

NoMoPy: Noise Modeling in Python

Results:

- *NoMoPy*: A tool for statistical modeling of noise in the time-domain
- Code ready for release

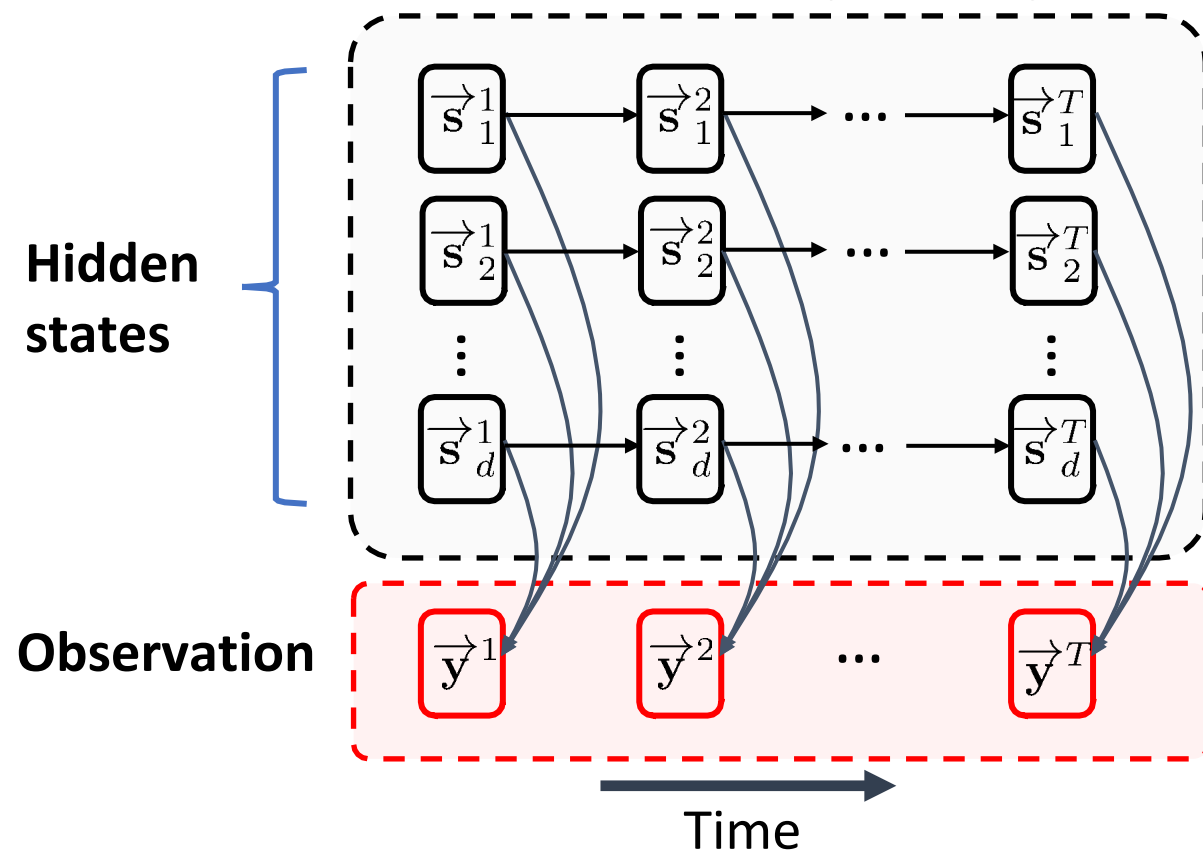


Outline

- Overview of *NoMoPy* and its capabilities
- Demonstration of inference from multi-dimensional observation vectors
- Next steps for the project

Factorial hidden Markov models (FHMMs)

[Ghahramani & Jordan, Machine Learning (1997)]



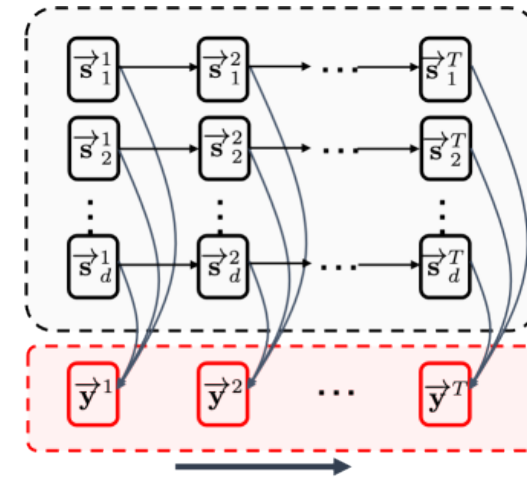
- d independent k -level fluctuators
- Additive contributions to the observable \mathbf{y}
- k^d -dimensional state space for d fluctuators
- White noise background of strength σ (representing e.g. noise floor of measurement)

This category of problem appears in diverse fields:

- Audio signal processing [Wholmayer & Pernkopf, ISCA Interspeech (2008)]
- Load monitoring/power disaggregation [Kolter & Jaakkola, AI & Statistics (2012)]
- Characterization of random telegraph noise in semiconductors [Puglisi & Pavan, ECTI Trans. (2014)]

Factorial hidden Markov models (FHMMs)

Model parameter	Description	Dimension
A	Transition probability matrix	(d,k,k)
W	Observation matrix	(d,o,k)
C	Observation covariance matrix	(o,o)
π	Initial state distribution	(d,k)



- d : (# independent hidden objects)
- k : (# possible states for each hidden object)
- o (dimension of the observation vector)
- Total hidden state space dimension: k^d

- **Standard hidden Markov model:** $d = 1$, k possibly large
- **Factorial hidden Markov model:** $d \geq 1$, k typically small
- Factorial structure of the problem dramatically reduces model parameter space (dk^2 transition matrix elements rather than k^{2d})
- Expensive part of fitting: Expectation step (i.e. computing partition function)

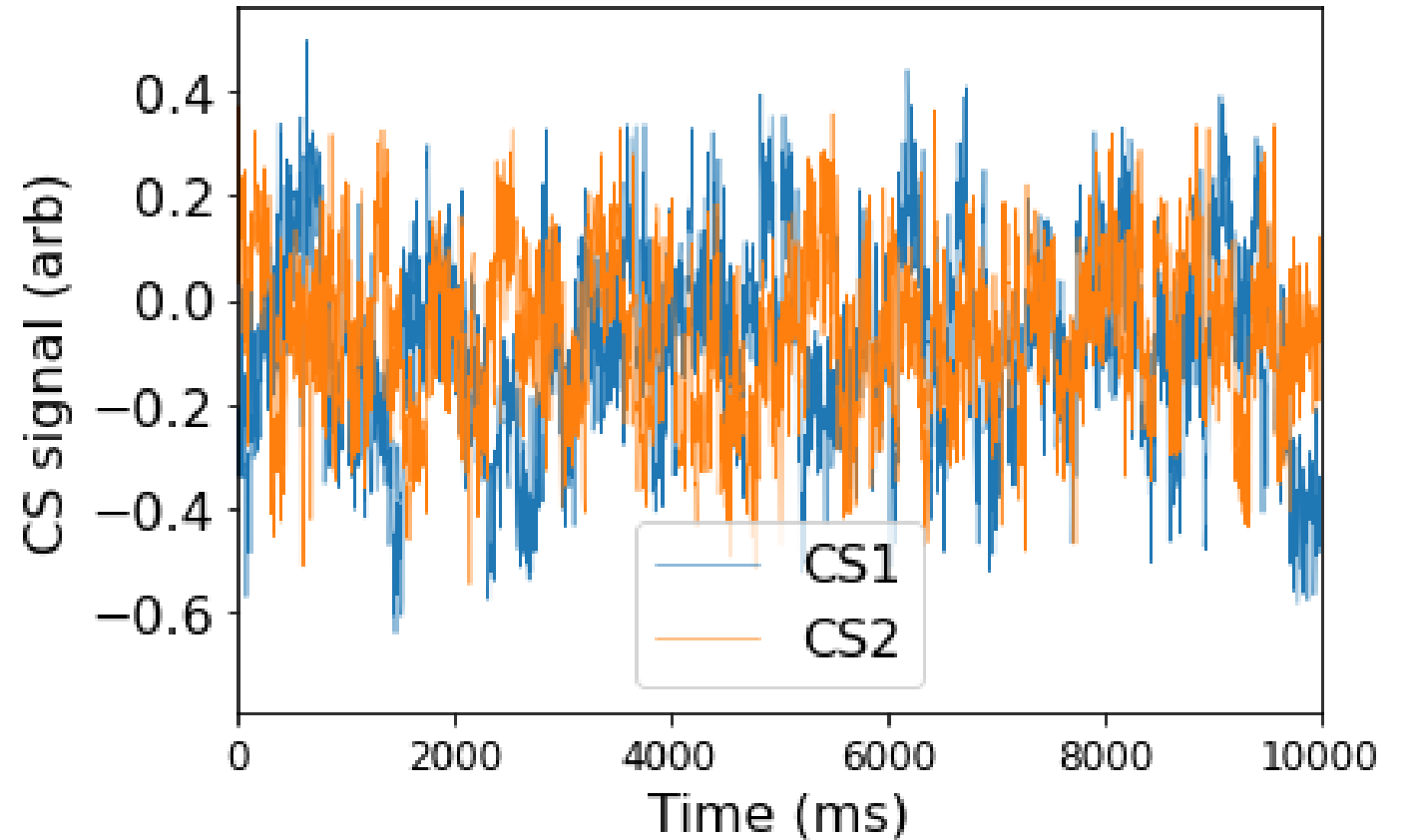
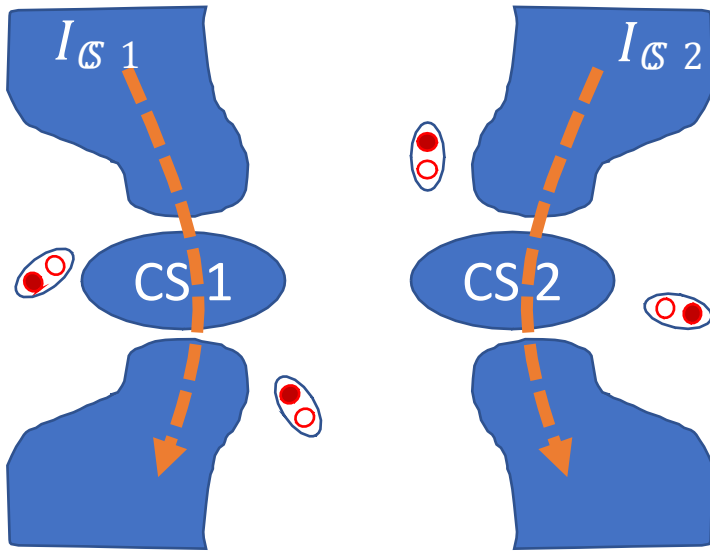
NoMoPy features:

- Factorial hidden Markov model fitting/simulation
 - Includes standard hidden Markov models as a special case
- Algorithms implemented for Expectation step [Ghahramani & Jordan (1997)]
 - Exact
 - Mean field
 - Gibbs sampling
 - Structured variational approximation
- Stochastic training for robust parameter optimization
- Reconstruction of hidden state trajectory w/ Viterbi algorithm
- Cross-validation for model selection through time-wise train/test splitting
- Uncertainty quantification
 - Hessian evaluation of log-likelihood
 - Bootstrapping
- Generation of simulated noise from ensembles of thermal two-level fluctuators
- Higher-order spectrum evaluation (bispectrum) for non-Gaussianity estimation
- Well-suited for parallel computing (multiple cores on a single machine or large-scale HPC

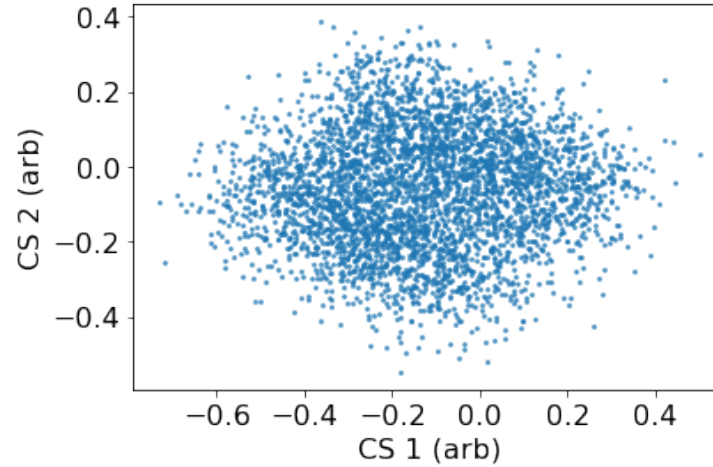
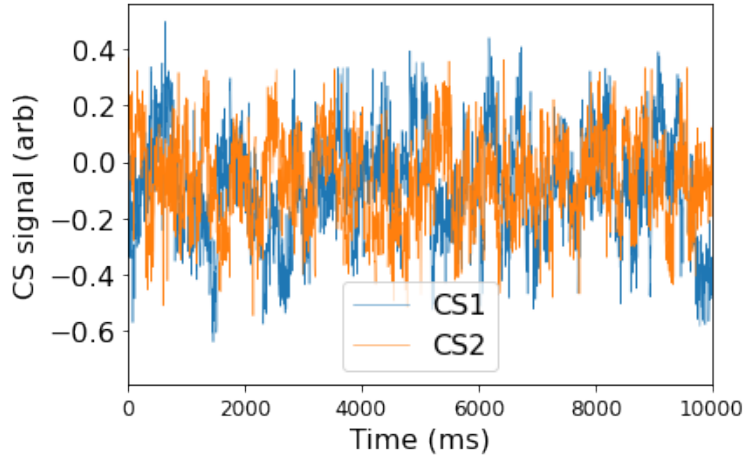


Example: Noise characterization with simultaneous measurements

- Assume white noise (mildly correlated) on each charge sensor
- **Q:** Can we infer the number of TLFs and their relative coupling strengths to each charge sensor?

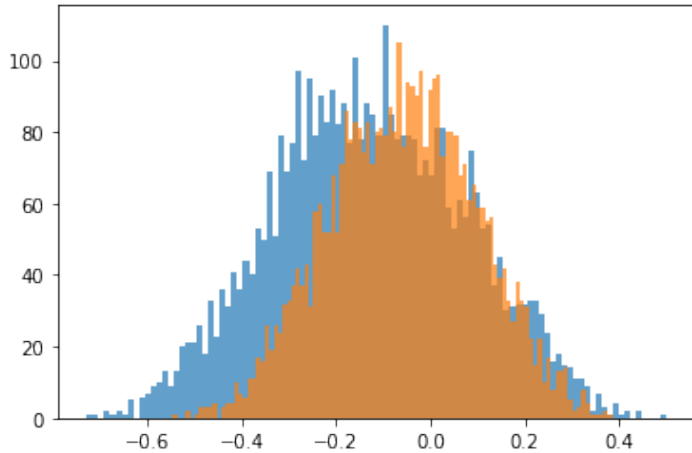


Example: Noise characterization through simultaneous measurements

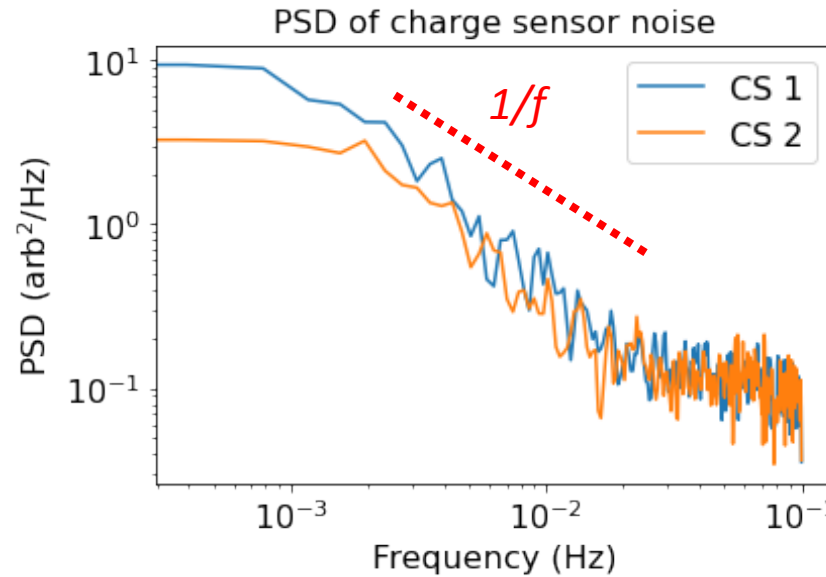


No obvious clustering

Noise histograms



Gaussian-distributed
(to eyeball norm)



Exact model:

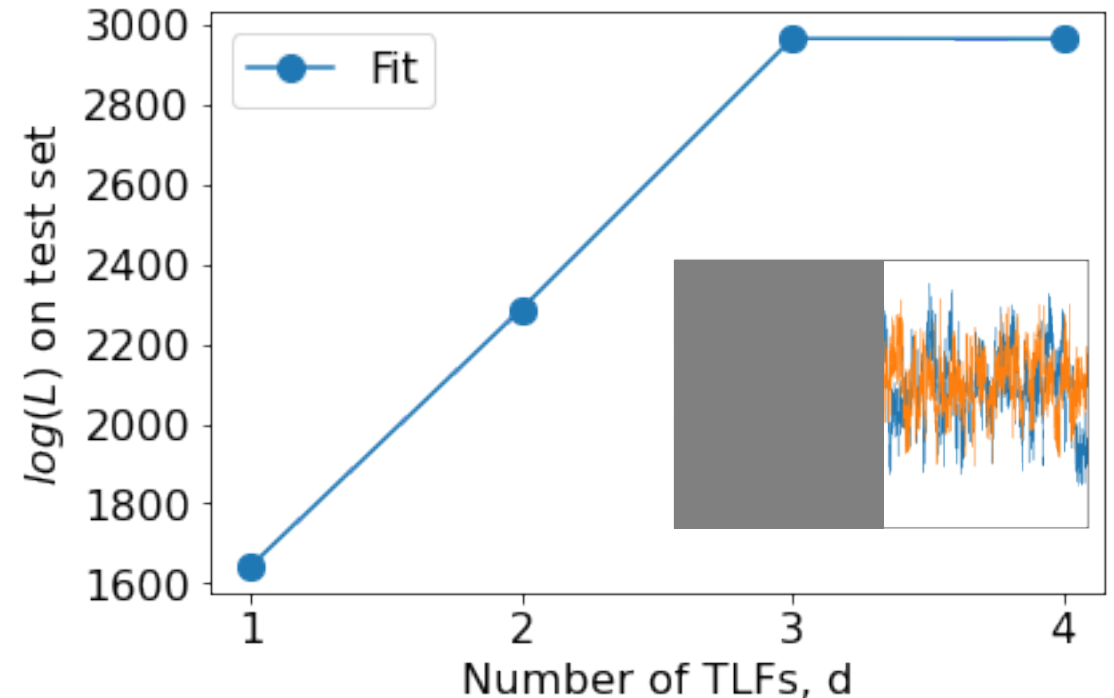
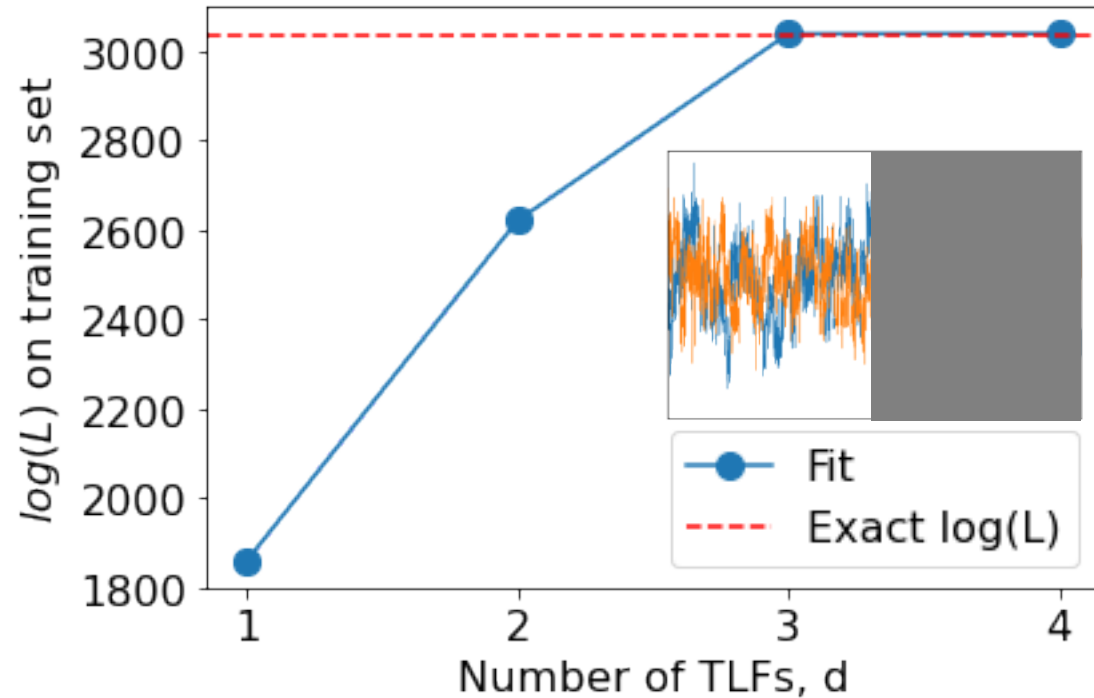
- d=3 TLFs
- White noise std deviation: 0.1
- Fluctuator weights: ~0.05-0.2 (noise "S/N" ~ 0.5-2)

Example: Noise characterization through simultaneous measurements
Simple cross-validation through train/test split of timeseries

Fit to training set

Fix model parameters

Test on training set



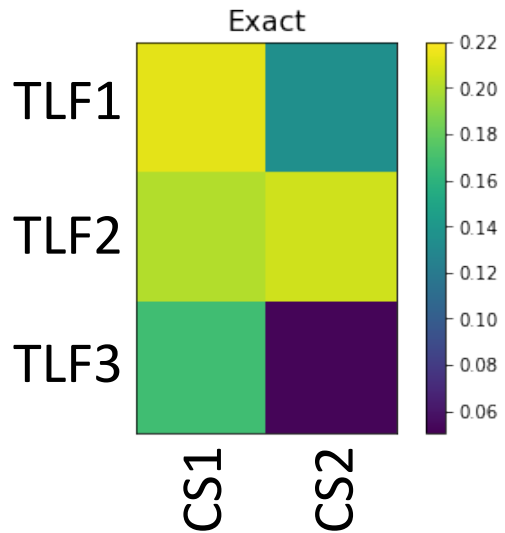
- It's clear that adding more fluctuator degrees of freedom doesn't improve $\log(L)$
- Based on this, we would settle on $d=3$ as most economical and descriptive

Exact:

Covariance matrix:

$$C_{\text{exact}} = \begin{pmatrix} 10^{-2} & 10^{-4} \\ 10^{-4} & 10^{-2} \end{pmatrix}$$

Fluctuator weights:



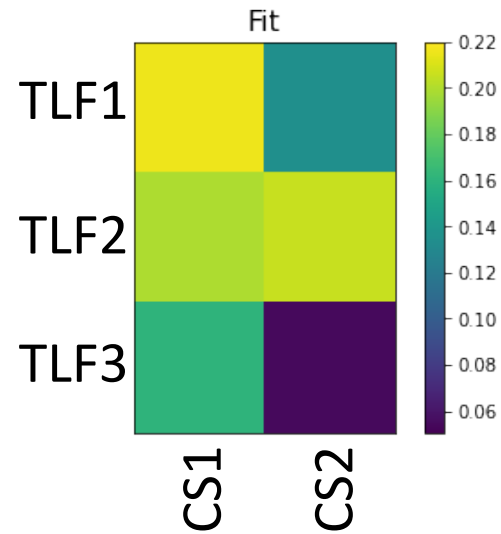
Switching rates (kHz):

$$\{10^{-2}, 3.16 \times 10^{-3}, 10^{-4}\}$$

Fit (d=3):

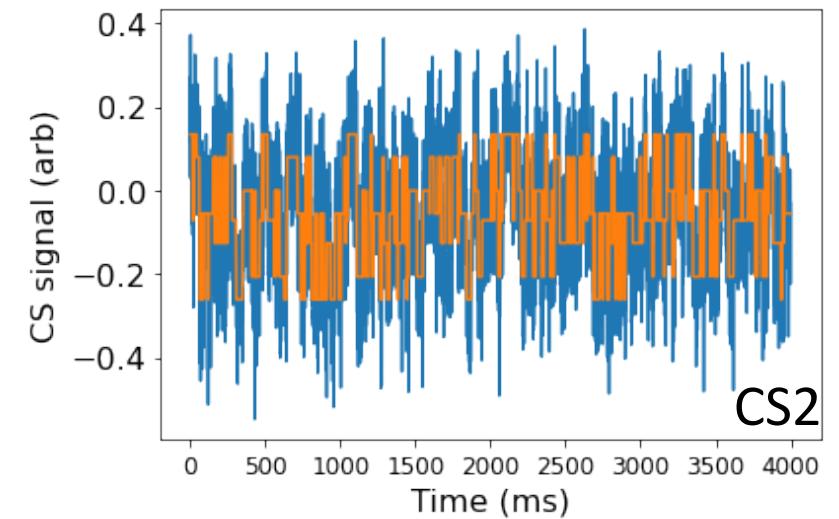
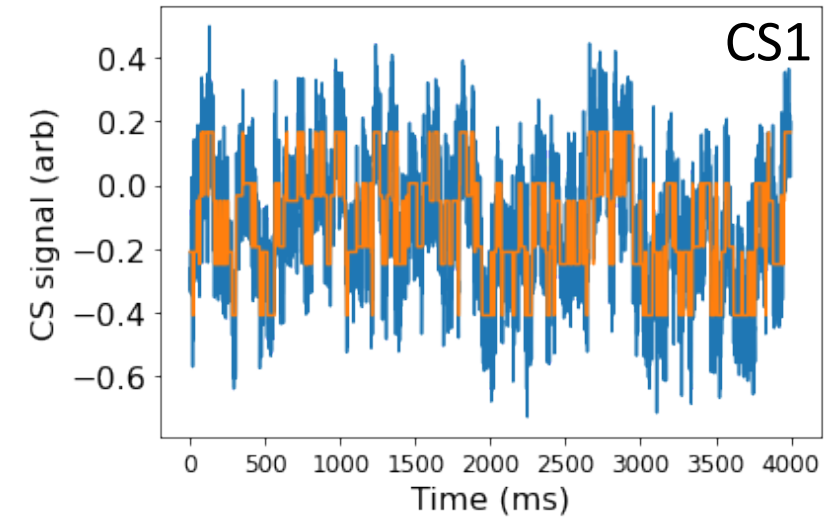
$$C_{\text{fit}} = \begin{pmatrix} 1.01 \times 10^{-2} & 1.26 \times 10^{-4} \\ 1.26 \times 10^{-4} & 0.996 \times 10^{-2} \end{pmatrix}$$

Inter-CS correlations captured well



$$\{0.88 \times 10^{-2}, 2.82 \times 10^{-3}, 1.57 \times 10^{-4}\}$$

Reconstruction of hidden state trajectory through Viterbi algorithm:



Applications for which *NoMoPy* may be useful:

- Characterization of charge noise environment (including correlated measurements)
 - Triangulation of discrete charge fluctuators
- Hyperfine noise from relatively few ($O(10)$) nuclear spins
- Qubit drift characterization

Results:

- *NoMoPy*: Stable and usable code for time-domain noise analysis
- Copyright assertion for public release completed at Sandia

Next steps for this project:

- Additional documentation and examples
- Citation paper writing
- *NoMoPy* release for community use (pending sponsor approval)

In parallel, switching gears to continue improved semiconductor qubit modeling:

- Multi-electron multi-valley effective mass theory capability implementation in Sandia's *Laconic* PDE solver code

