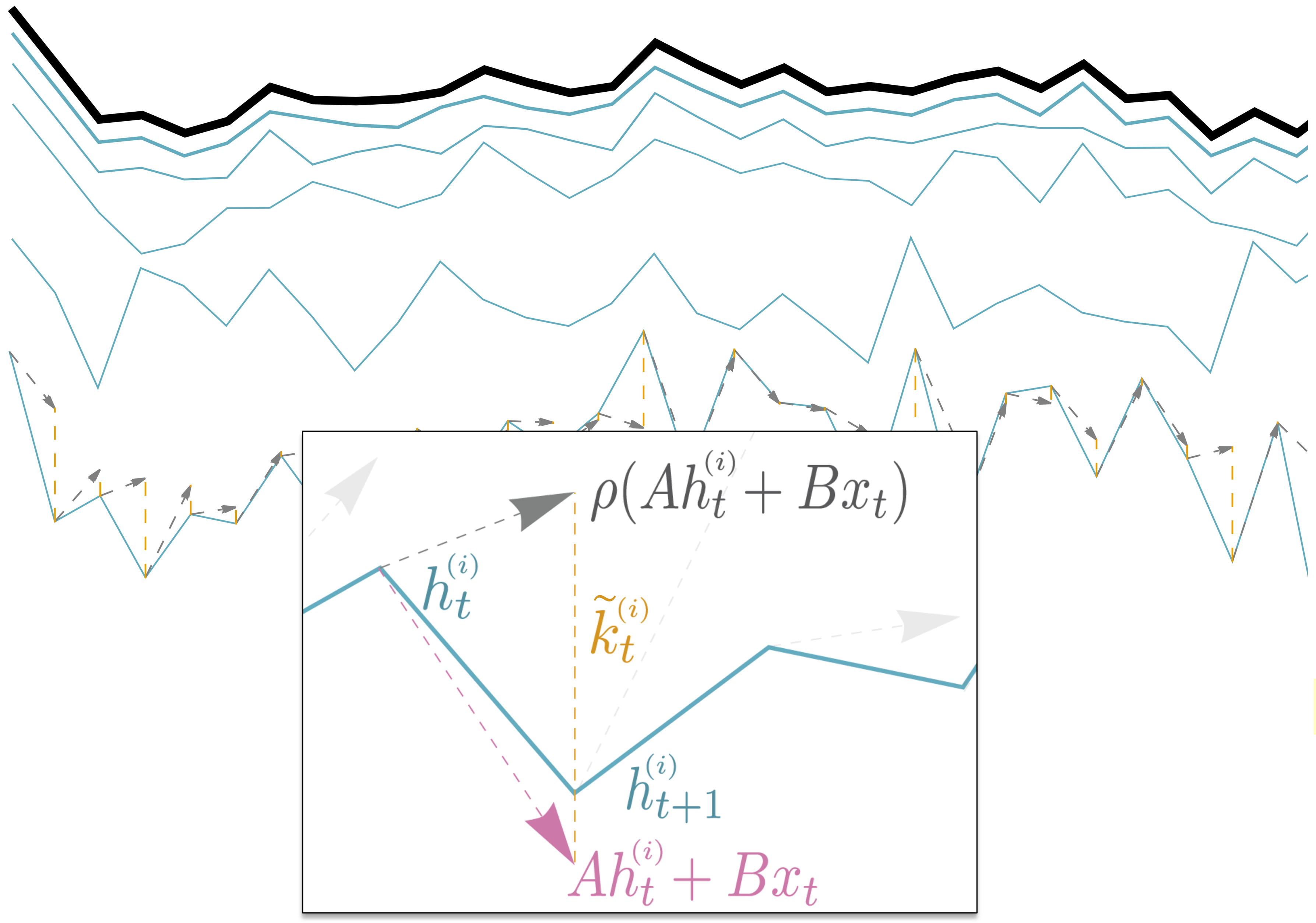
Shiva Kaul <skkaul@cs.cmu.edu>

NEURAL INFORMATION PROCESSING SYSTEMS

$$h_{t+1} = \rho(Ah_t + Bx_t)$$

- Takes *O(T)* time to run for *T* steps
- Not mathematically tractable

0. Nonlinear RNN



$h_{t+1}^{(\Delta)}$ Provably consistent: nonlinear RNN is exactly recovered for large enough $\triangle$

$$\vdots$$

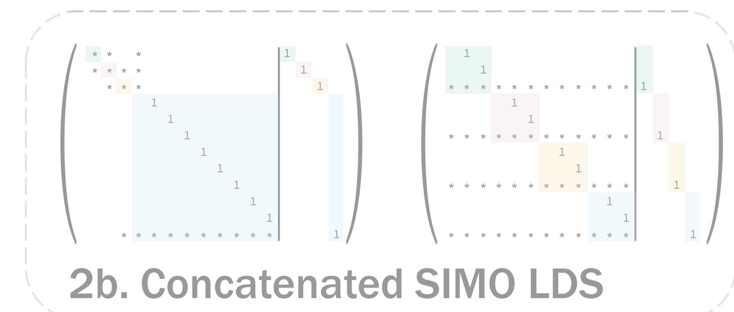
$$h_{t+1}^{(i)} = Ah_t^{(i)} + Bx_t + \tilde{k}_t^{(i-1)} \approx \mathbf{E}\begin{bmatrix}\tilde{s}_{t+1} \\ \tilde{s}_{t+1} \\ \tilde{s}_{t+1}\end{bmatrix}$$

$$\vdots$$

$$h_{t+1}^{(0)} = Ah_t^{(0)} + Bx_t$$

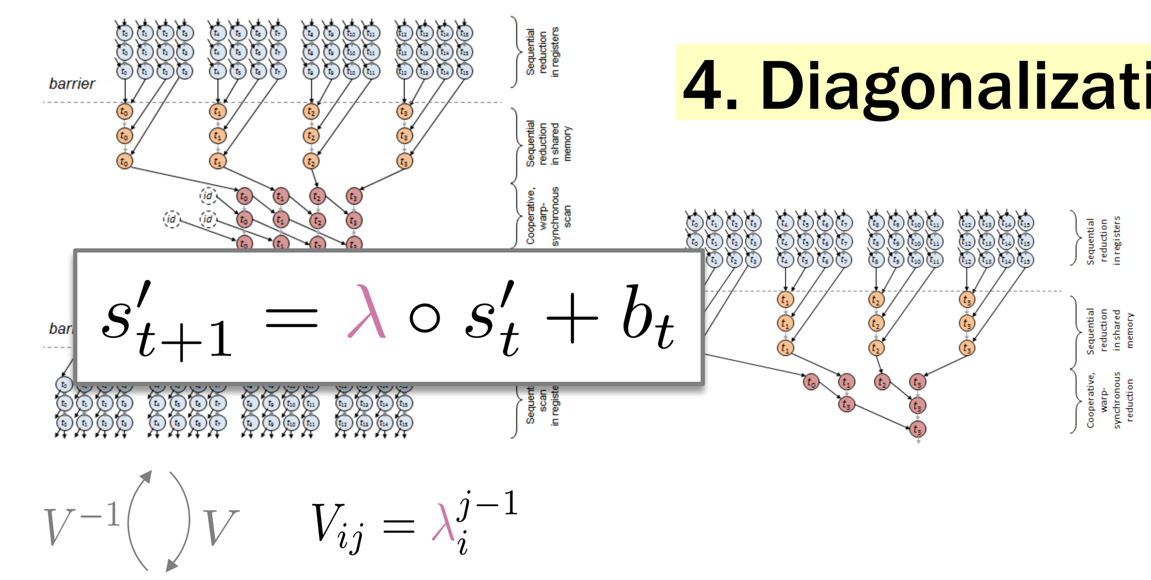
1. Stack of Corrected MIMO LDS

The first layer is a plain LDS. Subsequent layers have an additive corrections $\tilde{k}_t$, which is the deviation between linear and nonlinear steps.



Linear recurrence takes only *O(log T)* parallel time

4. Diagonalization



$$s'_{t+1} = \lambda \circ s'_t + b_t$$

V^{-1} V $\quad V_{ij} = \lambda_i^{j-1}$

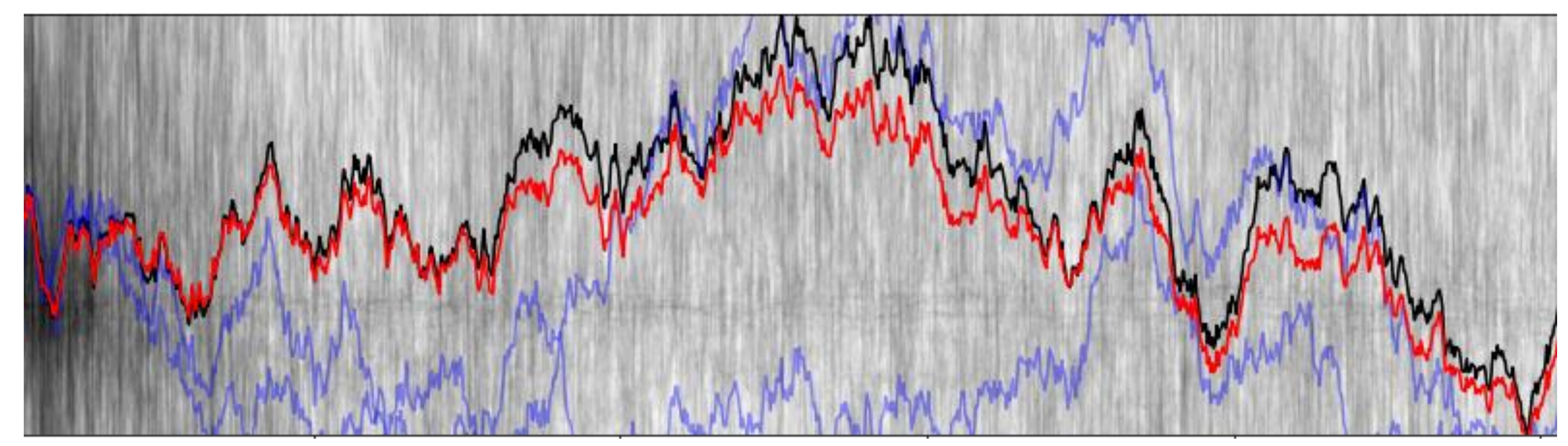
$$s_{t+1} = V^{-1}(\text{diag}\lambda)Vs_t + [1, 0, \ldots]^T x_t + k_t$$

3. Canonicalization

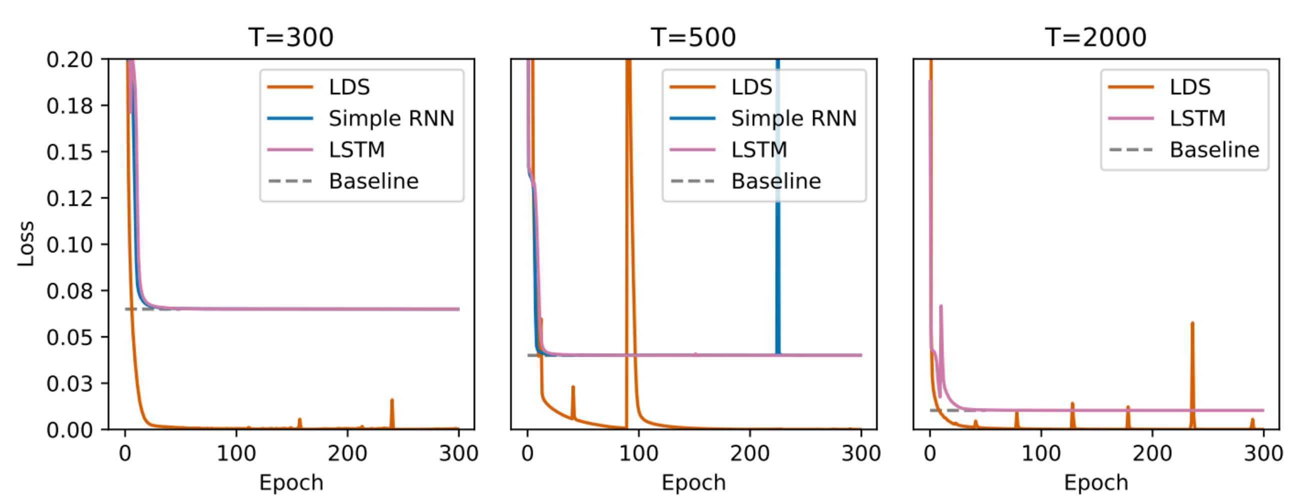
transforms *(A,B)* to structured form

2. Projected SIMO LDS

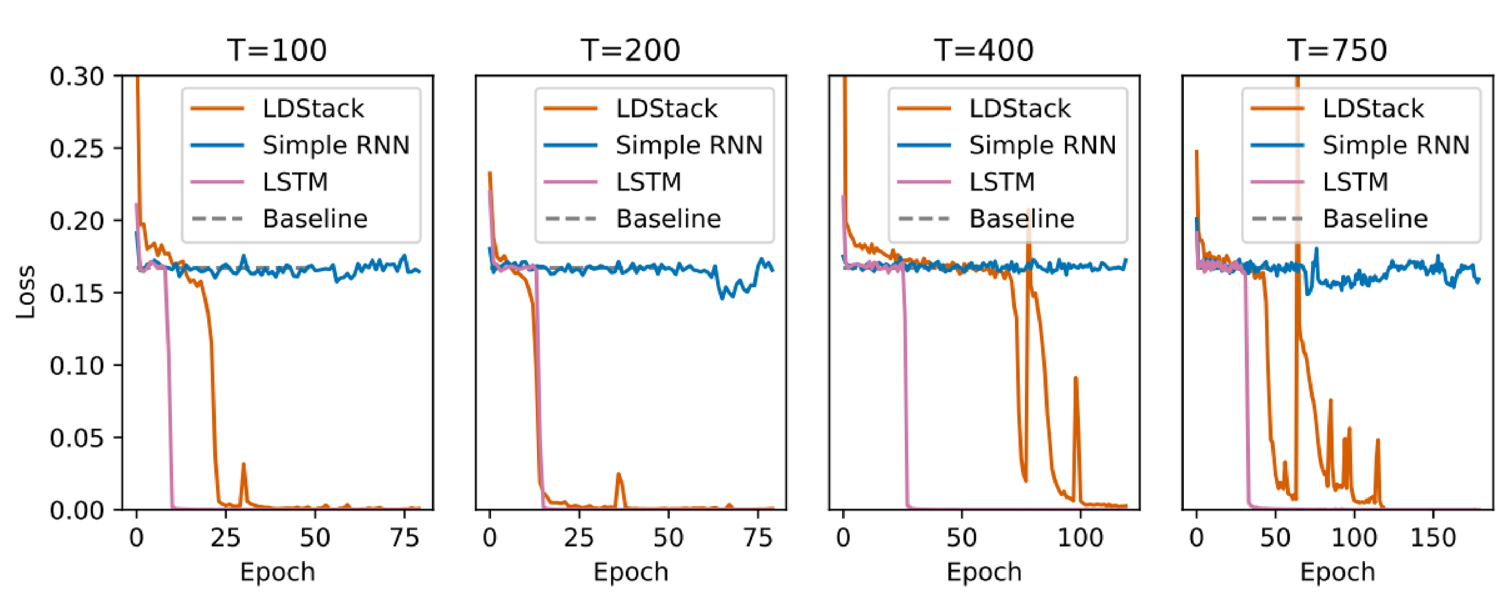
Original, random projections, big and small averages

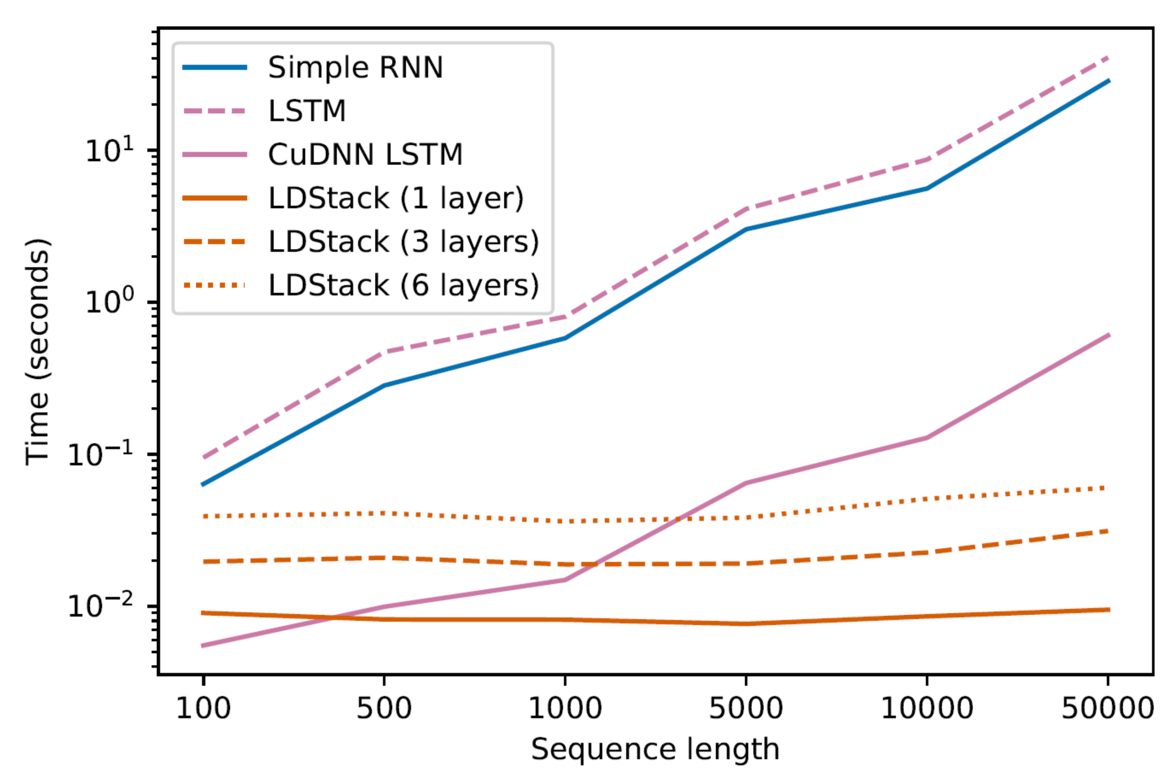


Approximately run nonlinear RNNs in O(log T) parallel time with a mathematically tractable construction

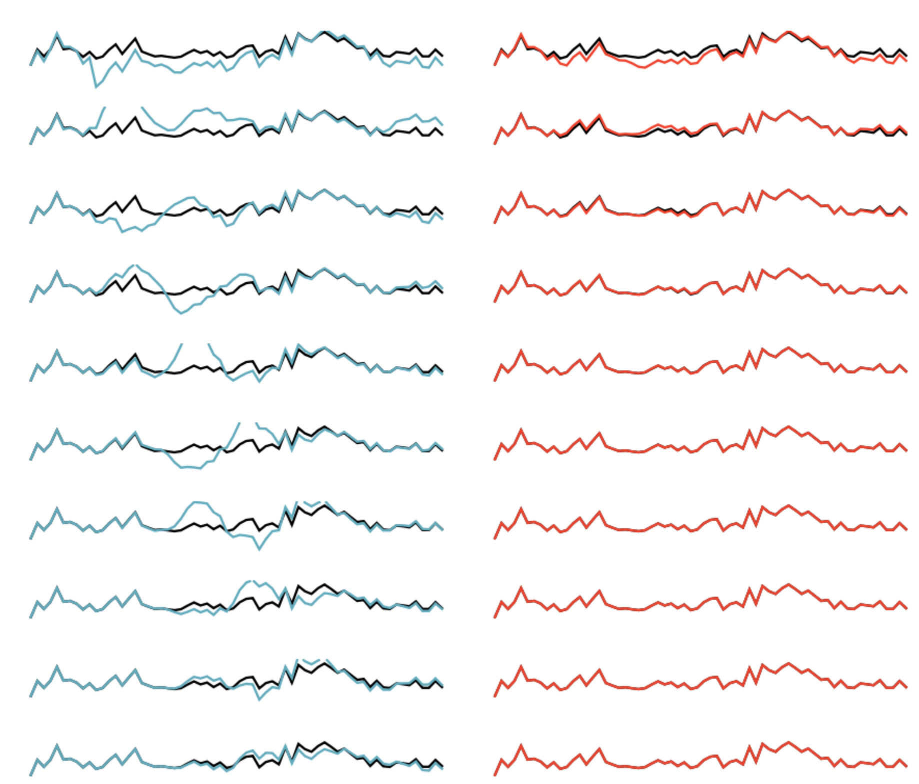


When LDS are parameterized by their (log) eigenvalues, a unitary constraint is trivial. This is helpful for capturing long dependencies, as in the copy problem (above) and the add problem (below).





Our construction, called LDStack, is always faster than standard GPU implementations of RNNs. On long sequences, it is even faster than the highly-optimized CuDNN LSTM. We expect these performance results to improve, since our implementation – research code in both Python and CUDA - is not yet optimized.



Our additive corrections are inspired by a multiplicative approximation technique in control theory. It has been used to analyze continuous nonlinear systems, and to develop controllers. Both are easier to analyze tractable than generic RNNs.

Future Work

- Memory use scales linearly with the height of the stack
- Has $O(n^2 d)$ parameters rather than $O(n^2 + nd)$
- Conditioning of Vandermonde diagonalization
- Tradeoffs of additive versus multiplicative corrections?

Key References

- Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In International Conference on Machine Learning, pages 1120–1128, 2016. (Unitary RNNs for the copy and add problems)
- Eric Martin and Chris Cundy. Parallelizing linear recurrent neural nets over sequence length. In International Conference on Learning Representations, 2018. (Algorithm for parallel linear recurrence)
- María Tomás-Rodríguez and Stephen P Banks. Linear, time-varying approximations to nonlinear dynamical systems: with applications in control and optimization, volume 400. Springer Science & Business Media, 2010. (Multiplicative approximation technique from control theory)