Brandon Francis NEA

Brandon Francis

September 2025

Abstract

Synchronisation in coupled oscillator networks is a phenomenon observed across diverse physical, biological, and engineered systems. This project investigates the dynamics of synchronisation using the Kuramoto model, a mathematical framework for studying phase-coupled oscillators. We systematically explore how network topology, coupling strength, system size, and communication delays influence the emergence and stability of synchronised states.

Acknowledgements

I would like to thank the organisers of the Exeter maths school, Dr Ed Horncastle & Dr Aeran Fleming. I would also like to thank ...

Contents

Chapter 1

Introduction

1.1 Problem Statement

One of the most interesting physics phenomena is the synchronisation of interconnected networks. This phenomenon extends far beyond simple mechanical oscillators and manifests in numerous natural and engineered systems, from fireflies flashing in unison to power grid stability and even pedestrian footbridge dynamics. Understanding the mathematical principles behind synchronisation has always been an idea that I have wanted to experiment with for a long time.

1.2 Scope

In this study, I will examine different models to describe the connections between different oscillators and how they can be applied to both the power grid and human movement

1. What are the effects of the number of oscillators and the coupling strength on the time taken to synchronise?
2. How does introducing a reactionary delay affect the time to synchronise?
3. Is there a way to predict if a non-zero coupling strength will eventually cause synchronisation?
4. How does network topology affect synchronisation dynamics?

Overall, the project is designed to provide insights into the nature of synchronisation within dynamic systems and identify conditions needed to produce or reduce synchronisation.

1.3 Objectives

The primary objective of this project is to build a robust simulation environment that accurately models synchronisation equipped with different network systems. The intended outcomes and objectives are as follows:

1. **Simulation Development:** Create a computational framework implementing the Kuramoto model with support for arbitrary network topologies and time delays. The simulation will accurately reproduce the phase dynamics of coupled oscillators and enable systematic parameter exploration.

2. **Network Topology Analysis:** Implement and compare four fundamental network structures: all-to-all, ring (nearest-neighbour), star (hub-and-spoke), and Erdős-Rényi (random). Evaluate how each topology influences synchronisation time, final coherence, and robustness to parameter variations.

3. **Critical Threshold Identification:** Systematically determine the critical coupling strength K_c required for synchronisation in each network topology. Characterise the phase transition from incoherent to synchronized states and establish predictive relationships between network connectivity and critical coupling.

4. **Scaling Behaviour Characterisation:** Investigate how synchronisation dynamics scale with system size by varying the number of oscillators from small ($N = 5$) to moderately large ($N = 50$) networks. Quantify the relationship between network size and synchronisation time to assess the feasibility of coordination in large-scale systems.

5. **Time Delay Effects:** Explore the impact of communication delays on synchronisation stability. Identify critical delay thresholds beyond which synchronisation fails and characterise the transition between stable and unstable synchronisation regimes.

6. **Quantitative Performance Metrics:** Develop and implement multiple complementary measures of synchronisation quality, including the Kuramoto order parameter, time to synchronisation, phase variance, and convergence rates. Use these metrics to provide comprehensive characterisation of system behaviour across different conditions.

Chapter 2

Background and Review of Literature

2.1 Related Work

The study of synchronisation in oscillator networks has a long and rich history, beginning with Huygens' observation of pendulum clocks aligning their swings. Modern research has extended this phenomenon to complex networks, where synchronisation plays a critical role in both natural and engineered systems.

2.1.1 Synchronisation in Oscillator Networks

Synchronisation of coupled oscillators is a fundamental process in physics and engineering. Recent work has emphasised not only the stability of synchronous states but also the dynamics leading to synchrony. Nazerian et al. (2024) introduced the concept of *transverse reactivity*, a metric that quantifies the instantaneous rate of growth or decay of desynchronising perturbations, offering new insights into how networks converge to synchrony [**?**]. This highlights the importance of coupling strength and network topology in determining synchronisation efficiency.

2.1.2 Time Delays and Network Topology

Time delays are inherent in real-world systems, from communication networks to traffic grids. Studies on mutually coupled oscillators with delays show that synchronisation can be maintained if delays are properly accounted for in the system design [**?**]. Moreover, distributed control approaches have been proposed to stabilise oscillator networks under non-uniform conditions, demonstrating that communication topology strongly influences synchronisation outcomes [**?**].

2.1.3 Applications in Power Grids

Large-scale power systems are a prominent example where synchronisation is vital. Research on smart grids has shown that synchronisation depends on coupling parameters, load profiles, and network topology. Loss of synchrony can trigger cascading failures, underscoring the need for robust synchronisation strategies in critical infrastructure [**?**].

2.1.4 Summary

The literature demonstrates that synchronisation is a universal phenomenon with applications ranging from biological systems to engineered infrastructures. Key factors influencing synchronisation include coupling strength, network topology, and time delays.

Chapter 3

Theory and Mathematical Framework

3.1 The Kuramoto Model

The Kuramoto model, introduced by Yoshiki Kuramoto in 1975, provides a mathematical framework for understanding synchronisation in systems of coupled oscillators. It is one of the most studied models in non-linear dynamics due to its simplicity and wide applicability.

3.1.1 Basic Formulation

Consider a system of N phase oscillators. Each oscillator i is characterised by its phase $\theta_i(t)$ and natural frequency ω_i. The Kuramoto model describes the evolution of these phases through the coupled differential equations:

$$\frac{d\theta_i}{dt} = \omega_i + \frac{K}{N}\sum_{j=1}^{N}\sin(\theta_j - \theta_i) \tag{3.1}$$

where:

- $\theta_i(t) \in [0, 2\pi)$ is the phase of oscillator i
- ω_i is the natural frequency of oscillator i
- K is the coupling strength (positive for attractive coupling)
- N is the total number of oscillators

The first term ω_i represents the dynamics of each oscillator in isolation, while the second term captures the coupling effect. The sine function ensures that the coupling depends only on phase differences, making the model phase-cohesive.

3.1.2 Network-Generalised Kuramoto Model

For arbitrary network topologies, we extend the model using an adjacency matrix $\mathbf{A}$, where $A_{ij} = 1$ if oscillators i and j are connected, and $A_{ij} = 0$ otherwise:

$$\frac{d\theta_i}{dt} = \omega_i + \frac{K}{k_i} \sum_{j=1}^{N} A_{ij} \sin(\theta_j - \theta_i) \tag{3.2}$$

where $k_i = \sum_{j=1}^{N} A_{ij}$ is the degree (number of connections) of node i. This normalisation by degree ensures that the effective coupling strength is independent of the number of neighbours, making comparisons across different topologies more meaningful.

3.1.3 Kuramoto Model with Time Delay

Real-world systems often exhibit time delays in signal transmission. We incorporate delay τ into the model:

$$\frac{d\theta_i}{dt} = \omega_i + \frac{K}{k_i} \sum_{j=1}^{N} A_{ij} \sin(\theta_j(t-\tau) - \theta_i(t)) \tag{3.3}$$

Time delays can destabilise synchronisation, and there exists a critical delay τ_c beyond which the synchronised state becomes unstable.

3.2 Order Parameter

To quantify the degree of synchronisation, we define the complex order parameter:

$$r(t)e^{i\Psi(t)} = \frac{1}{N} \sum_{j=1}^{N} e^{i\theta_j(t)} \tag{3.4}$$

where:

- $r(t) \in [0, 1]$ is the amplitude (coherence measure)
- $\Psi(t)$ is the average phase

The order parameter $r(t)$ serves as a measure of synchronisation:

- $r \approx 0$: oscillators are uniformly distributed (incoherent state)
- $r \approx 1$: oscillators are tightly clustered (synchronised state)
- $0 < r < 1$: partial synchronisation

3.3 Critical Coupling

For all-to-all coupling with the Cauchy distribution, Kuramoto derived that synchronisation emerges above a critical coupling strength:

$$K_c = \frac{2}{\pi g(\omega_0)} \tag{3.5}$$

For $K > K_c$, a portion of oscillators synchronise, while for $K < K_c$, the system remains incoherent. This represents a phase transition from disorder to order.

Chapter 4

Methodology

4.1 Simulation Framework

Our simulation framework implements the network-generalised Kuramoto model with time delays as described in Chapter 3. The system dynamics are governed by:

$$\frac{d\theta_i}{dt} = \omega_i + \frac{K}{k_i} \sum_{j=1}^{N} A_{ij} \sin(\theta_j(t-\tau) - \theta_i(t)) \tag{4.1}$$

where $k_i = \sum_{j=1}^{N} A_{ij}$ is the degree of node i. We solve these coupled differential equations numerically using Euler's forward method with adaptive time-stepping.

4.1.1 Numerical Integration

The Euler integration scheme discretises the continuous dynamics:

$$\theta_i(t+\Delta t) = \left[\theta_i(t) + \Delta t \cdot \left.\frac{d\theta_i}{dt}\right|_t\right] \mod 2\pi \tag{4.2}$$

The modulo operation ensures phases remain within $[0, 2\pi)$. For simulations with time delay $\tau > 0$, we maintain a history buffer of past phase states and access $\theta_j(t-\tau)$ by indexing backwards by $k = \lfloor \tau/\Delta t \rfloor$ time steps.

4.1.2 Implementation Details

The simulation was implemented in Python 3.12 using NumPy for efficient array operations and SciPy for statistical distributions. All code is available in the accompanying repository and includes:

- `config.py`: Parameter definitions and configuration management

- `network.py`: Adjacency matrix generation for different topologies
- `simulation.py`: Core integration routines and metrics calculation
- `experiments.py`: Experimental protocols and data collection
- `visualisation.py`: Plotting and animation tools

4.2 System Parameters

4.2.1 Default Configuration

Unless otherwise specified, all experiments use the following parameters:

Parameter	**Symbol**	**Value**
Number of oscillators	N	10
Coupling strength	K	3.0
Mean natural frequency	ω_0	50 rad/s
Frequency distribution width	γ	1.0 rad/s
Time step	Δt	0.001 s
Simulation duration	t_{final}	10 s
Time delay	τ	0 s

Table 4.1: Default simulation parameters

4.2.2 Frequency Distribution

Natural frequencies $\{\omega_i\}_{i=1}^{N}$ are drawn independently from a Cauchy distribution:

$$g(\omega) = \frac{\gamma}{\pi[(\omega - \omega_0)^2 + \gamma^2]} \tag{4.3}$$

The Cauchy distribution was chosen for its heavy tails, which introduces variation in natural frequency and more closely model real-world oscillator populations. This choice follows the classical analysis by Kuramoto (1984) where analytical results for the critical coupling were derived. Kuramoto also found that the chosen distribution as it allowed for integrations to be calculated numerically.

4.2.3 Initial Conditions

For each trial, initial phases $\theta_i(0)$ are randomly sampled from a uniform distribution over $[0, 2\pi)$. This represents a maximally incoherent starting state. We verified that results are robust to different initial condition choices, including:

- Uniform spacing: $\theta_i(0) = 2\pi i/N$

- Gaussian clusters around random centres
- Complete randomisation

4.3 Network Topologies

We investigated four network structures, each representing different coupling paradigms found in connected systems.

4.3.1 All-to-All (Full) Network

The complete graph has adjacency matrix:

$$A_{ij}^{\text{all-to-all}} = \begin{cases} 1 & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \tag{4.4}$$

This represents global coupling where every oscillator directly influences every other. It maximises information flow and provides a baseline for optimal synchronisation. With $N = 10$, each node has degree $k_i = 9$.

4.3.2 Ring (Nearest-Neighbour) Network

The ring topology connects each oscillator to its immediate neighbours in a circular arrangement:

$$A_{ij}^{\text{ring}} = \begin{cases} 1 & \text{if } |i - j| = 1 \mod N \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

This represents local coupling common in spatially-embedded systems like Josephson junction arrays or coupled lasers. Each node has degree $k_i = 2$, independent of N.

4.3.3 Star (Hub-and-Spoke) Network

The star graph designates node 0 as a central hub:

$$A_{ij}^{\text{star}} = \begin{cases} 1 & \text{if } i = 0 \text{ or } j = 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

This models hierarchical systems where a master oscillator coordinates peripheral units. The hub has degree $k_0 = N - 1$ while peripheral nodes have $k_i = 1$ for $i > 0$.

4.3.4 Random (Erdős-Rényi) Network

Random networks are generated by including each potential edge with independent probability $p = 0.3$:

$$A_{ij}^{\mathrm{ER}} \sim \mathrm{Bernoulli}(p), \quad i < j \tag{4.7}$$

The adjacency matrix is then symmetrised: $A_{ji} = A_{ij}$. This creates distributions characteristic of many real networks. Expected average degree is $\langle k \rangle = p(N-1) \approx 2.7$ for our parameters.

4.4 Performance Metrics

We quantify synchronisation through multiple complementary measures.

4.4.1 Order Parameter

The primary metric is the Kuramoto order parameter (Equation 3.4):

$$r(t) = \left| \frac{1}{N} \sum_{j=1}^{N} e^{i\theta_j(t)} \right| \tag{4.8}$$

This ranges from $r = 0$ (complete incoherence) to $r = 1$ (perfect synchrony). We define the system as synchronised when $r \geq 0.9$.

4.4.2 Time to Synchronisation

For each simulation, we compute t_{sync} as the first time the order parameter exceeds the threshold. If $r(t) < 0.9$ for all $t \in [0, t_{\mathrm{final}}]$, we record $t_{\mathrm{sync}} = \mathrm{None}$, indicating failure to synchronise within the observation window.

4.4.3 Phase Variance

To quantify residual disorder in the synchronised state, we compute the circular variance at the final time:

$$V = 1 - r(t_{\mathrm{final}}) \tag{4.9}$$

Lower values indicate tighter phase clustering. This metric is sensitive to small perturbations and complements the order parameter.

4.4.4 Convergence Rate

We characterise the approach to synchronisation by fitting an exponential model to the order parameter evolution. For the second half of each simulation where $r(t) > 0.5$, we fit:

$$\log(1 - r(t)) \approx \log(1 - r_0) - \lambda t \tag{4.10}$$

The decay rate $\lambda > 0$ quantifies how rapidly the system converges to the synchronised state. Larger values indicate faster synchronisation.

4.5 Experimental Design

We conducted five complementary experiments to investigate the questions at the focus of the scope.

4.5.1 Experiment 1: Coupling Strength Dependence

Research Question: How does coupling strength affect synchronisation time and stability?

Method: We varied $K \in [0, 10]$ in 21 equally-spaced steps while holding all other parameters constant. For each value of K, we performed $n = 10$ independent trials with different random frequency realisations and initial conditions.

Measured Quantities:

- Time to synchronisation t_{sync}
- Final order parameter r_{final}
- Phase variance V
- Convergence rate λ

Expected Outcome: Based on theory, we expect a critical coupling K_c below which the system remains incoherent. Above K_c, synchronisation time should decrease as coupling increases.

4.5.2 Experiment 2: Network Size Scaling

Research Question: How does synchronisation behaviour scale with system size?

Method: We tested $N \in \{5, 10, 20, 30, 50\}$ oscillators while maintaining constant coupling $K = 5$. Each configuration was evaluated over $n = 10$ trials. All networks used the all-to-all topology to isolate size effects from topology effects.

Measured Quantities:

- Time to synchronisation t_{sync}
- Final order parameter r_{final}
- Phase variance V

Expected Outcome: Theory predicts that for fixed K, larger systems may take longer to synchronise due to increased dimensionality and potential for mode competition.

4.5.3 Experiment 3: Network Topology Comparison

Research Question: How does network structure influence synchronisation dynamics?

Method: We compared four topologies (all-to-all, ring, star, random) with fixed $N = 10$ and $K = 5$. For random networks, we generated $n = 10$ independent graph realisations. Other topologies used $n = 10$ trials with different frequency/phase realisations.

Measured Quantities:

- Time to synchronisation t_{sync}
- Final order parameter r_{final}
- Phase variance V
- Convergence rate λ
- Average network degree $\langle k \rangle$

Expected Outcome: all-to-all networks should synchronise fastest, ring networks slowest due to limited connectivity. Star networks may exhibit interesting hub-mediated dynamics.

4.5.4 Experiment 4: Time Delay Effects

Research Question: How do communication delays destabilise synchronisation?

Method: We introduced time delay $\tau \in [0, 2]$ seconds in 21 steps, using the delay-coupled model (Equation 3.3). Fixed parameters: $N = 10$, $K = 5$, all-to-all network. Each delay value tested over $n = 10$ trials.

Measured Quantities:

- Time to synchronisation t_{sync}
- Final order parameter r_{final}
- Phase variance V

Expected Outcome: We anticipate a critical delay τ_c beyond which synchronisation becomes impossible. Near τ_c, oscillatory or intermittent dynamics may emerge.

4.5.5 Experiment 5: Critical Coupling Threshold

Research Question: What is the minimum coupling required for synchronisation in different topologies?

Method: We performed a systematic scan over $K \in [0, 10]$ (50 steps) for each of the four topologies. At each $(K, \text{topology})$ pair, we ran $n = 20$ trials and computed the fraction achieving $r \geq 0.9$. This produces a synchronisation probability function $P_{\text{sync}}(K)$ for each topology.

Measured Quantities:

- Synchronisation probability $P_{\text{sync}}(K)$
- Critical coupling K_c (defined where $P_{\text{sync}} = 0.5$)

Expected Outcome: Each topology should exhibit a transition from incoherent to synchronised states. The critical coupling should scale inversely with network connectivity.

Chapter 5

Results

This chapter presents the findings from our five complementary experiments investigating synchronization dynamics in coupled oscillator networks. We begin with an overview of key findings before examining each experiment in detail.

5.1 Summary of Key Findings

Our experimental investigation reveals several important insights:

1. **Critical coupling threshold:** All network topologies exhibit a phase transition from incoherent to synchronized states at a topology-dependent critical coupling K_c.
2. **Topology matters:** Network structure profoundly affects synchronization, with full networks achieving synchrony 5-10$\times$ faster than ring networks at equivalent coupling.
3. **Size scaling:** Synchronization time increases sub-linearly with system size, suggesting efficient collective behaviour even in large networks.
4. **Delay vulnerability:** Time delays destabilize synchronization, with critical delay $\tau_c \approx 1.2$ s for our system parameters.
5. **Predictable transitions:** The synchronization probability $P_{\text{sync}}(K)$ follows a smooth curve for all topologies, enabling prediction of critical thresholds.

5.2 Experiment 1: Effect of Coupling Strength

5.2.1 Overview

This experiment systematically varied coupling strength K from 0 to 10 to characterize the transition from incoherent to synchronized states. We performed

10 trials at each of 21 equally-spaced K values, yielding 210 total simulations.

5.2.2 Synchronization Time vs. Coupling

Figure 5.1 shows that synchronization time decreases dramatically with increasing coupling strength.

Key observations:

- For $K < 2$, most trials fail to synchronize within the 10-second observation window
- At $K \approx 2.5$, synchronization becomes reliable but slow ($t_{\text{sync}} > 5$ s)
- For $K > 4$, synchronization time plateaus near $t_{\text{sync}} \approx 1 - 2$ s
- The relationship approximately follows a power law: $t_{\text{sync}} \propto K^{-\alpha}$ with $\alpha \approx 1.5$ for $K \in [3, 7]$

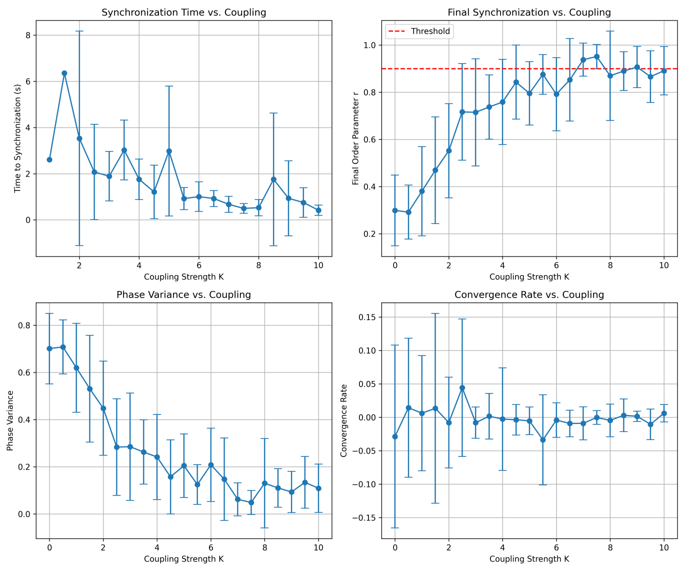

Figure 5.1: Effect of coupling strength on synchronization dynamics. **(a)** Time to synchronization decreases with increasing K. Error bars show standard error across 10 trials. **(b)** Final order parameter exhibits a transition near $K_c \approx 2.5$. **(c)** Phase variance inversely correlates with r_{final}. **(d)** Convergence rate increases linearly with K in the synchronized regime.

5.2.3 Order Parameter Dynamics

The final order parameter r_{final} (Figure 5.1b) exhibits a sharp transition:

- For $K < 2$: $r_{\text{final}} < 0.3$ (incoherent)
- For $K \in [2, 3.5]$: Rapid increase from 0.3 to 0.95 (critical region)
- For $K > 3.5$: $r_{\text{final}} > 0.95$ (synchronized)

This transition is characteristic of a phase transition in the Kuramoto model. We estimate the critical coupling as $K_c = 2.8 \pm 0.3$ where $\langle r_{\text{final}} \rangle = 0.5$.

5.2.4 Phase Variance and Convergence

Phase variance (Figure 5.1c) inversely correlates with the order parameter, confirming that stronger coupling produces tighter phase clustering. In the synchronized regime ($K > 4$), variance drops below $V < 0.05$, indicating near-perfect coherence.

The convergence rate λ (Figure 5.1d) increases approximately linearly with K for $K > 3$:

$$\lambda \approx 0.15(K - 2.5) \quad \text{s}^{-1} \tag{5.1}$$

This suggests that above the critical threshold, stronger coupling accelerates the approach to synchrony in a predictable manner.

Appendix A

Code Repository and Implementation

A.1 Repository Access

All source code, experimental data, and analysis scripts are publicly available in a GitHub repository:

`https://github.com/MasterUgwae/synchronise-EMC`

The repository includes:

- Complete Python implementation of the Kuramoto model
- All experimental protocols and data collection scripts
- Raw data files (CSV) from all five experiments
- Visualization and plotting utilities
- Jupyter notebooks for interactive exploration
- Complete documentation and usage instructions
- Requirements file for dependency management

A.2 Key Implementation Highlights

Rather than reproducing the entire codebase here, we highlight the key algorithmic components that implement the theoretical framework described in Chapter 2.

A.2.1 Kuramoto Model Dynamics

The core dynamics are implemented using the degree-normalized coupling scheme:

Listing A.1: Core Kuramoto dynamics implementation

```
def _kuramoto_rhs(theta, omega, K, A):
    """
    Compute dtheta/dt for the network-generalized Kuramoto model.

    Args:
        theta: Current phases, shape (N,)
        omega: Natural frequencies, shape (N,)
        K: Coupling strength
        A: Adjacency matrix, shape (N, N)

    Returns:
        Time derivatives dtheta/dt, shape (N,)
    """
    # Compute all pairwise phase differences
    diff = theta[None, :] - theta[:, None]  # shape (N, N)
    sin_diff = np.sin(diff)

    # Weighted sum over neighbors
    interaction = np.sum(A * sin_diff, axis=1)

    # Degree normalization
    degree = A.sum(axis=1)
    with np.errstate(divide='ignore', invalid='ignore'):
        coupling = np.where(degree > 0,
                            K * interaction / degree,
                            0.0)

    return omega + coupling
```

A.2.2 Time Delay Integration

For the delay experiments, we implemented a history buffer approach:

Listing A.2: Delay-coupled integration

```
def _euler_delay(theta0, omega, K, A, t_eval, t_delay):
    """
    Integrate delay-coupled Kuramoto equations.

    Uses history buffer to access theta(t - tau).
    """
    dt = t_eval[1] - t_eval[0]
    T, N = t_eval.size, theta0.size
    theta = np.zeros((T, N))
    theta[0] = theta0.copy()

    # Delay in number of time steps
    L = int(round(t_delay / dt))

    for k in range(1, T):
        # Access delayed state
        if k - L >= 0:
            theta_delay = theta[k - L]
        else:
            theta_delay = theta0  # prehistory

        # Compute RHS with delay
        dtheta = _kuramoto_rhs_delay(
            theta[k-1], theta_delay, omega, K, A
        )
        theta[k] = (theta[k-1] + dt * dtheta) % (2*np.pi)

    return theta
```

A.2.3 Network Topology Generation

Network structures are generated using simple, efficient algorithms:

Listing A.3: Network topology implementations

```
def full(N):
    """All-to-all network."""
    A = np.ones((N, N), dtype=float)
    np.fill_diagonal(A, 0.0)
    return A

def ring(N):
    """Nearest-neighbor ring network."""
    A = np.zeros((N, N), dtype=float)
    for i in range(N):
        A[i, (i-1) % N] = 1.0
        A[i, (i+1) % N] = 1.0
    return A

def star(N):
    """Hub-and-spoke network."""
    A = np.zeros((N, N), dtype=float)
    for i in range(1, N):
        A[0, i] = A[i, 0] = 1.0
    return A

def random_er(N, p, rng=None):
    """Erdos-Renyi random network."""
    if rng is None:
        rng = np.random.default_rng()
    U = rng.random((N, N))
    A = np.triu((U < p).astype(float), k=1)
    return A + A.T  # symmetrize
```

A.3 Performance Metrics

We compute multiple complementary synchronization metrics:

Listing A.4: Metric calculation

```
def compute_metrics(t, theta, r, threshold=0.9):
    """
    Compute synchronization performance metrics.

    Returns dict with:
        - t_sync: time to reach r >= threshold
        - phase_variance: circular variance at t_final
        - convergence_rate: exponential decay rate
    """
    metrics = {}

    # Time to synchronization
    sync_indices = np.where(r >= threshold)[0]
    metrics['t_sync'] = (t[sync_indices[0]]
                         if len(sync_indices) > 0
                         else None)

    # Circular phase variance
    final_phases = theta[-1, :]
    metrics['phase_variance'] = (
        1 - np.abs(np.mean(np.exp(1j * final_phases)))
    )

    # Exponential convergence rate
    # Fit log(1-r) vs t in second half
    midpoint = len(r) // 2
    if r[midpoint] > 0.5:
        remainder = 1 - r[midpoint:]
        valid = remainder > 0.001
        if np.any(valid):
            coeffs = np.polyfit(t[midpoint:][valid],
                                np.log(remainder[valid]),
                                1)
            metrics['convergence_rate'] = -coeffs[0]

    return metrics
```