Accelerating OED for Kuramoto Synchronization with ML

Project: “Accelerating Optimal Experimental Design for Robust Synchronization of Uncertain Kuramoto Oscillator Model Using Machine Learning”.

Paper Introduction

- **Context**: Synchronization in uncertain coupled oscillator networks (Kuramoto) is fundamental in power systems, neuroscience, and distributed control.
- **Gap**: OED for robust synchronization is computationally intensive due to repeated trajectory simulations and decision searches.
- **Contribution**: A machine-learning-accelerated OED framework using a neural surrogate on GPU to estimate MOCU and select experiments efficiently.
- **Validation**: Demonstrates accuracy vs ODE baselines and substantial speedups across multiple network sizes and uncertainty settings.

Paper Purpose

- Formulate robust synchronization as minimizing MOCU under coupling uncertainty.
- Develop scalable selection policies (iNN, NN, iODE, ODE, ENTROPY, RANDOM).
- Replace expensive ODE-based marginal evaluations with a trained NN surrogate without degrading decisions.
- Empirically assess accuracy and runtime across network sizes (e.g., N=5, N=7) and uncertainty regimes.

Kuramoto Model Introduction

- **Dynamics**: For N oscillators with phases θ_i and natural frequencies ω_i:

$$\dot{\theta}_i = \omega_i + \sum_{j \neq i} a_{ji} \sin(\theta_j - \theta_i).$$

- **Order parameter**: coherence measure

$$r\, e^{i\psi} = \frac{1}{N} \sum_{j=1}^{N} e^{i\theta_j}, \quad r \in [0, 1],$$

  $r \to 1$ indicates phase locking (synchronization), $r \approx 0$ incoherence.

- **Coupling**: matrix $A = [a_{ij}]$ typically symmetric with zero diagonal; strength and heterogeneity drive critical behavior.

- **Heterogeneity**: spread in ω_i competes with coupling to determine if/when synchronization emerges.

- **Applications**: power-grid frequency control, neural synchrony, Josephson junctions, chemical oscillators.

MOCU Concept and Mathematics

- Kuramoto model: $\dot{\theta}_i = \omega_i + \sum_{j \neq i} a_{ji} \sin(\theta_j - \theta_i)$.
- Prior over uncertain couplings: $\Pi(A) = \prod_{i<j} \mathrm{Unif}([a_{ij}^{\mathrm{L}}, a_{ij}^{\mathrm{U}}])$.
- Virtual hub augmentation ($N{+}1$ embedding): add node $N{+}1$ with $a_{i,N+1} = a_{N+1,i} = c$.
- Per-realization minimal augmentation: $c^*(A) = \inf\{c \geq 0 : D(A, c) = 1\}$, where D is a sync test.
- Mean Objective Cost of Uncertainty: $\mathrm{MOCU}(\mathcal{B}) = \mathbb{E}_{A \sim \Pi}[c^*(A)]$.
- OED objective per action e: $e^* = \arg\min_e \; \mathbb{E}_y[\, \mathrm{MOCU}(\mathcal{U}(\mathcal{B}, e, y)) \,]$.

Computing $c^*(A)$ and MOCU

- Decision test $D(A, c)$:
  - ODE backend: RK4 integration with $h = 1/160$, $T = 5\,\mathrm{s}$; sync if post-transient increment spread $\leq 10^{-3}$.
  - NN backend: classifier on $(N{+}1)^2$ features decides sync/non-sync.
- Search: exponential bracketing on c then binary search until interval $< 2.5 \times 10^{-4}$.
- Monte Carlo: $K_{\max} = 20480$ samples on GPU; trim 0.5% tails; average to estimate MOCU.

Methods (Selection Policies)

- **NN / iNN**: NN-accelerated MOCU; i updates bounds after each chosen edge.
- **ODE / iODE**: ODE-based MOCU with/without iterative updating.
- **ENTROPY**: Information-gain heuristic on coupling uncertainty.
- **RANDOM**: Baseline random edge selection.

Experiment Groups in 2021 Paper

- **Network sizes**: $N \in \{5, 7, 8, \dots\}$ (paper evaluates multiple sizes; this repo includes N=5 and N=7).
- **Uncertainty regimes**: bounds proportional to $\frac{1}{2}|\omega_i - \omega_j|$ with scaling; additional structured scalings in N5; file-defined bounds in N7.
- **Backends**: ODE vs NN surrogate for MOCU estimation.
- **Policies**: iNN, NN, iODE, ODE, ENTROPY, RANDOM.
- **Metrics**: MOCU vs iteration, time per iteration, total runtime, selected sequences.

Experiment Setup (per paper and code)

- Time grid: $\Delta t = 1/160$, horizon $T = 5\,\mathrm{s} \Rightarrow M = T/\Delta t$.
- Sampling: $K_{\max} = 20480$ Monte Carlo samples (GPU parallel), trimming for stability.
- N5: hardcoded ω; bounds from $0.85/1.15 \times \frac{1}{2}|\omega_i - \omega_j|$ with subset scalings (0.3, 0.45), symmetrized.
- N7: ω, lower/upper bounds from `uncertaintyClass/` files.
- For each sampled A: skip if synchronized; otherwise run each policy, log MOCU/time/sequence; repeat for 100 unstable networks.

Results in Paper (Insert)

- NN/iNN closely match ODE/iODE MOCU reduction with *large* runtime savings.
- iNN/iODE typically outperform batch variants as iterations progress.
- *Insert paper tables/plots here: final MOCU, speedups, convergence across sizes.*

Reproduction Results: N=5 and N=7 Oscillators

Experimental Setup:

- **N=5:** Natural frequencies $\omega = [-2.5, -0.667, 1.167, 2.0, 5.833]$
- **N=7:** Seven oscillators with diverse frequency distribution
- Monte Carlo samples: $K = 20,480$ (RTX 4090 optimization)
- Simulations per method: 100
- Time integration: $\Delta t = 1/160$, $T = 5$ seconds

N=5 Reproduction Results (100 Simulations):

Method	**Initial MOCU**	**Final MOCU**	**Improvement**
iNN	0.3094	0.2859	7.6%
NN	0.3084	0.2859	7.3%
ODE	0.3088	0.2860	7.4%
ENTROPY	0.3087	0.2860	7.4%
RANDOM	0.3083	0.2863	7.2%

N=7 reproduction results pending.

Analysis and Takeaways

- NN/iNN achieve similar MOCU reduction to ODE/iODE while reducing runtime dramatically (per paper and code profiling).
- Iterative strategies (iNN/iODE) adapt to updated bounds, offering consistent gains over batch selection.
- Scaling to N7 increases cost, but NN surrogate preserves acceleration and decision quality.