Developing a Parallel Machine Learning Approach for Network Predictions

Nolan J. Coble

Research was performed as part of the TREND 2019 REU at the University of Maryland, College Park, under the direction of Dr. Michelle Girvan, Dr. Ed Ott, and Dr. Thomas Antonsen.

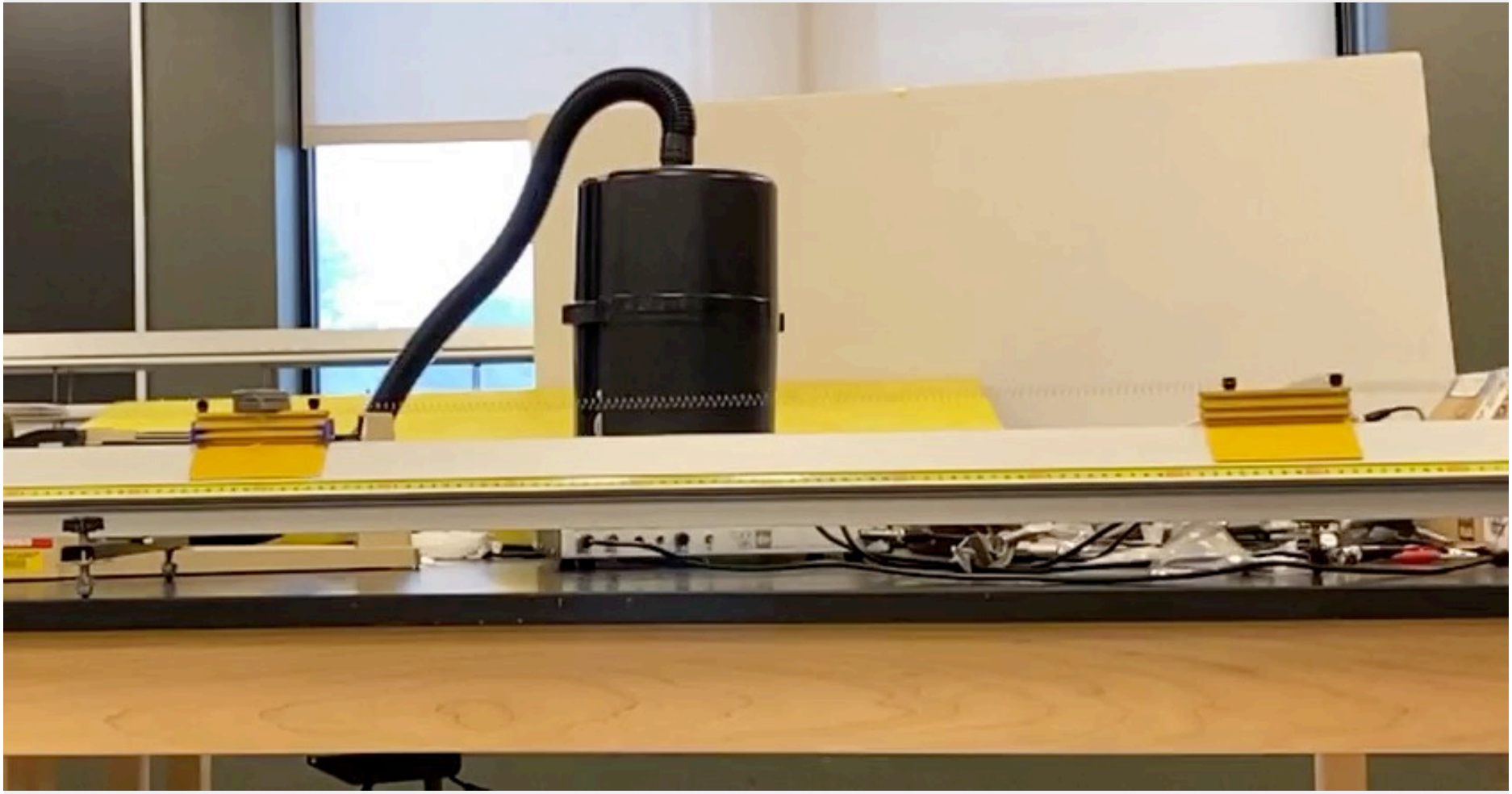
Things to pay attention to…

- Don’t focus too much on the inner workings of the machine learning device; the important part is **what data the device is being trained on and what data the device produces as predictions**.
- Don’t focus too much on the inner workings of the machine learning device! Neural networks are “black box” prediction techniques, **no one *really* knows why they work so well**.
- Focus on what a **network system** is, and how we can use it to our advantage to **develop a parallel prediction method**.

Beginning with an example…

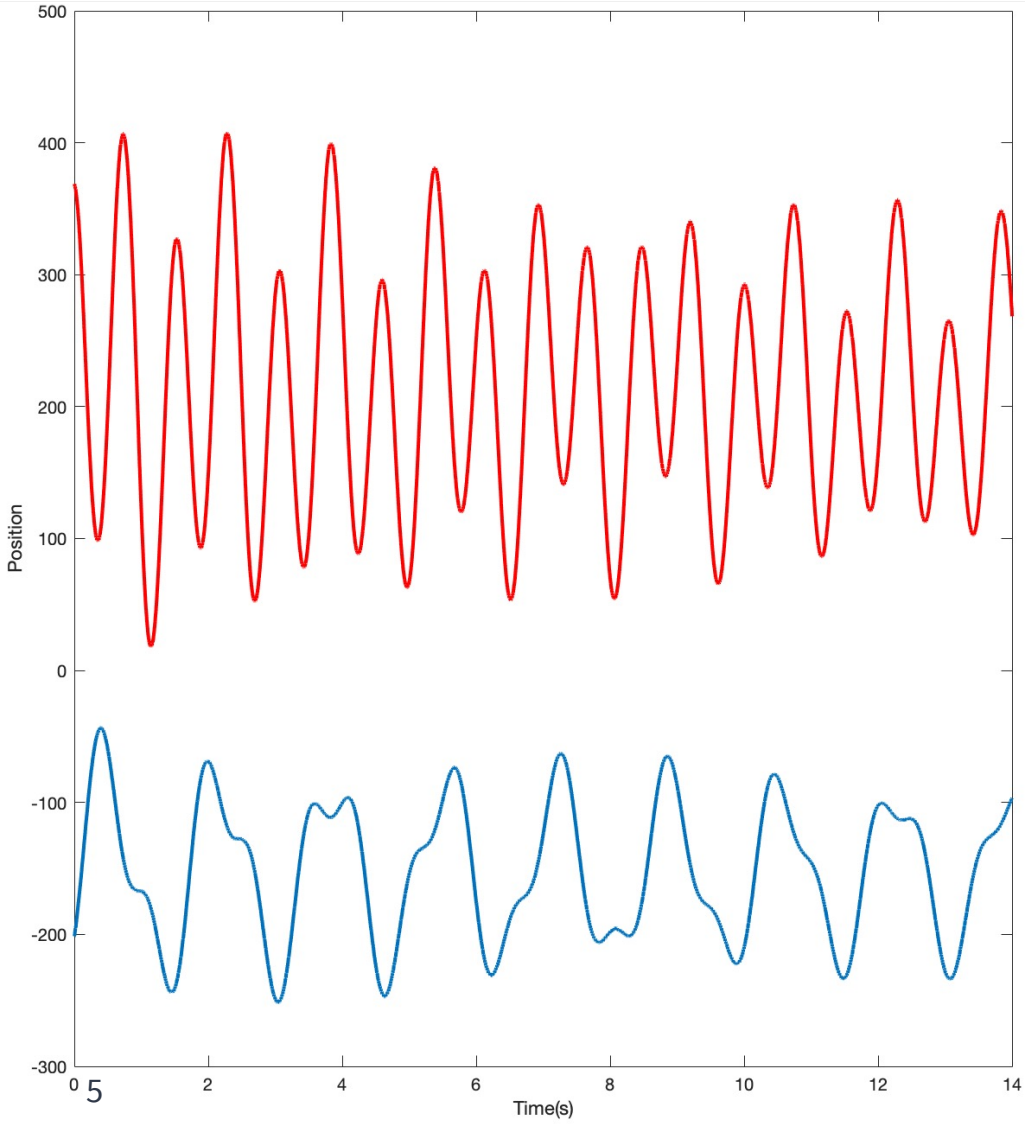
Predicting a 2-mass, 3-spring system

Predicting a 2-mass, 3-spring system



Predicting a 2-mass, 3-spring system

Problem: Given 14 seconds of data for a 2-mass, 3-spring system, determine how the system will continue to evolve in the future.



A physicist would typically solve this using theory. But we're not physicists today, we're computer scientists!

Solution: We will use machine learning!

Reservoir Computing

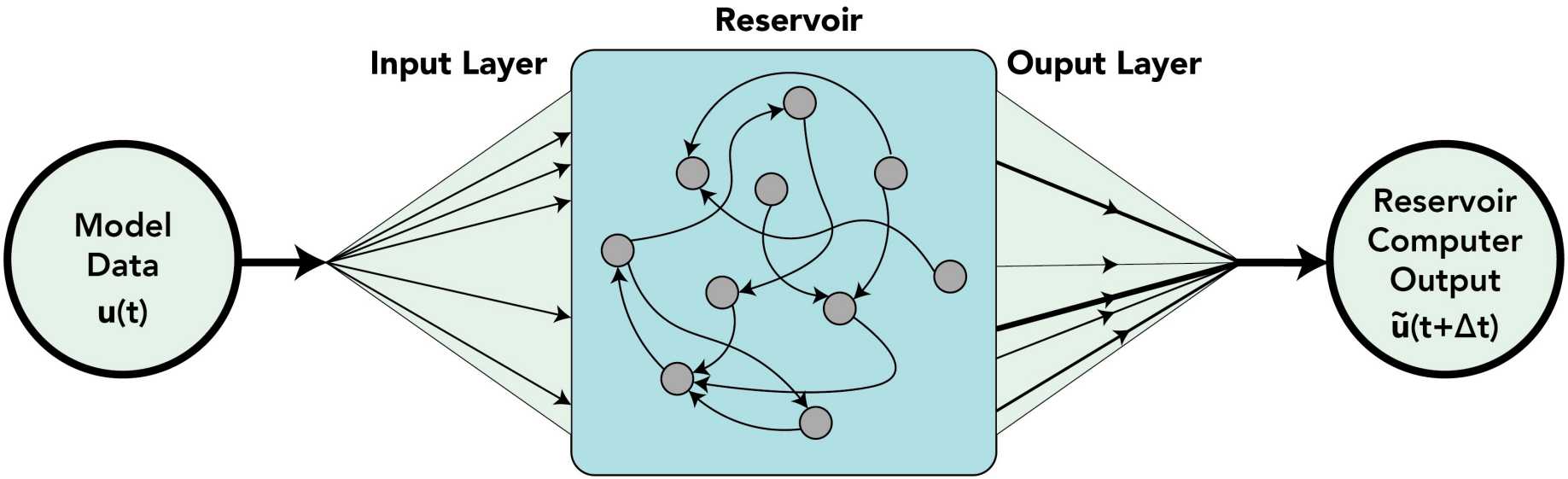
The machine learning technique we will be using.

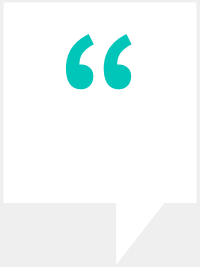
What is Reservoir Computing?

- Relatively new framework for **Recurrent Neural Networks**
  - RNN: Sparsely connected nodes, meant to model neuron behavior
- Much faster than other neural networks/ machine learning techniques
- Good at predicting oscillatory/bounded data

RC Breakdown

- System consists of three layers: **input**, **reservoir**, and **output**





- *Disclaimer:* I will not go through the mathematical theory behind reservoir computing; I will simply describe the framework and the process here.

Reservoir Computing Framework

- For each time step, we have a vector representing the **current state of the system**
- From this vector, we will generate a **corresponding reservoir state** via an *activation function*
- We extract an **output vector** from a reservoir state via an *output function*

Reservoir Computing Framework

- For each time step, we have a vector representing the **current state of the system**
- From this vector, we will generate a **corresponding reservoir state** via an *activation function*
- We extract an **output vector** from a reservoir state via an *output function*

NOTES

- Current state of the system will be a **low dimensional vector** (only 2 elements for our sample)
- The corresponding reservoir state will be a **high dimensional vector**. The activation function uses the current system state, and the previous reservoir state
- Output vector should be **exactly the same** as the next state of the system

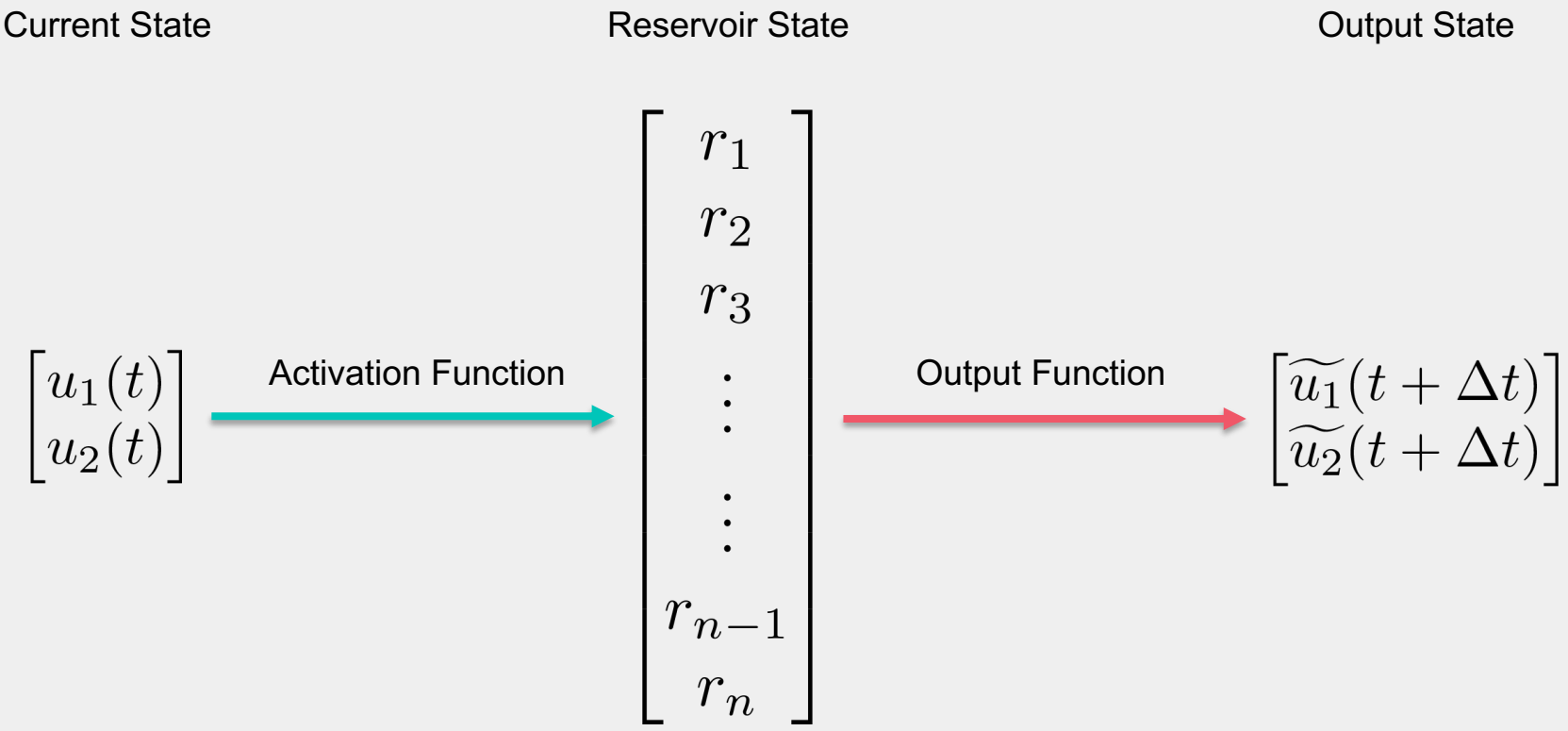
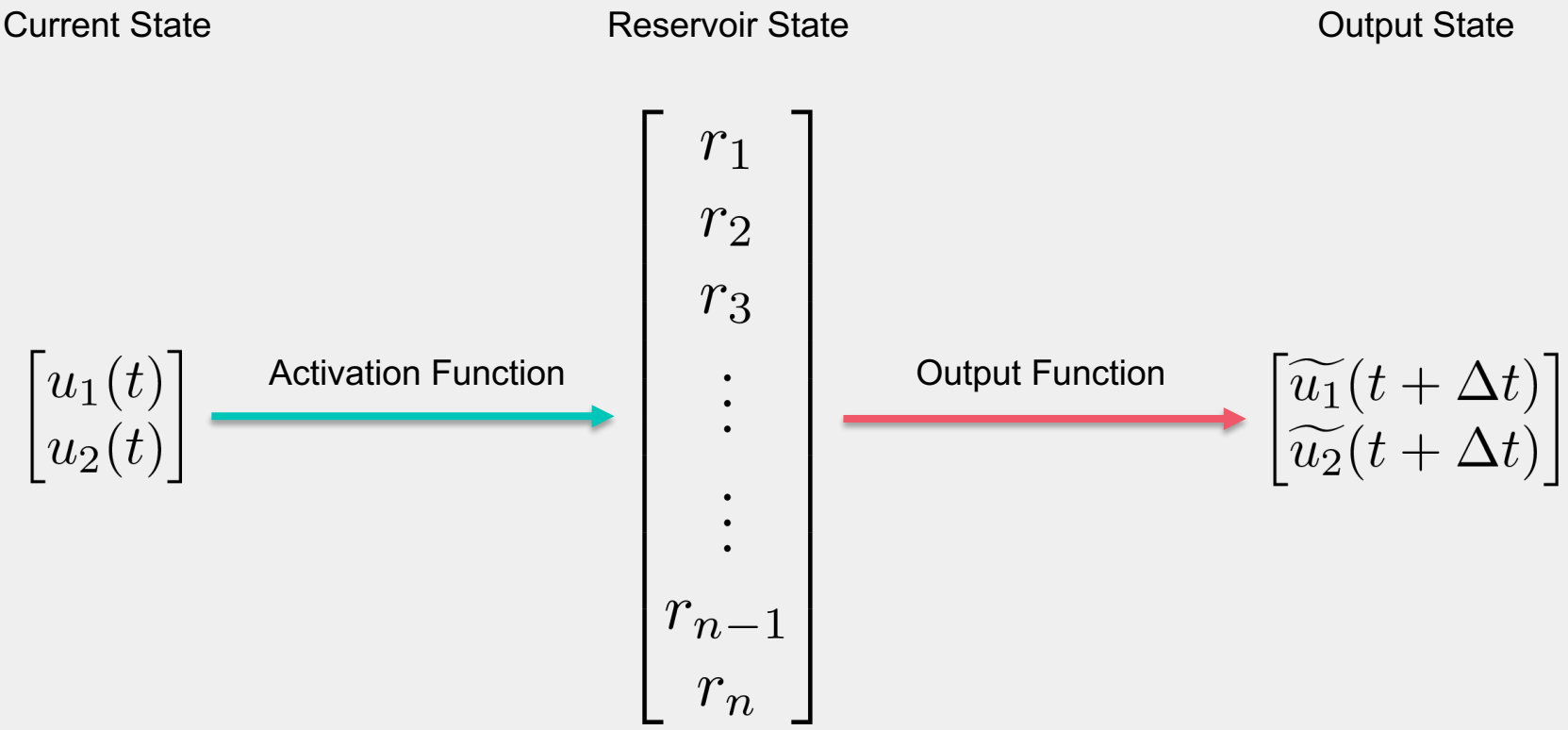
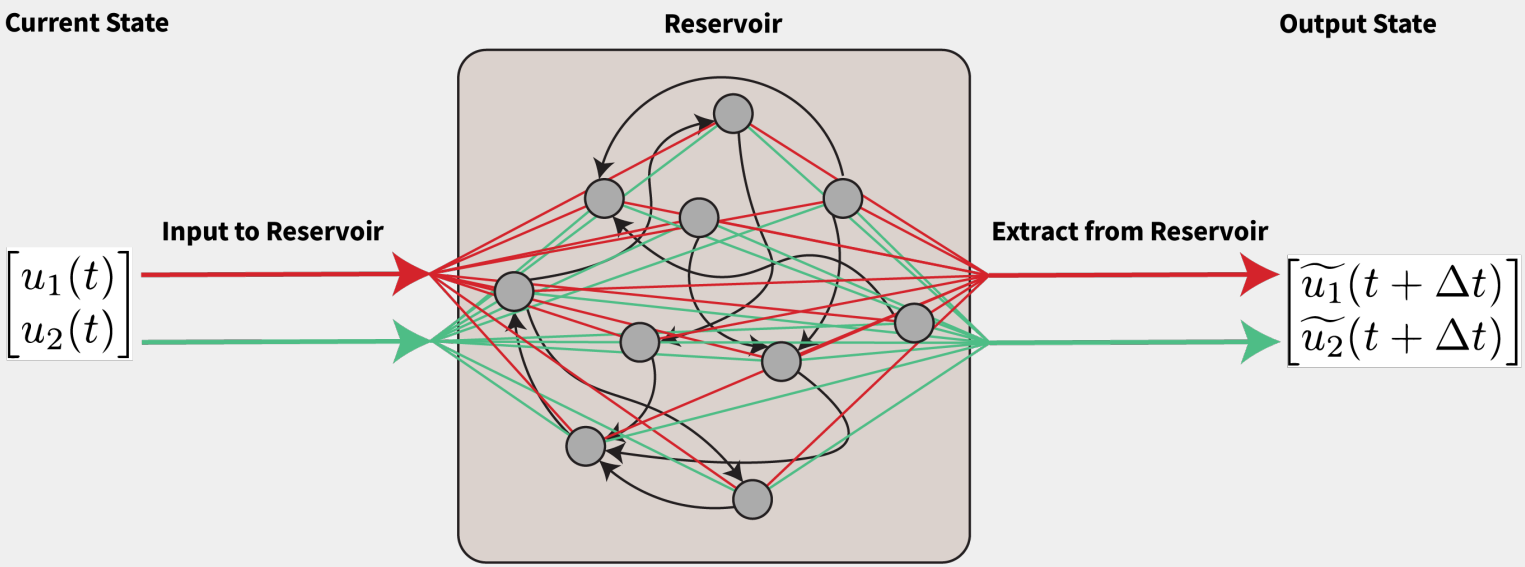


Diagram for reservoir computing (This is actually the basis of all neural networks)



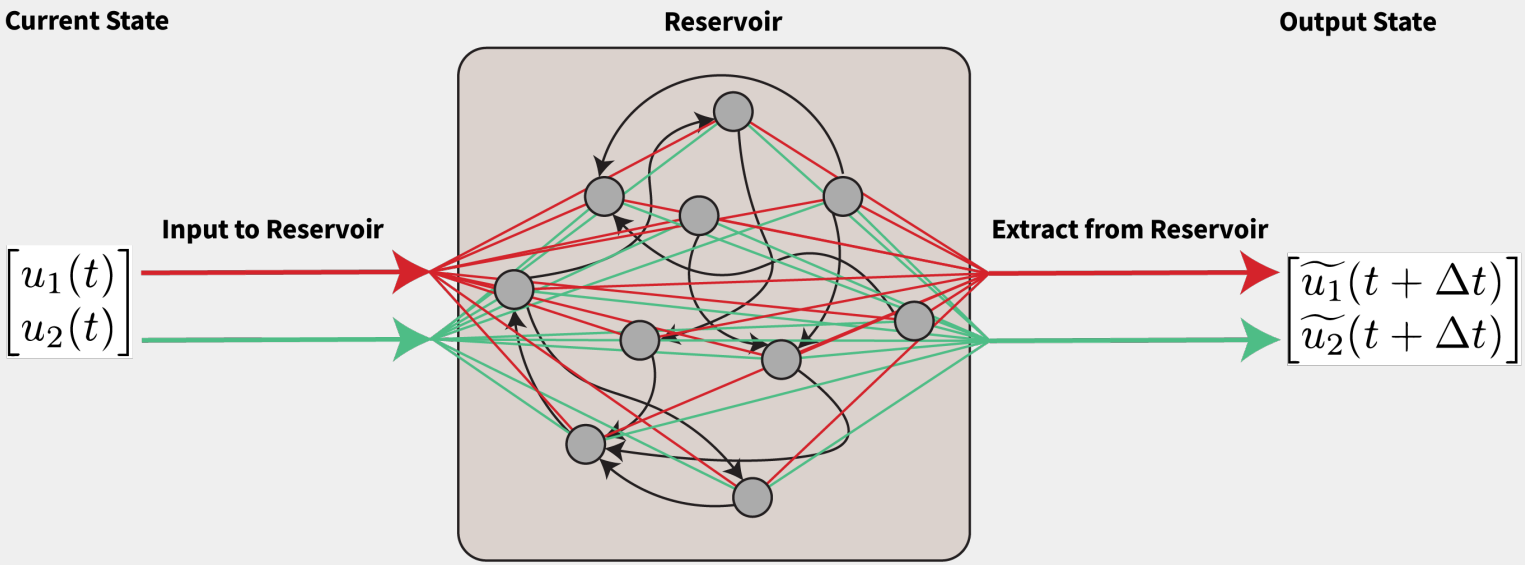
We refer to the dimension of the reservoir states as the **size of the reservoir**

Diagram for reservoir computing (This is actually the basis of all neural networks)



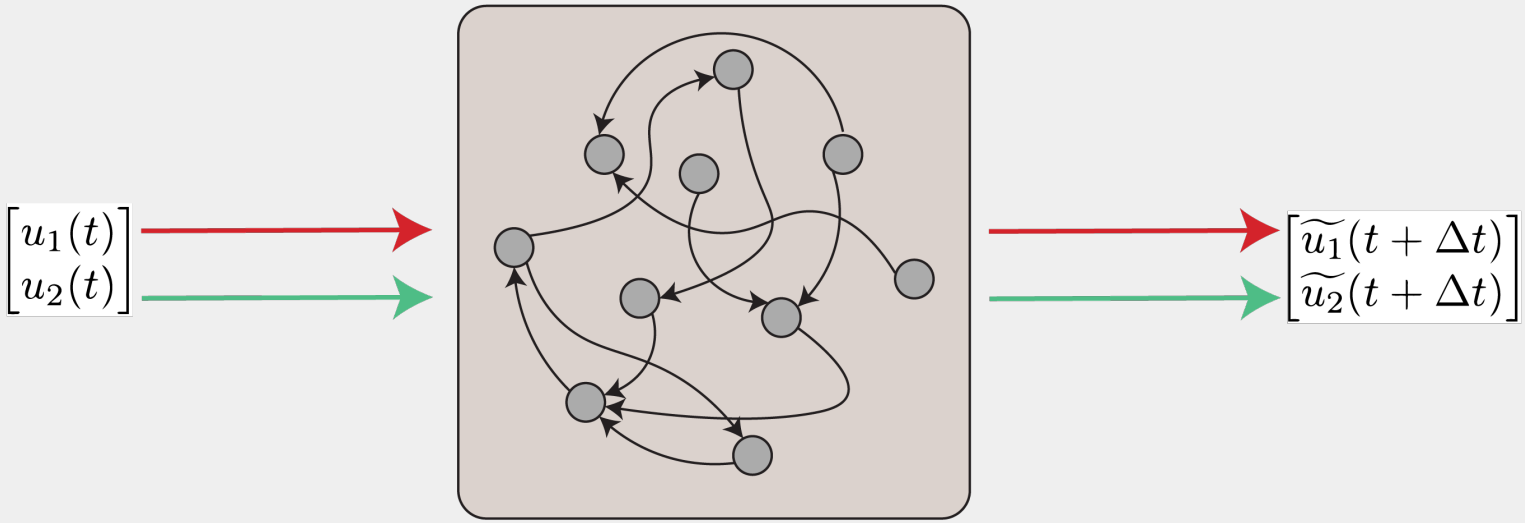
Reservoir computing diagram.

Note that each vector component is connected with every node of the reservoir.



Reservoir computing diagram.

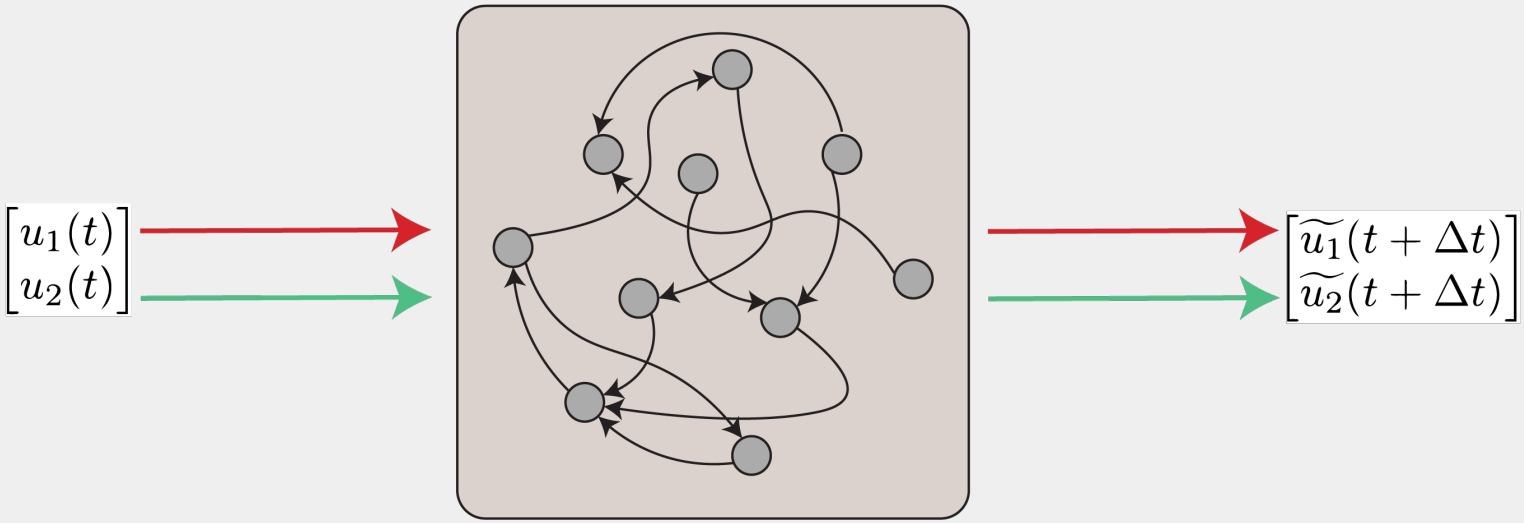
We will typically simply to this diagram. It is still implied that the arrows represent total connections to the reservoir nodes.



Reservoir computing diagram.

We will adopt the following terminology.

- The reservoir **receives training input from** oscillators 1 and 2
- The reservoir **outputs predictions for** oscillators 1 and 2



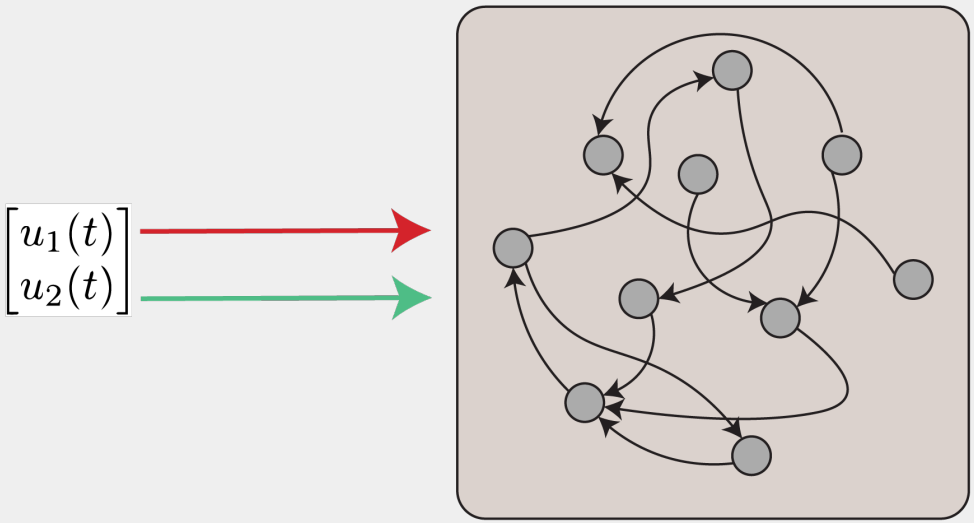
Reservoir computing diagram.

Reservoir Computing Process

Stages:
(1) Listening, (2) Training, (3) Predicting

(1) Listening: Building reservoir states one by one

- For each piece of training data, $u(t)$, we generate an associated reservoir state, $r(t)$.
- So, if we have 14 seconds worth (maybe 3000 points) of data for the 2-mass, 3-spring system, we will generate 14 seconds worth of corresponding reservoir states.



(2) Training: Generating an output matrix

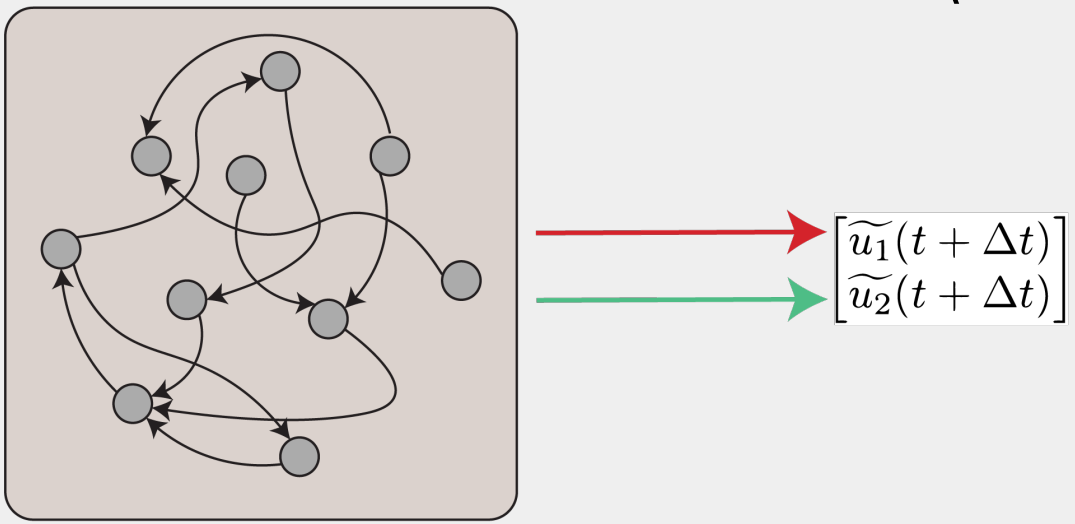
- We will extract information from a reservoir state $r(t)$ via the following equation:

$$H * r(t) = \tilde{u}(t + \Delta t)$$

- For each of our 3000 data points $\vec{u}$, we have 3000 reservoir states $\vec{r}$. Since these reservoir states supposedly correspond to the input states, we expect the following:

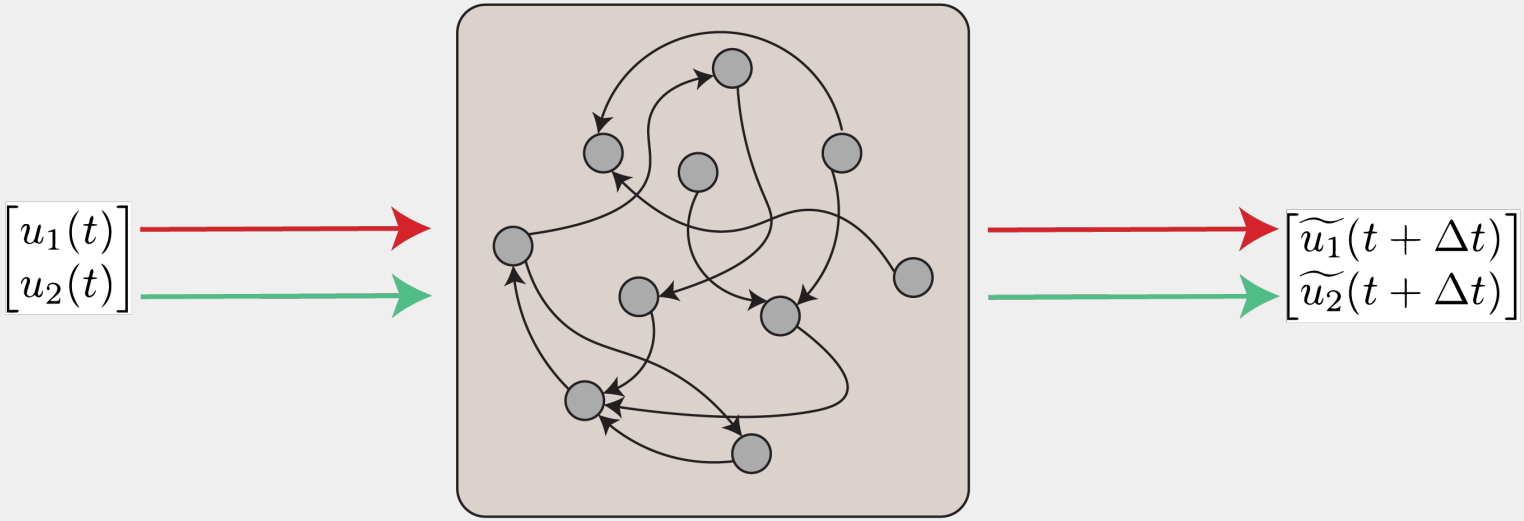
$$H * \vec{r} = \vec{u}$$

- We will **train** an output matrix doing the following linear regression: $H = \vec{u} \backslash \vec{r}$



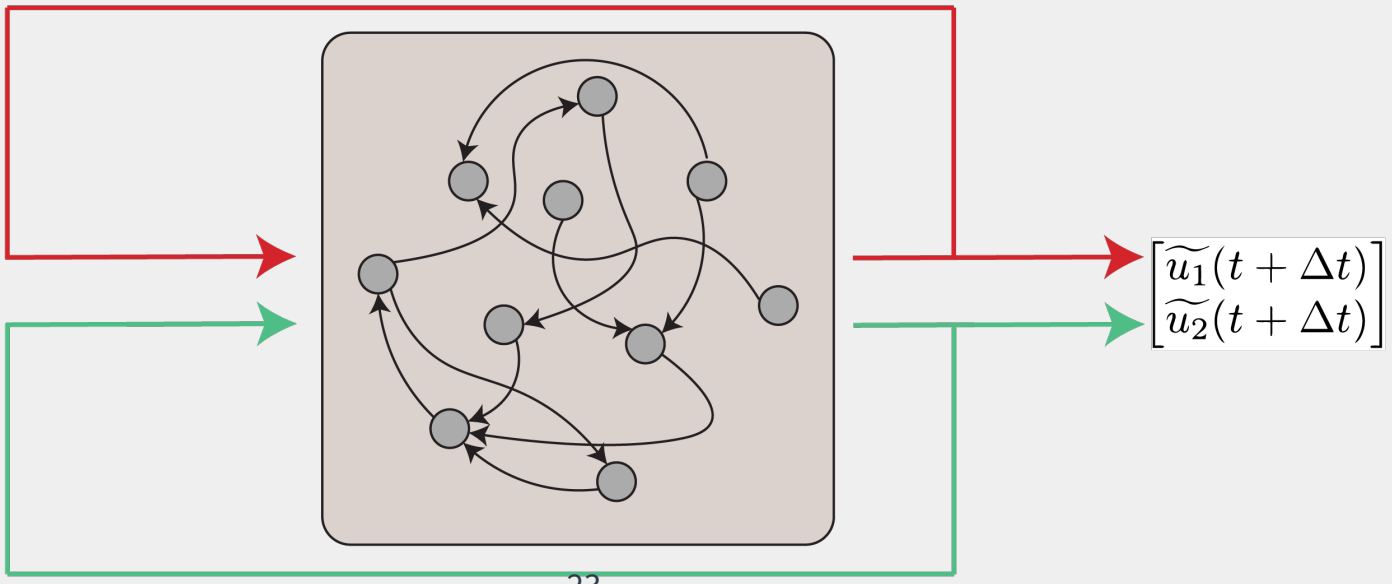
(1)(2) Training

- Typically, we will refer to the listening and training stages together as a single training stage.
- We note the terminology :
  - The reservoir **receives training input from** oscillators 1 and 2



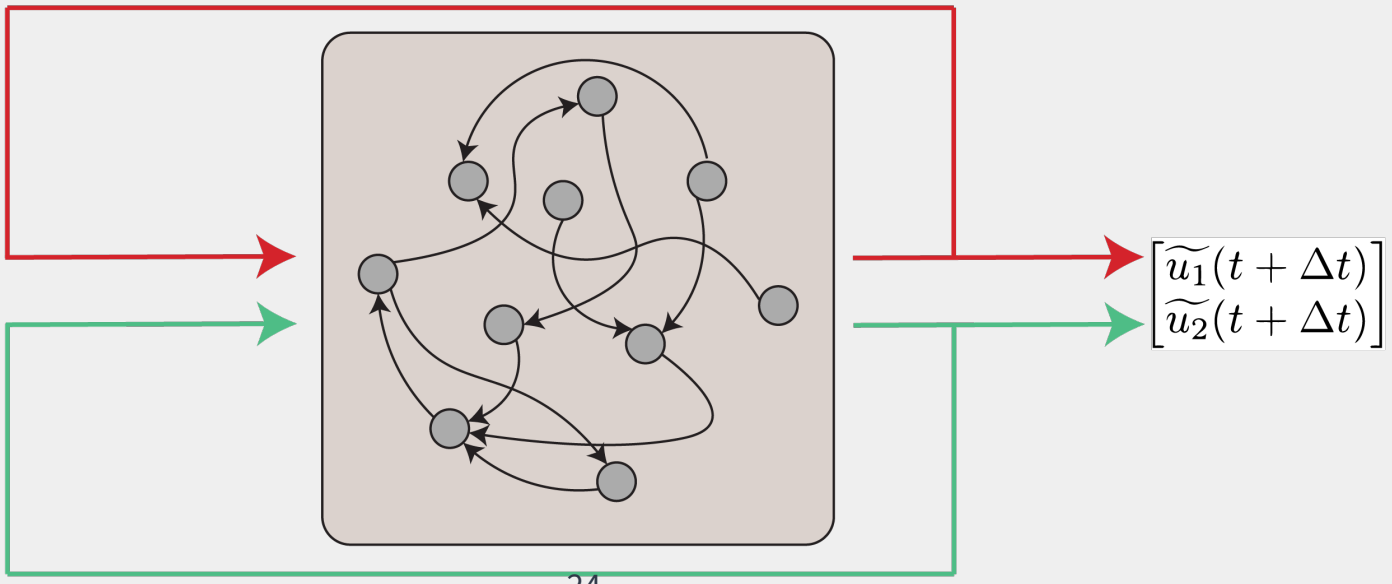
(3) Predicting: Using output states as input

- Once we have trained our output matrix H, we are ready to predict.
- Essentially, we generate a new reservoir state from our previous output $u(t + \Delta t)$.
- When we extract this from the reservoir, we obtain $H * r(t + \Delta t) = u(t + 2\Delta t)$, which is a prediction for the next state of our system



(3) Predicting: Using output states as input

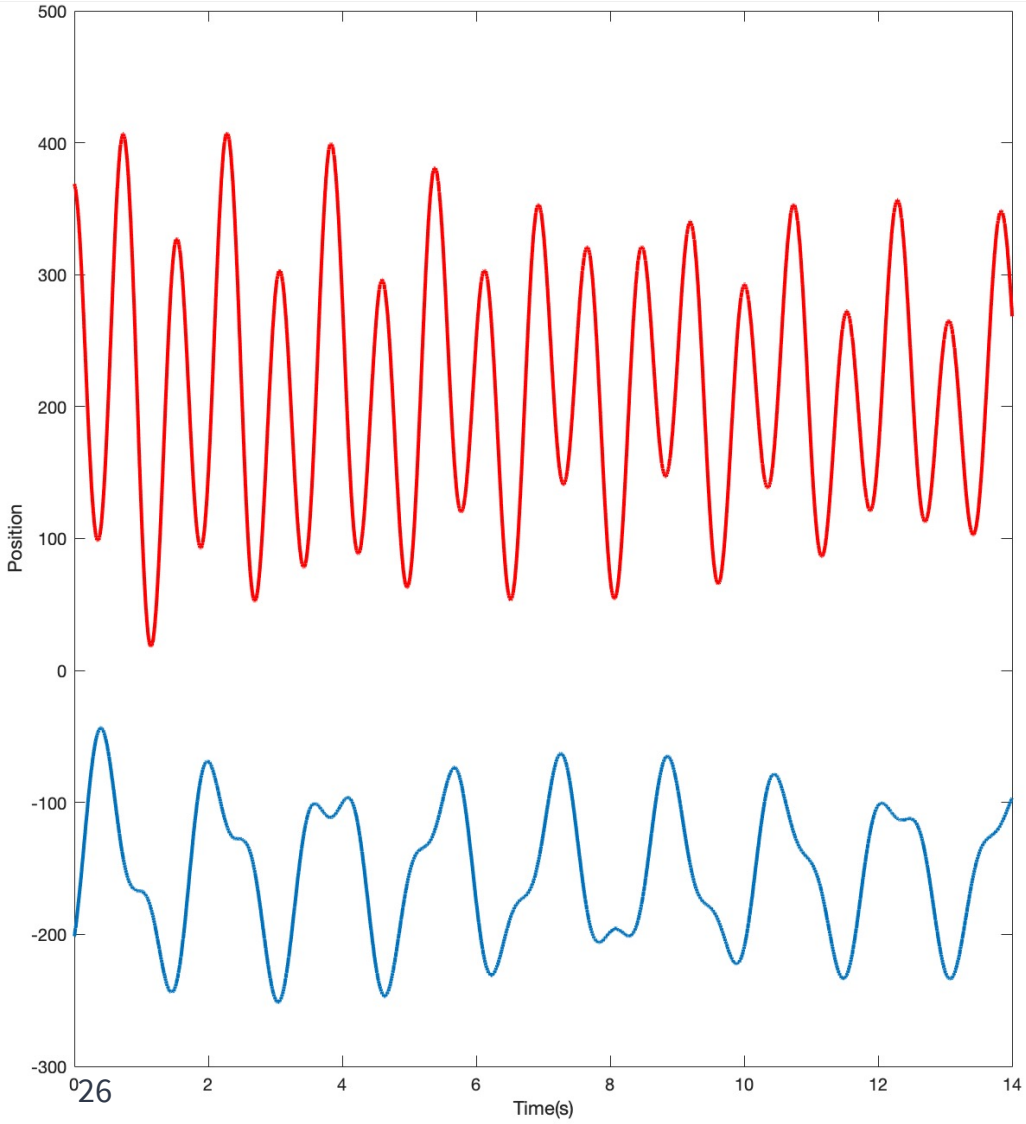
- We note the terminology:
  - The reservoir **outputs predictions** for oscillators 1 and 2
  - The reservoir **receives prediction input from** the predictions for oscillators 1 and 2



Back to the example…

Predicting a 2-mass, 3-spring system

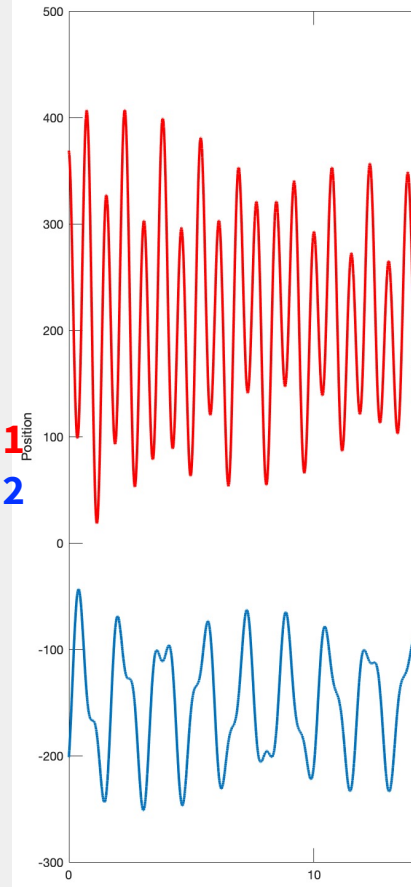
Problem: Given 14 seconds of data for a 2-mass, 3-spring system, determine how the system will continue to evolve in the future.



Predicting a 2-mass, 3-spring system

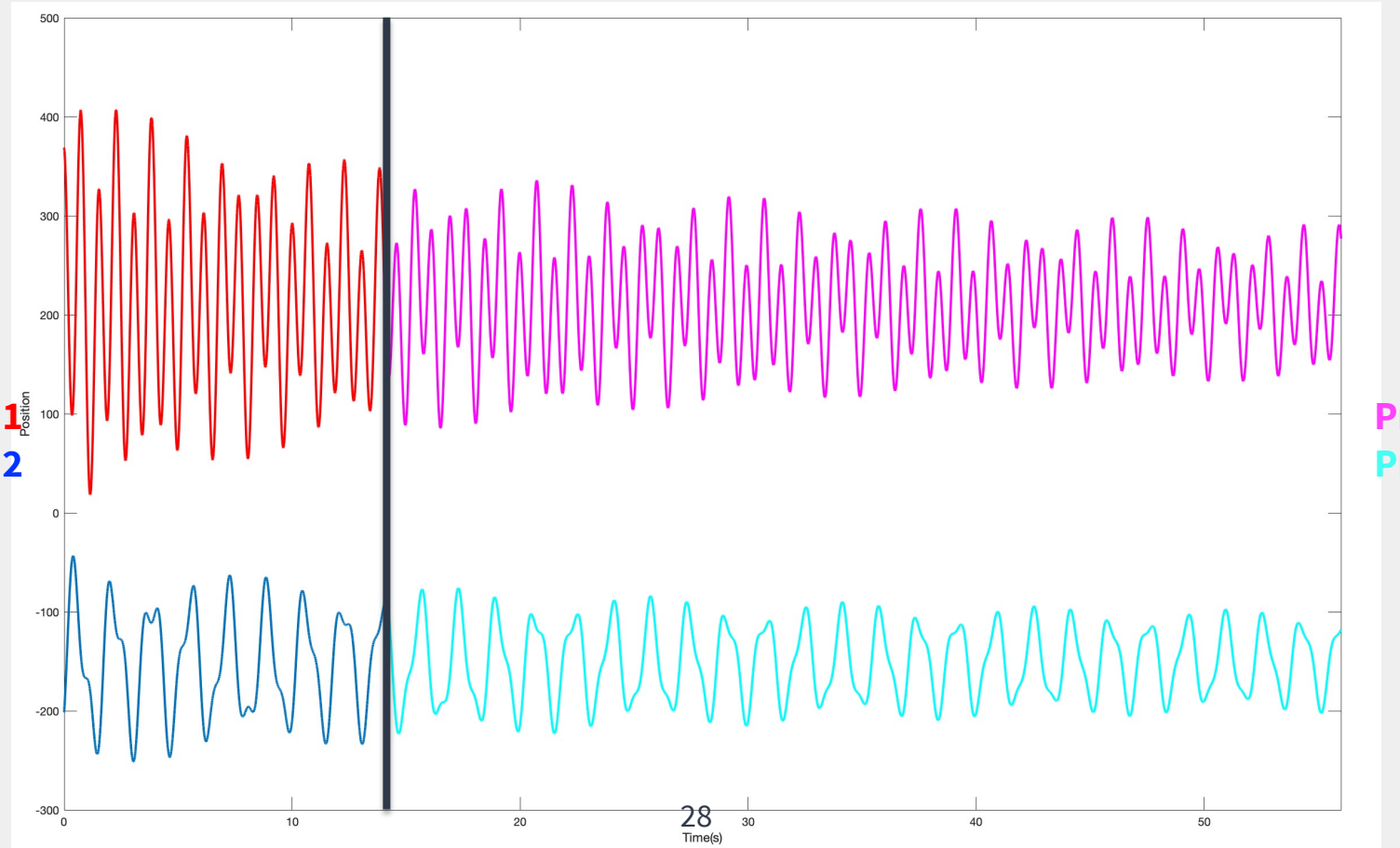
- **Training stage:** input all 14 seconds of data to the reservoir, and obtain the output matrix

Oscillator 1
Oscillator 2



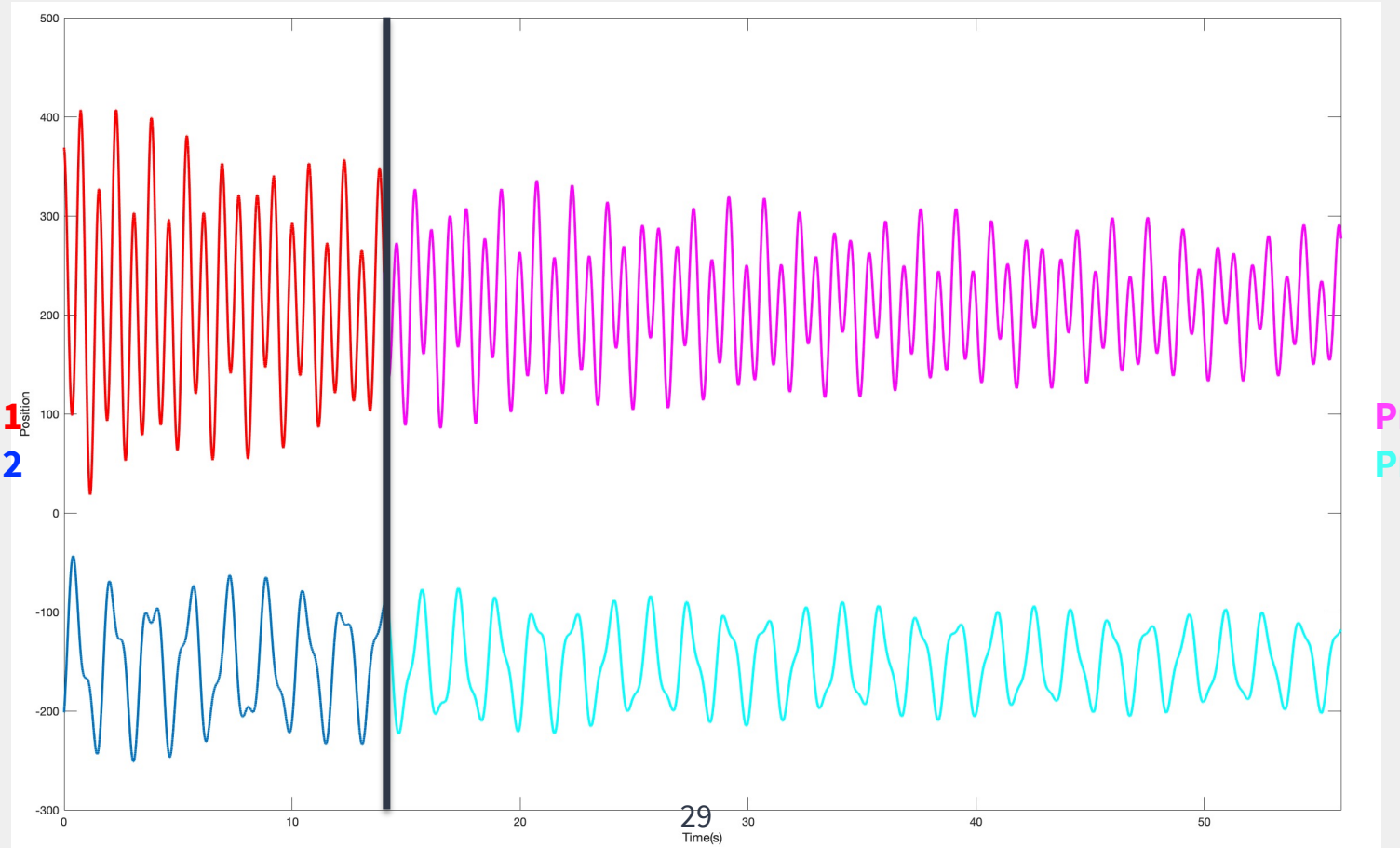
Predicting a 2-mass, 3-spring system

- **Predicting stage:** use the reservoir output as prediction input



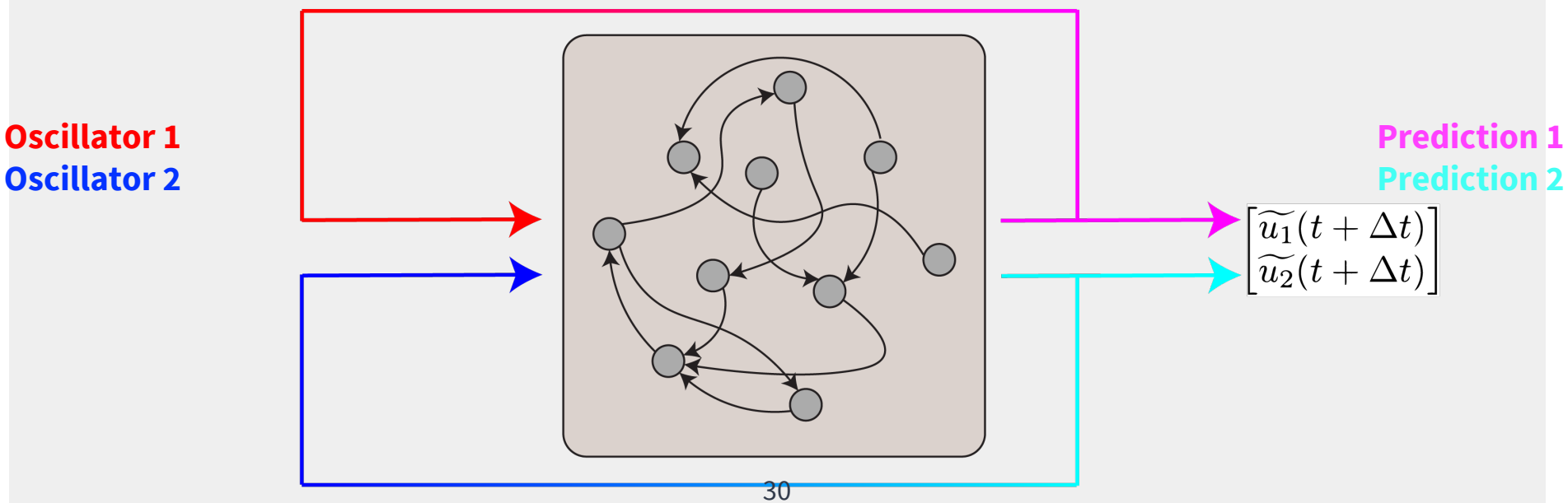
Predicting a 2-mass, 3-spring system

- The reservoir has no knowledge of the system data, it is evolving on its own



Predicting a 2-mass, 3-spring system

- The reservoir has no knowledge of the system data, it is evolving on its own



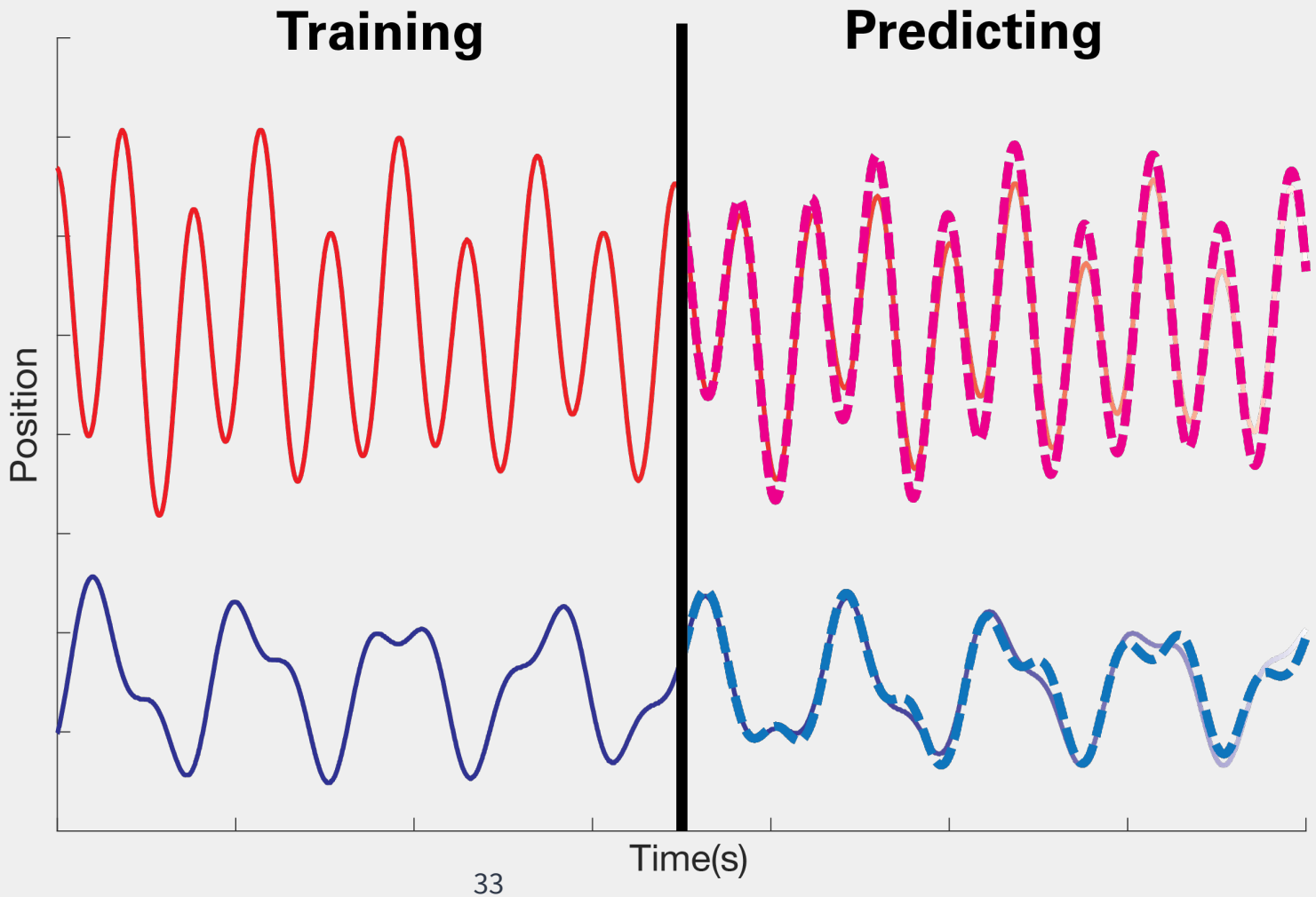
The result appears to be a viable solution, but how can we be sure it is a good prediction?

To see how accurate predictions are, we will use only 50% of our data for training. The other 50% will be reserved for validation.

Predicting a 2-mass, 3-spring system

Dotted lines represent predictions.

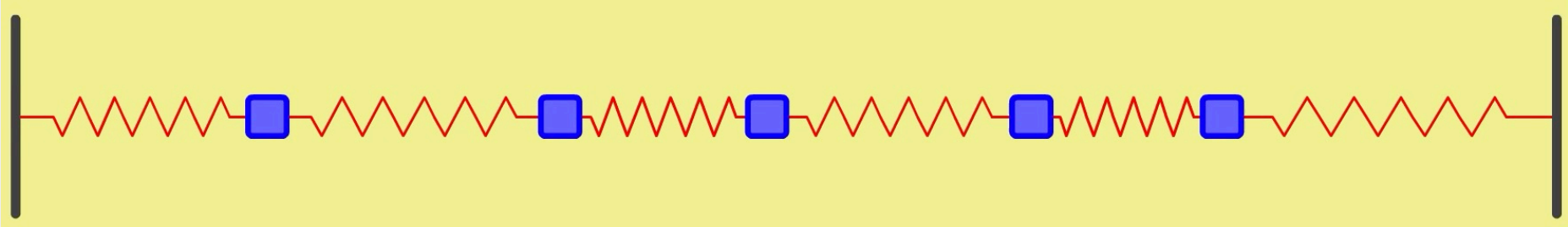
For the most part, the predictions accurately represent the true data.



How might the accuracy be affected by looking at a bigger system?

Predicting a 5-mass, 6-spring system

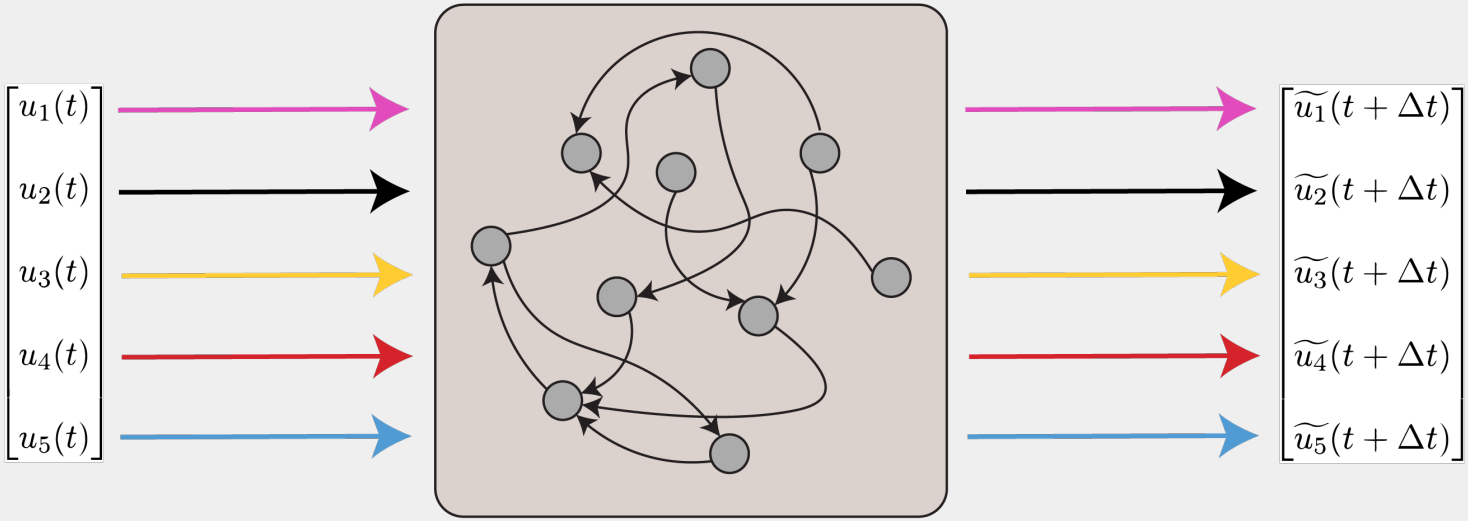
Will we still be able to predict this?



Predicting a 5-mass, 6-spring system

Training Framework

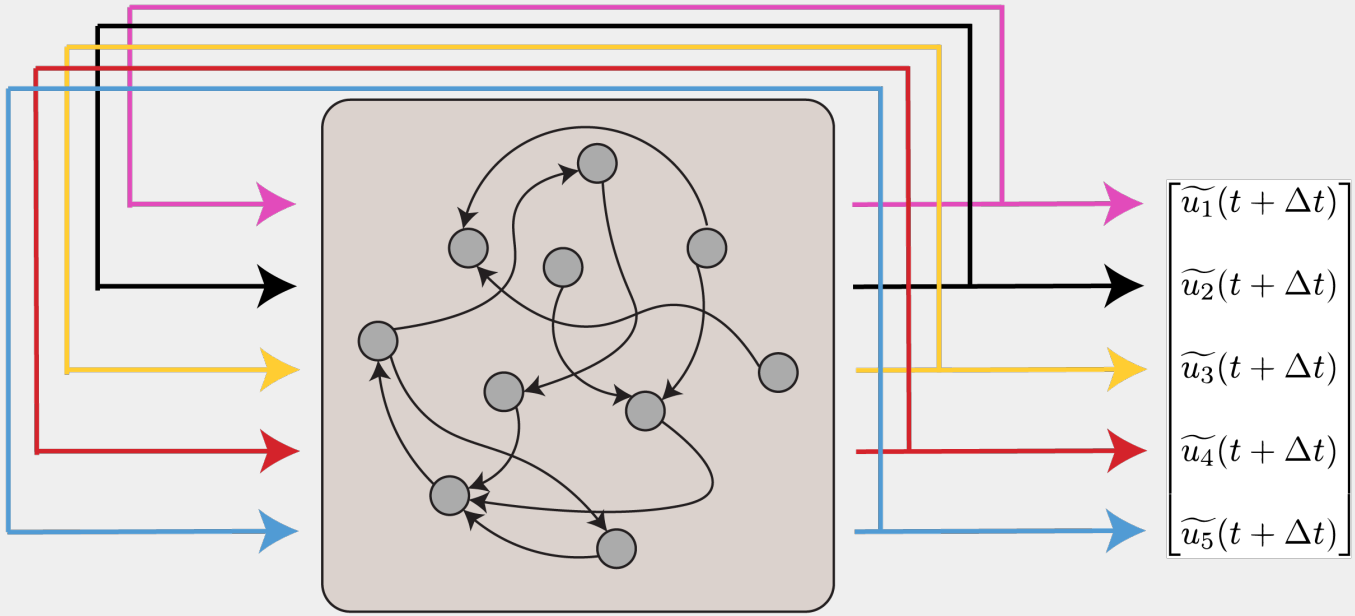
- We note the terminology:
  - The reservoir **receives training input from** oscillators 1, 2, 3, 4, and 5



Predicting a 5-mass, 6-spring system

Predicting Framework

- We note the terminology:
  - The reservoir **outputs predictions** for oscillators 1, 2, 3, 4, and 5
  - The reservoir **receives prediction input from** the predictions for oscillators 1, 2, 3, 4, and 5

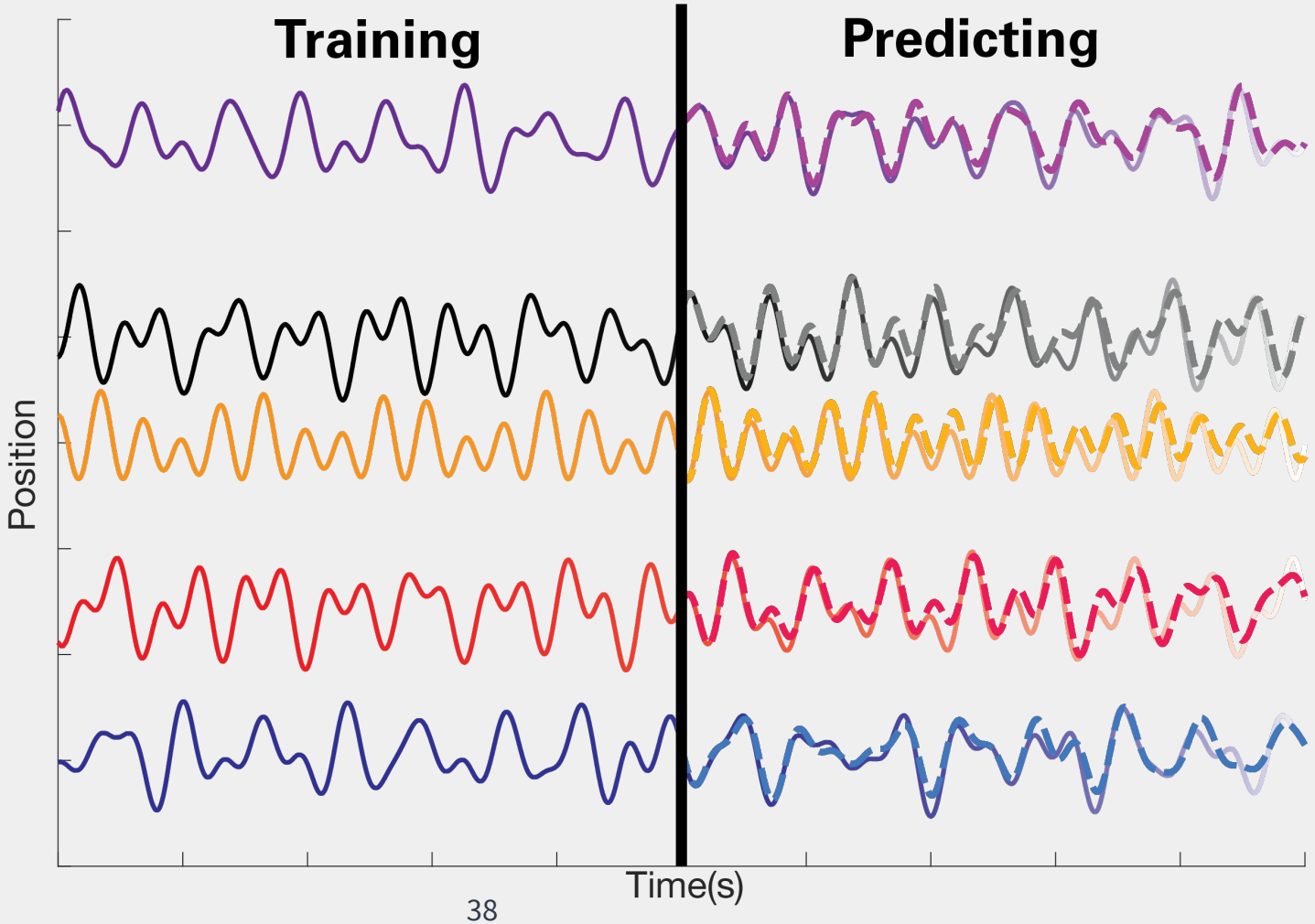


Predicting a 5-mass, 6-spring system

Dotted lines represent predictions.

For the most part, the predictions accurately represent the true data.

Not as accurate as the 2-mass



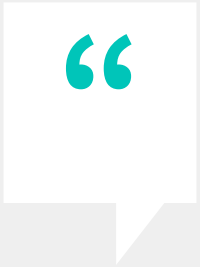
Still accurate, but as the system gets bigger we would see a breakdown in the solutions.

What do I mean?

- We use a reservoir because it is a **high-dimensional analogue** of our low-dimensional system.
- If we constantly increase the size of our system, we will have to **scale the reservoir** accordingly.
- If computer resources were unlimited this wouldn’t matter, but computer resources **are limited**.

The fix…

We will have to develop a parallel prediction method.



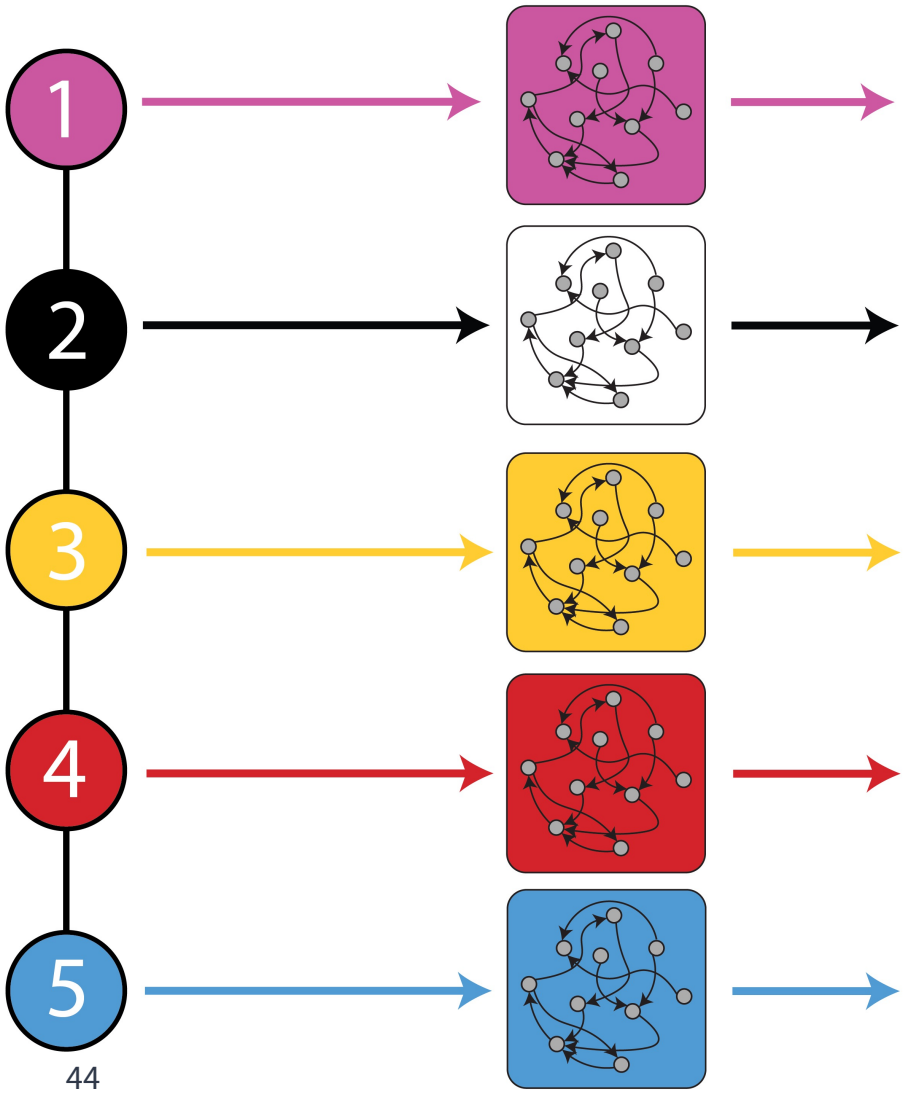
- We will take advantage of the fact that the masses **are not independent** systems, they are **coupled together**.
- In fact, these mass-spring systems are examples of **network-coupled dynamical systems**.

Developing a Parallel Method — naïve solution

- The 5 masses are connected together as shown to the right (this is the **network** that describes them)
- Instead of using one very large reservoir to predict the system, we will **assign each node** a smaller reservoir

Developing a Parallel Method — naïve solution

- Naïve solution: each reservoir trains and predicts a single node.
- In our terminology, the i^{th} reservoir **receives training and prediction input** from the i^{th} node.
- And, the i^{th} reservoir **outputs predictions** for the i^{th} node.

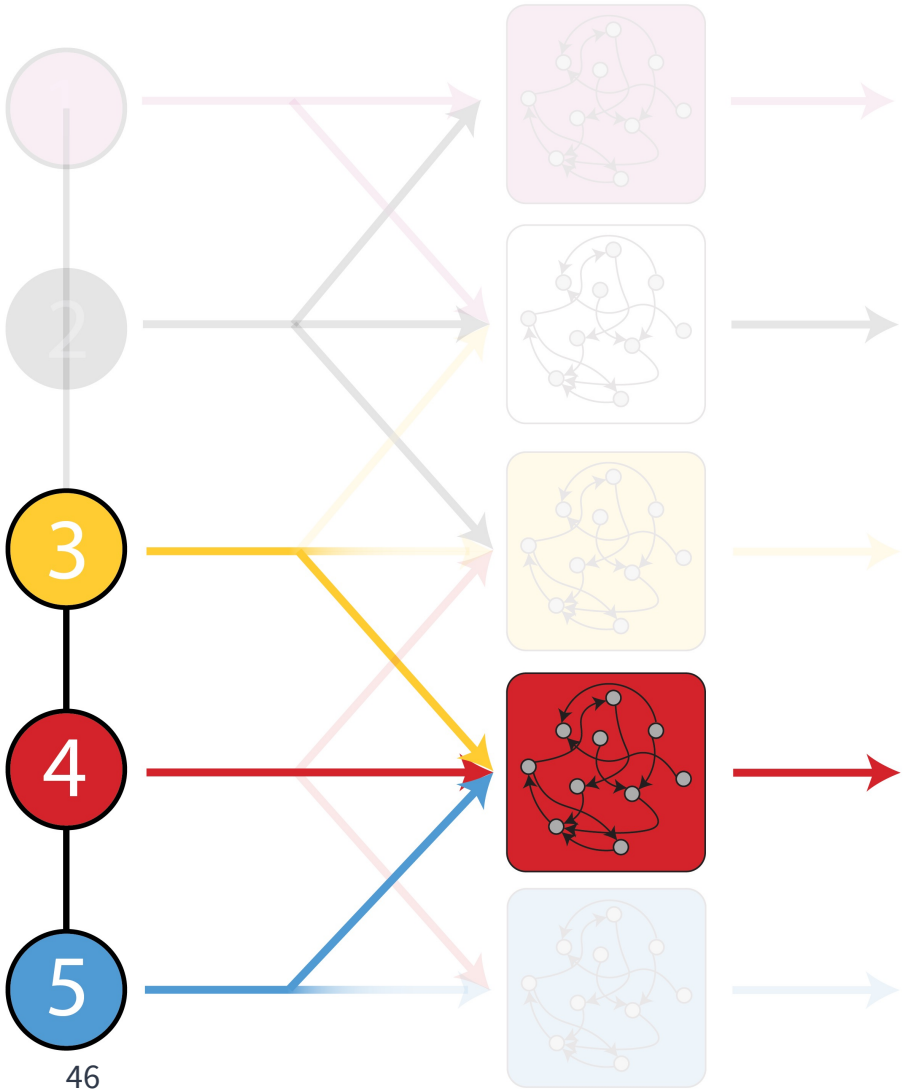


This solution **does not** work since the masses affect each other.

We **need** to make use of the network connections.

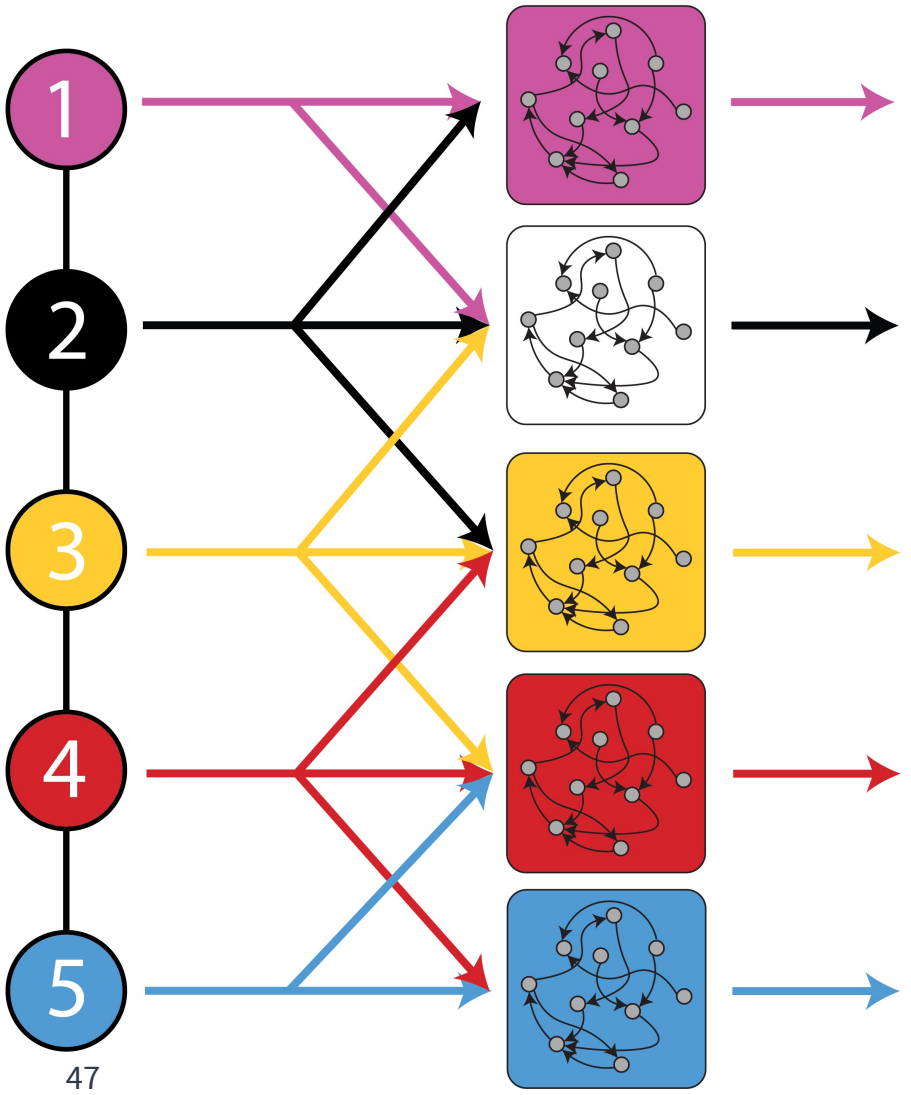
Developing a Parallel Method — better solution

- We assign one reservoir to every node.
- Each reservoir will **train on its own node and that node's neighbors** but will only **predict its own node**.
- i.e. the 4^{th} reservoir **receives training and prediction input** from the 4^{th} node and 3^{rd} and 5^{th} nodes.
- The 4^{th} reservoir **outputs predictions** for the 4^{th} node only.



Developing a Parallel Method — better solution

- We assign one reservoir to every node.
- Each reservoir will **train on its own node and that node's neighbors** but will only **predict its own node**.
- i.e. the i^{th} reservoir **receives training and prediction input** from the i^{th} node and that node's neighbors.
- The i^{th} reservoir **outputs predictions** for the i^{th} node only.



We now look at network systems…

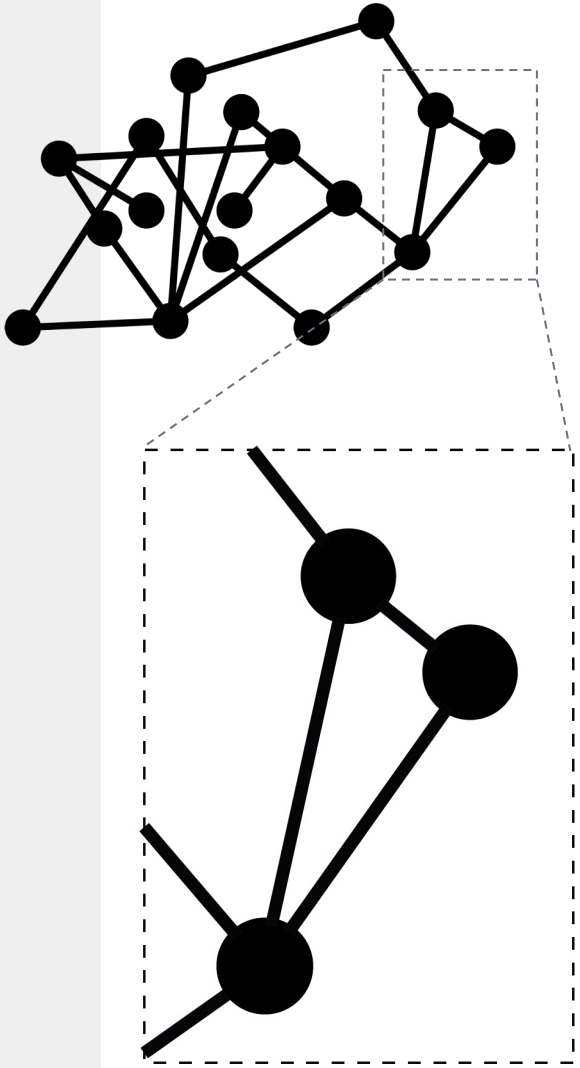
First generic network-coupled dynamical systems,
then a more concrete example.

Network-Coupled Dynamical Systems

- By this, I mean individual dynamical systems that affect one another.
- Common examples: mass-spring systems, neurons, the weather, literally every dynamical system with multiple pieces can be thought of as network-coupled.
- The important thing to note: the most apparent network systems are **spatial**, but many network examples **are not**.

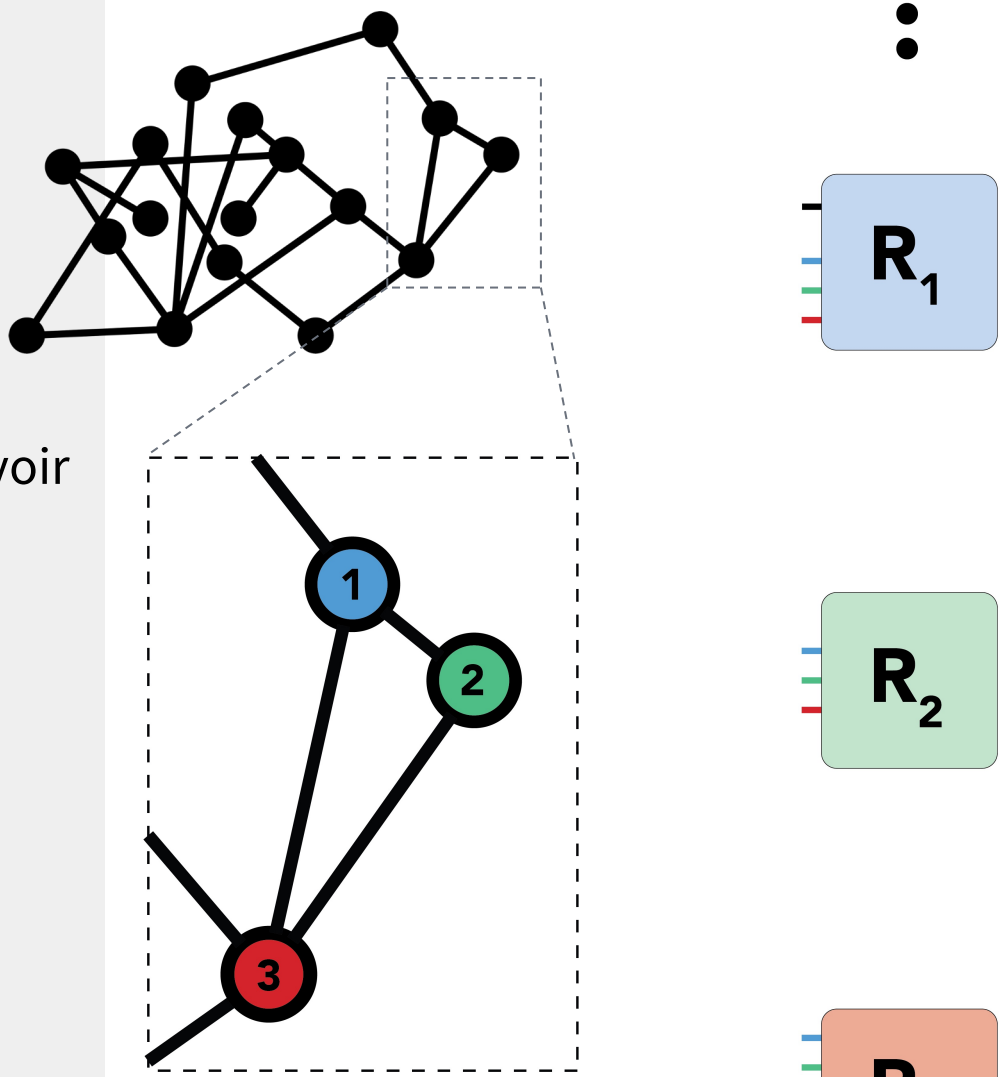
Training Framework

Parallel Reservoir Training Framework



Parallel Reservoir Training Framework

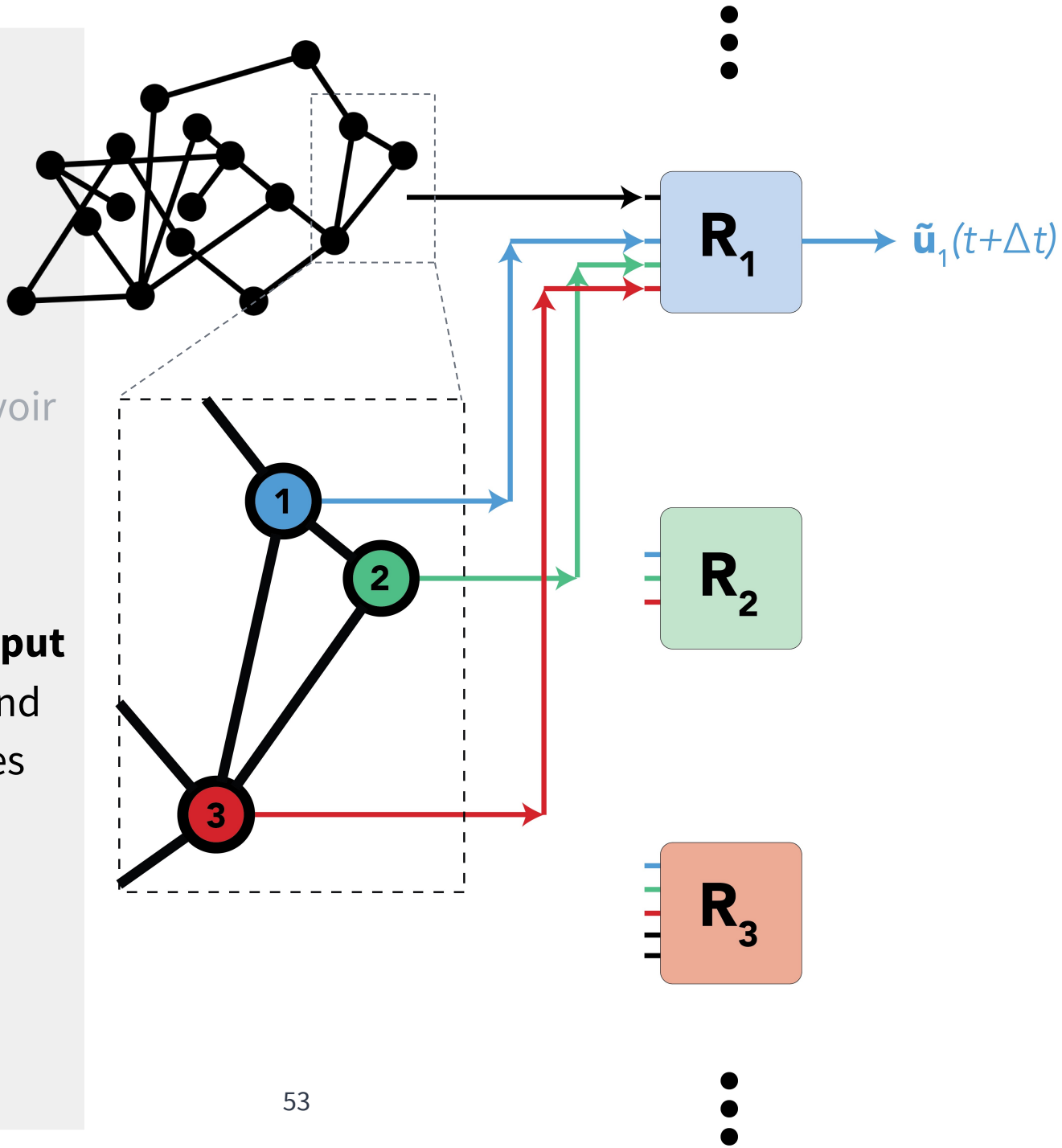
1. Assign a small reservoir to each node



Parallel Reservoir Training Framework

1. Assign a small reservoir to each node

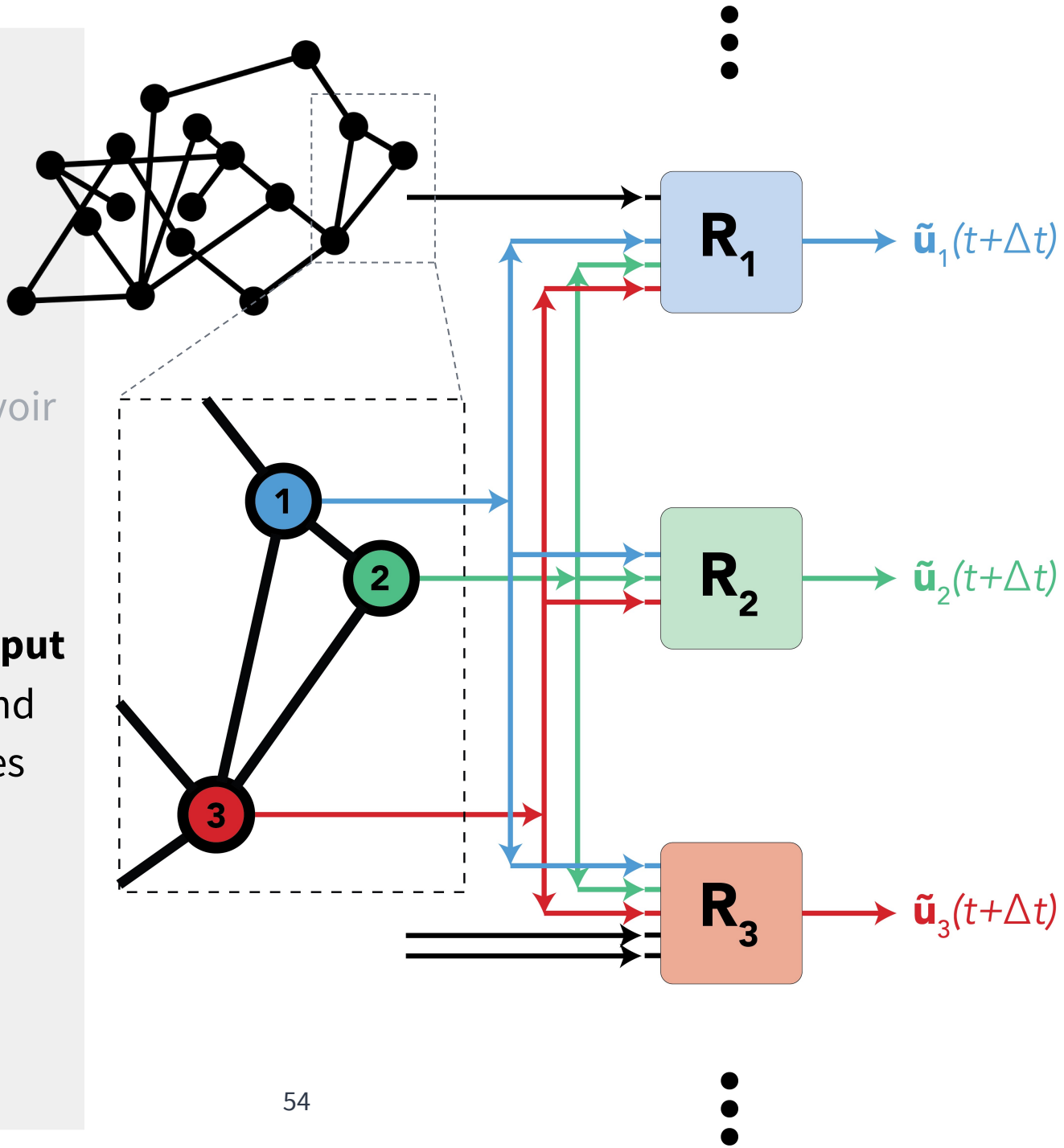
2. The 1^{st} reservoir **receives training input from** the 1^{st} node and its neighboring nodes



Parallel Reservoir Training Framework

1. Assign a small reservoir to each node

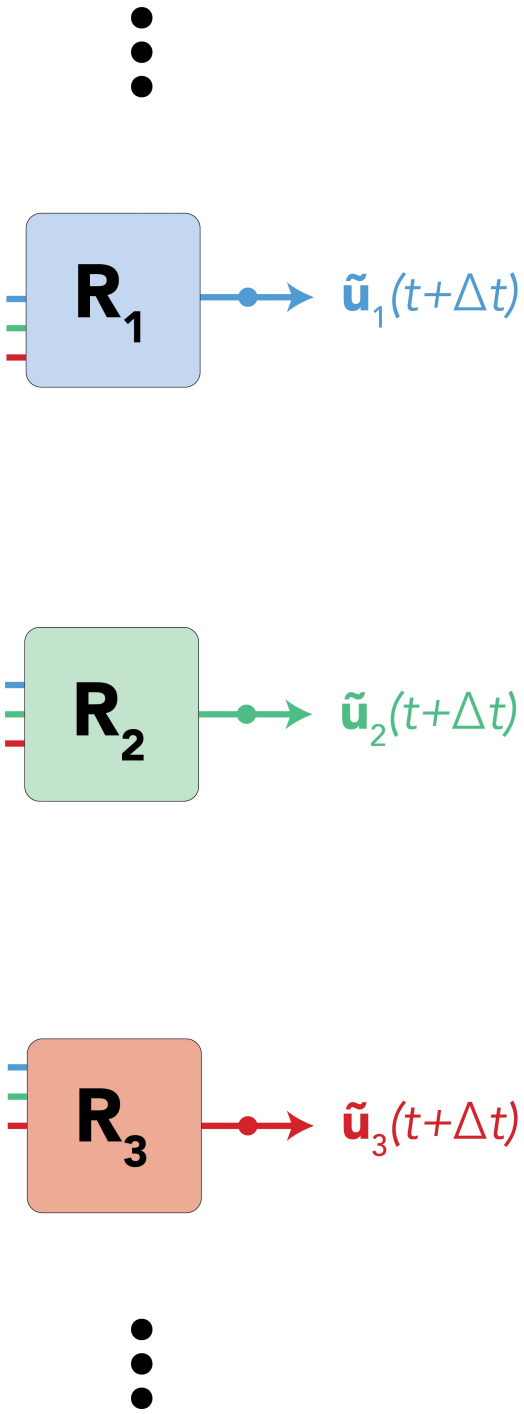
2. The i^{th} reservoir **receives training input from** the i^{th} node and its neighboring nodes



Predicting Framework

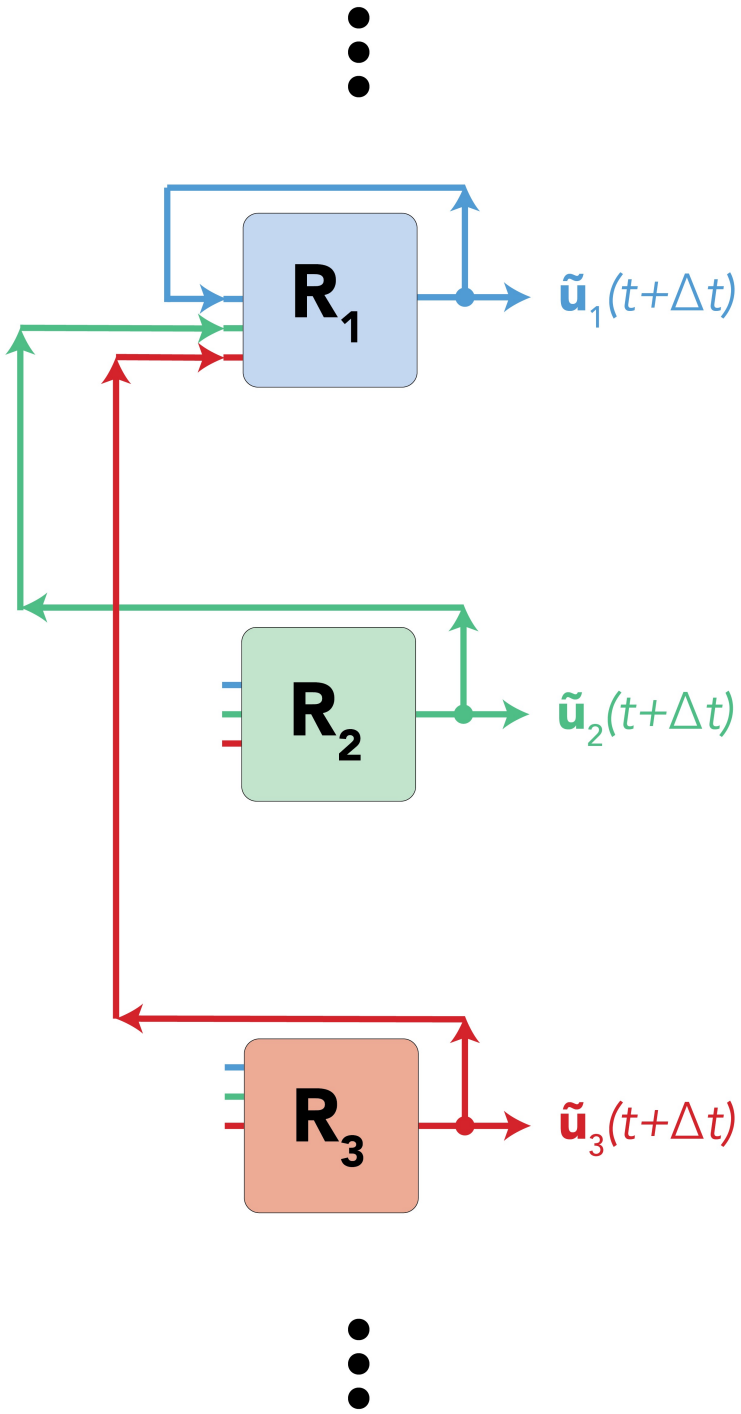
Parallel Reservoir Prediction Framework

1. The i^{th} reservoir **outputs predictions for** the i^{th} node **only**



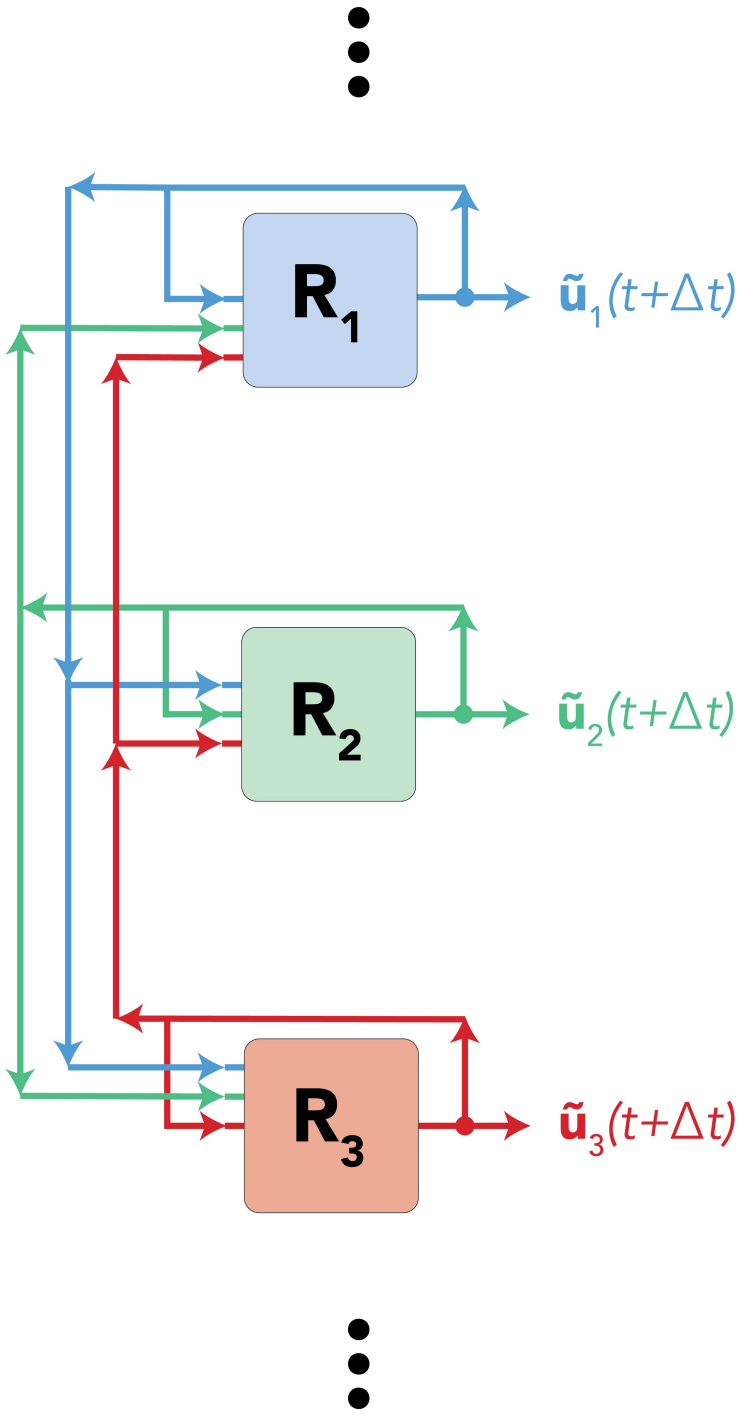
Parallel Reservoir Prediction Framework

1. The i^{th} reservoir **outputs predictions for** the i^{th} node **only**
2. The 1^{st} reservoir **receives prediction input from** the 1^{st} node prediction and its neighboring nodes' predictions



Parallel Reservoir Prediction Framework

1. The i^{th} reservoir **outputs predictions for** the i^{th} node **only**
2. The i^{th} reservoir **receives prediction input from** the i^{th} node prediction and its neighboring nodes' predictions



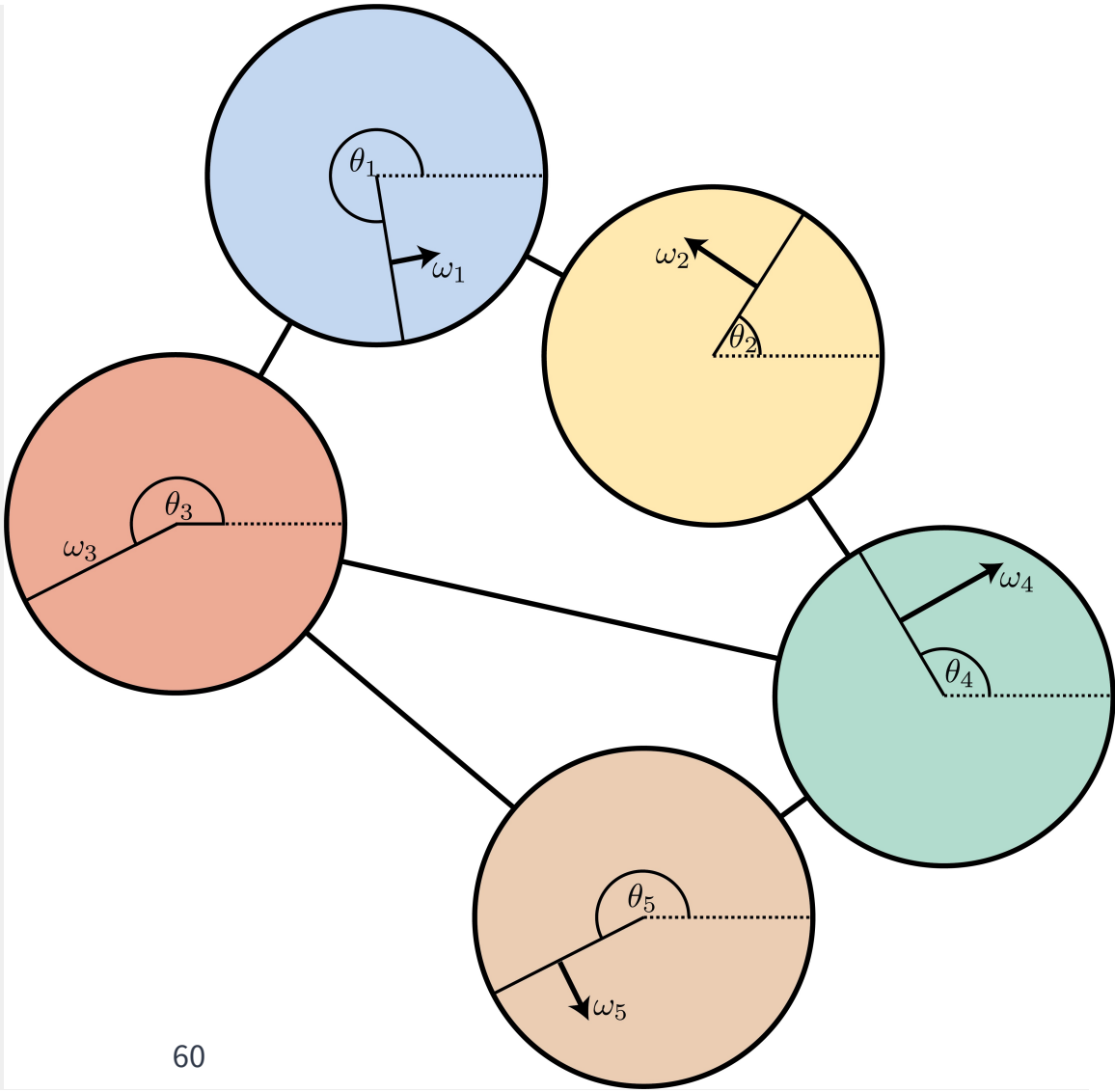
One last example…

Kuramoto Oscillators

- **System of N oscillators described by their phase angles, θ_i**

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

- ω_i: natural frequency
- κ: strength of coupling
- A: connectivity matrix



Kuramoto Oscillators

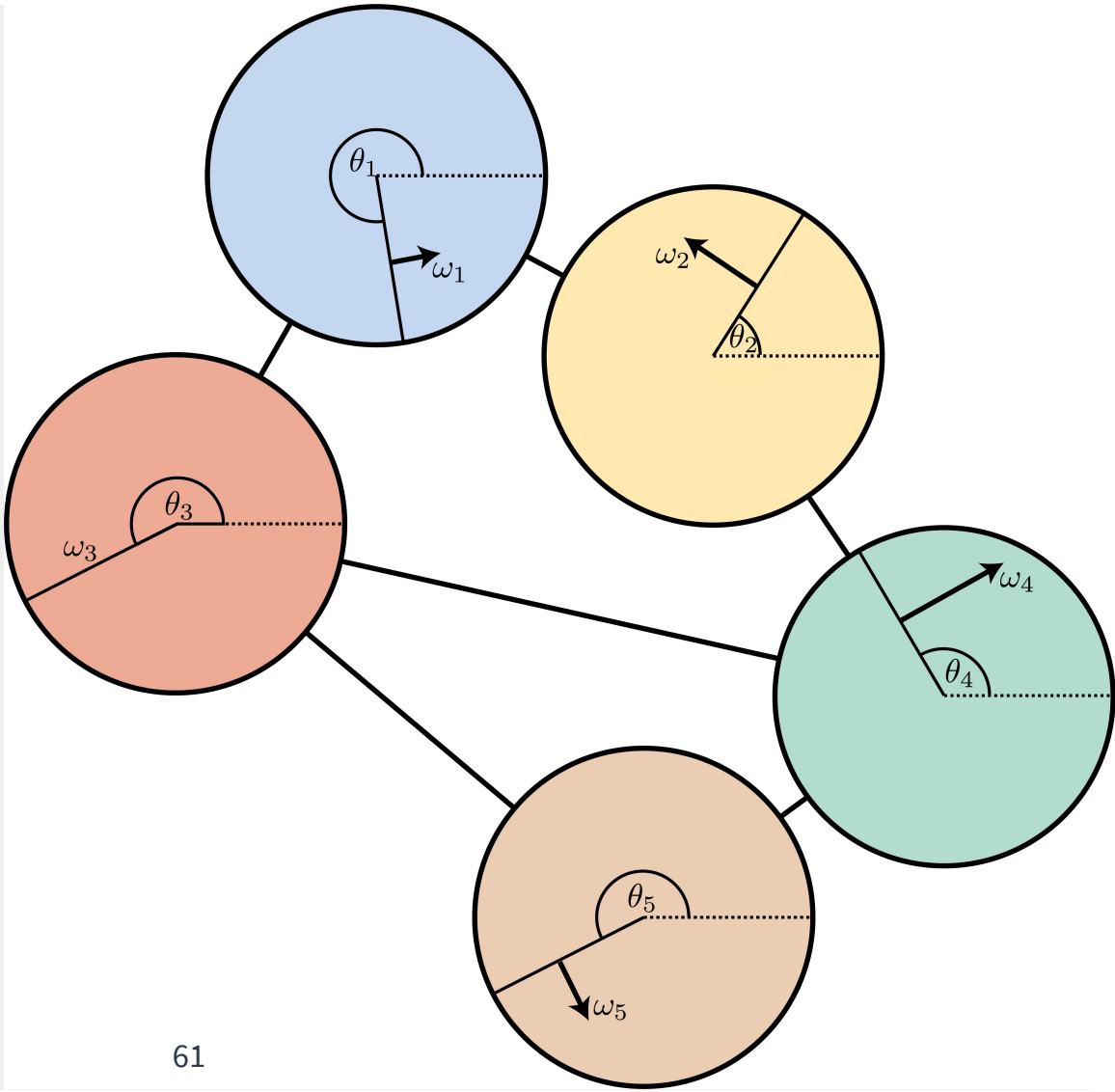
- System of N oscillators described by their phase angles, θ_i

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

- ω_i: natural frequency
- κ: strength of coupling
- A: connectivity matrix

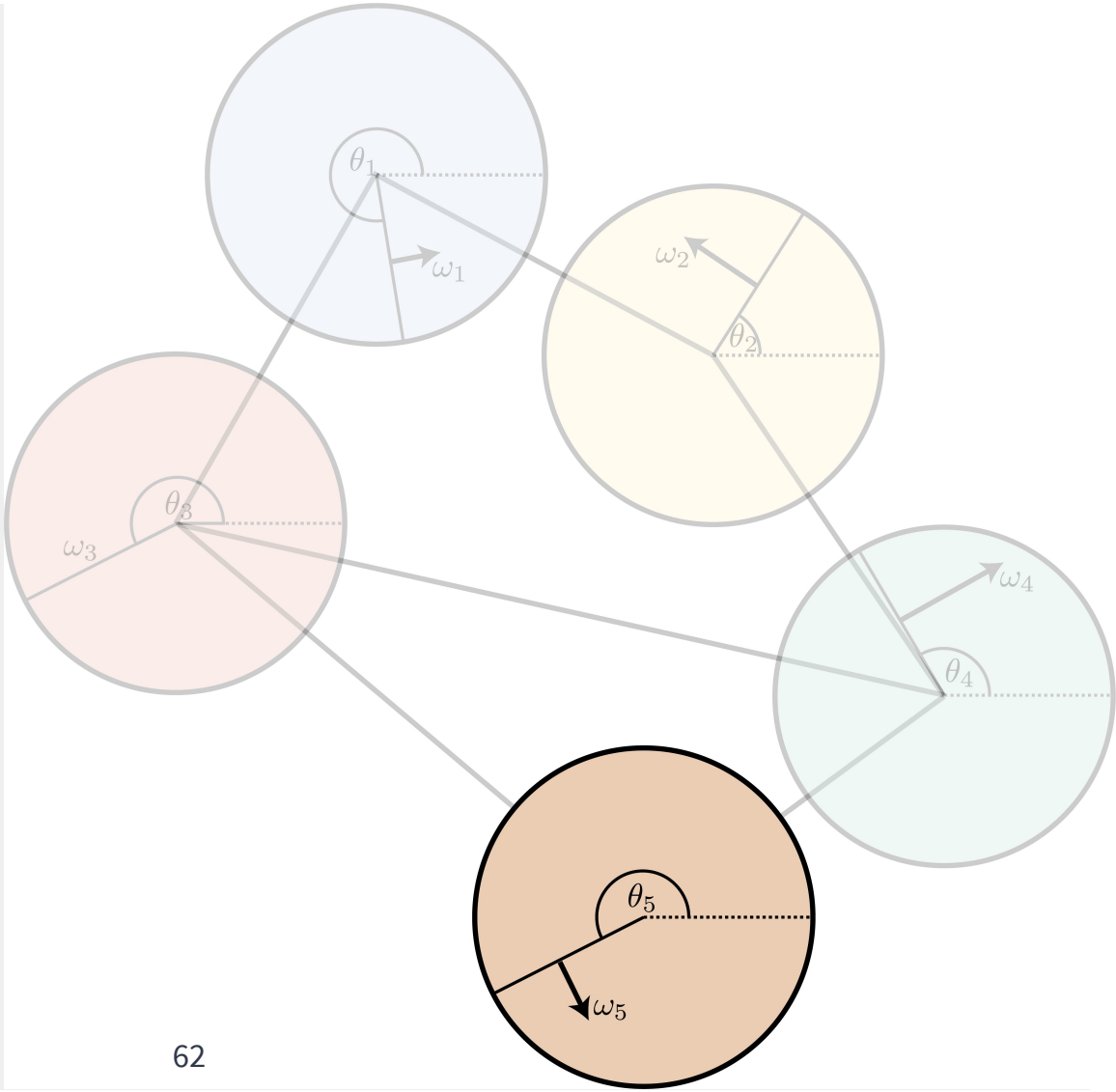
- **Kuramoto oscillators are essentially spinning tops that affect each others' angular frequencies**

- The angle difference between connected oscillators is directly proportional to the change in oscillator frequency



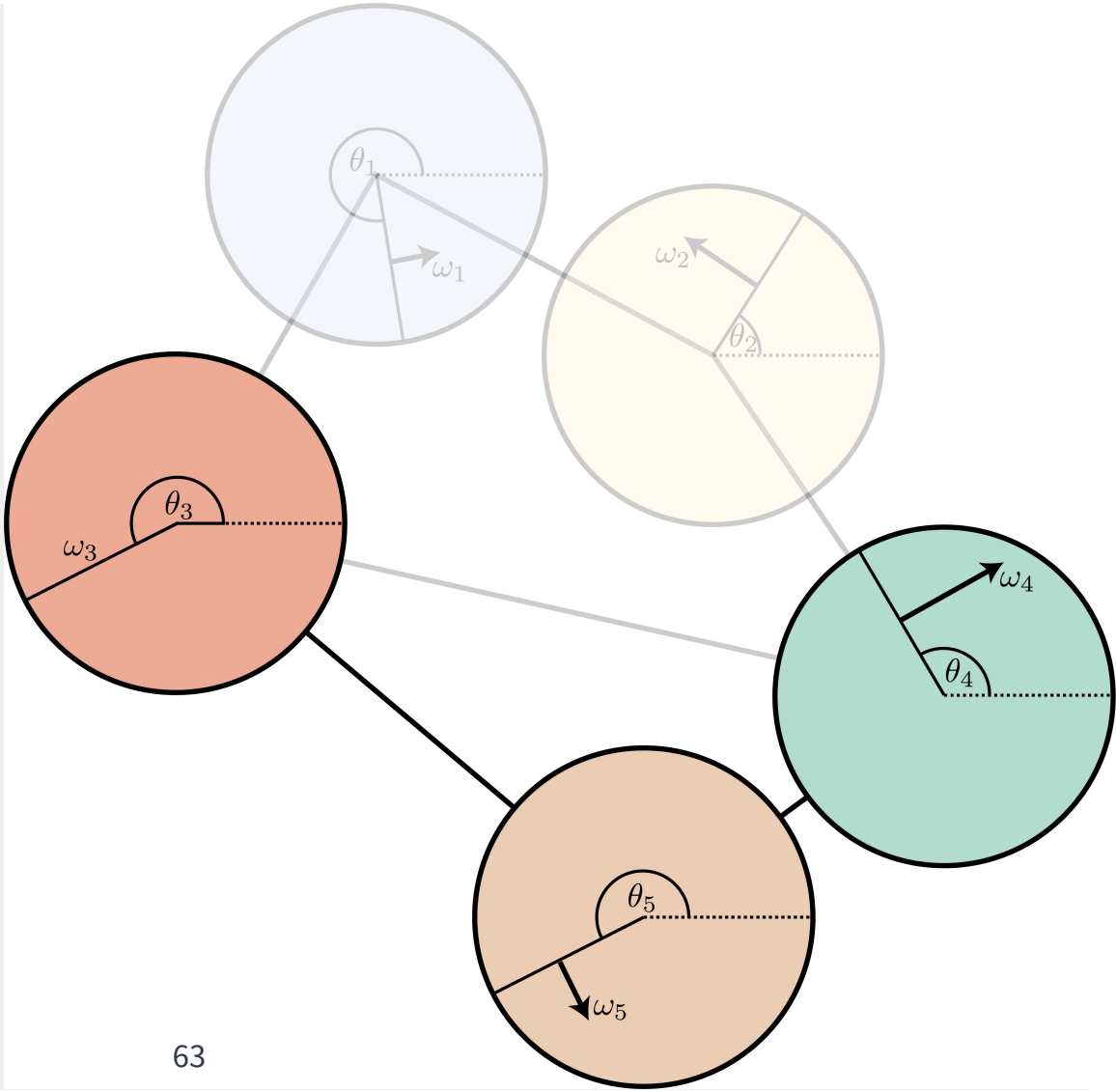
Kuramoto Oscillators

- Looking at the 5^{th} oscillator in particular:



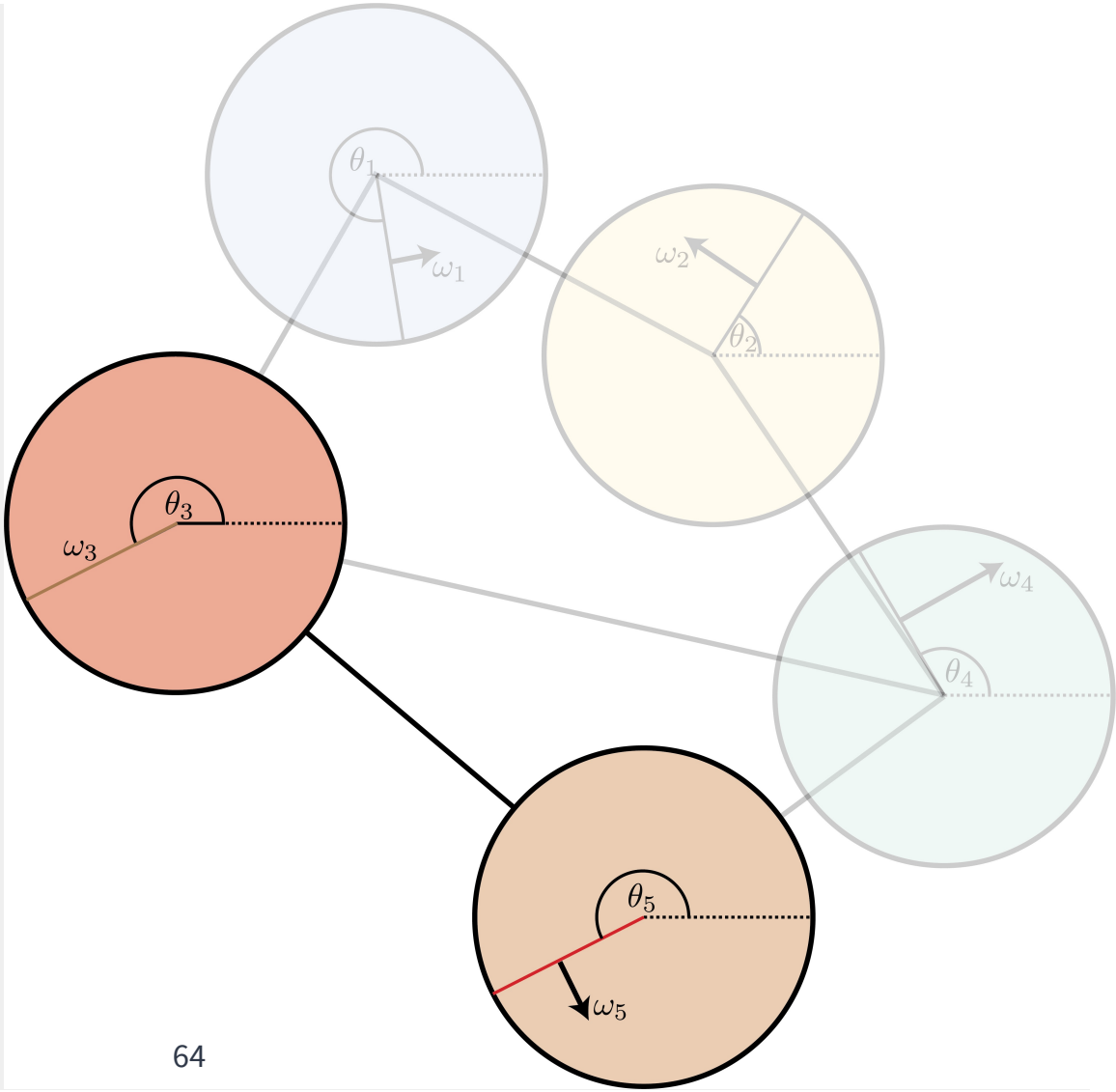
Kuramoto Oscillators

- Looking at the 5^{th} oscillator in particular:
- It is connected to both the 3^{rd} and the 4^{th} oscillators.



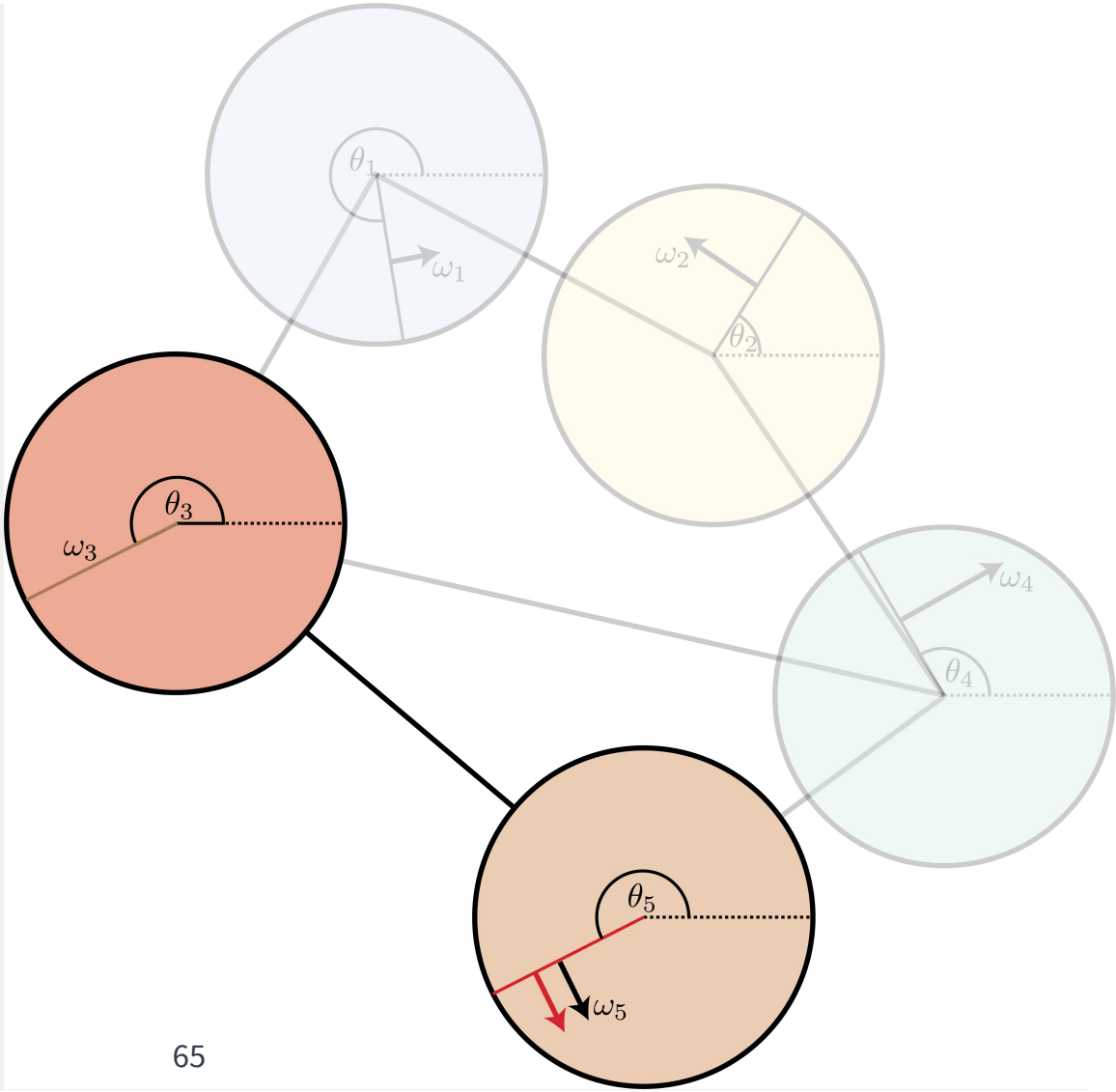
Kuramoto Oscillators

- Looking at the 5^{th} oscillator in particular:
- It is connected to both the 3^{rd} and the 4^{th} oscillators.
- The angle difference between the 5^{th} and 3^{rd} oscillators is 0°, so neither frequency will be affected.



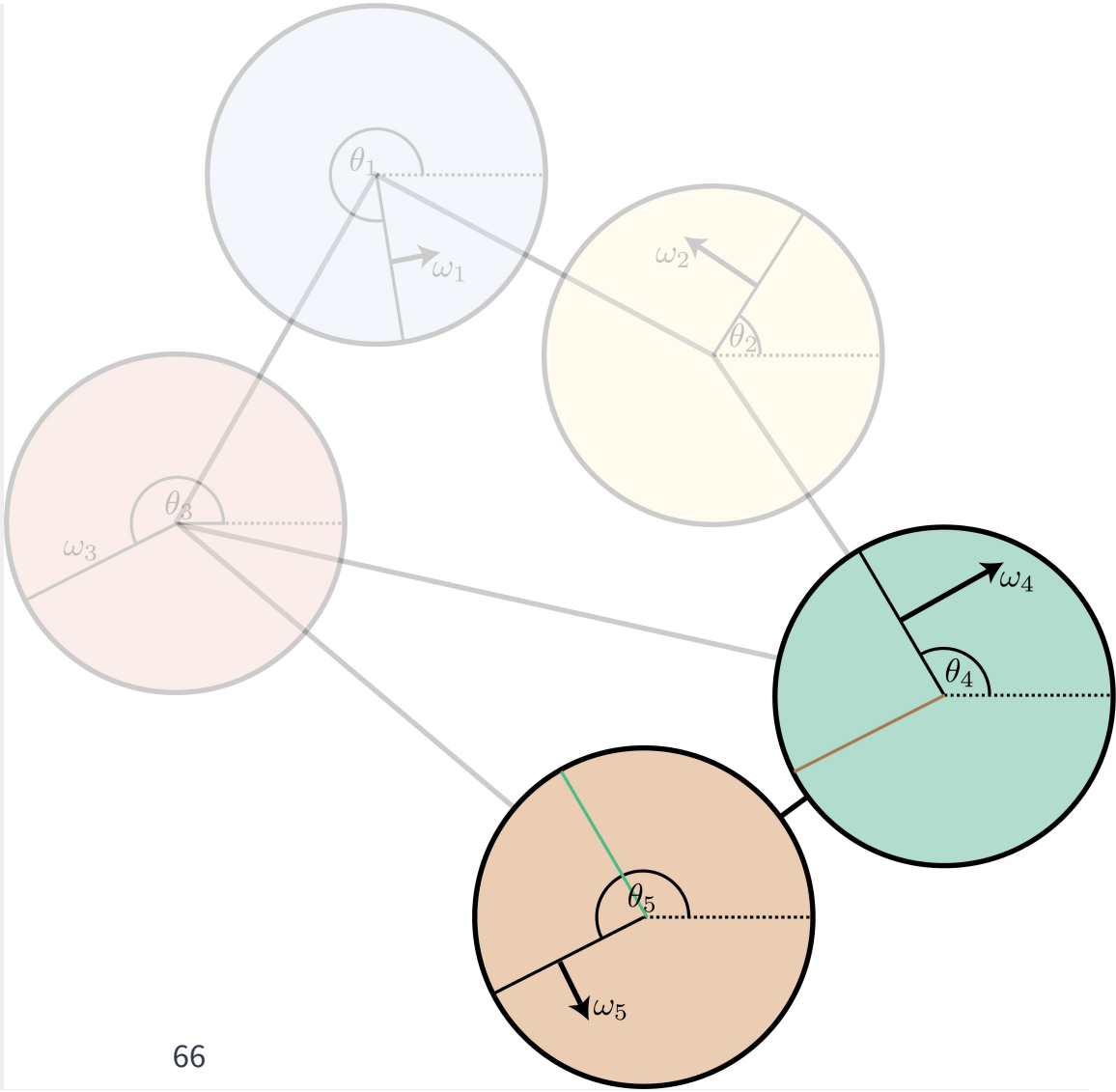
Kuramoto Oscillators

- Looking at the 5^{th} oscillator in particular:
- It is connected to both the 3^{rd} and the 4^{th} oscillators.
- The angle difference between the 5^{th} and 3^{rd} oscillators is 0°, so neither frequency will be affected.



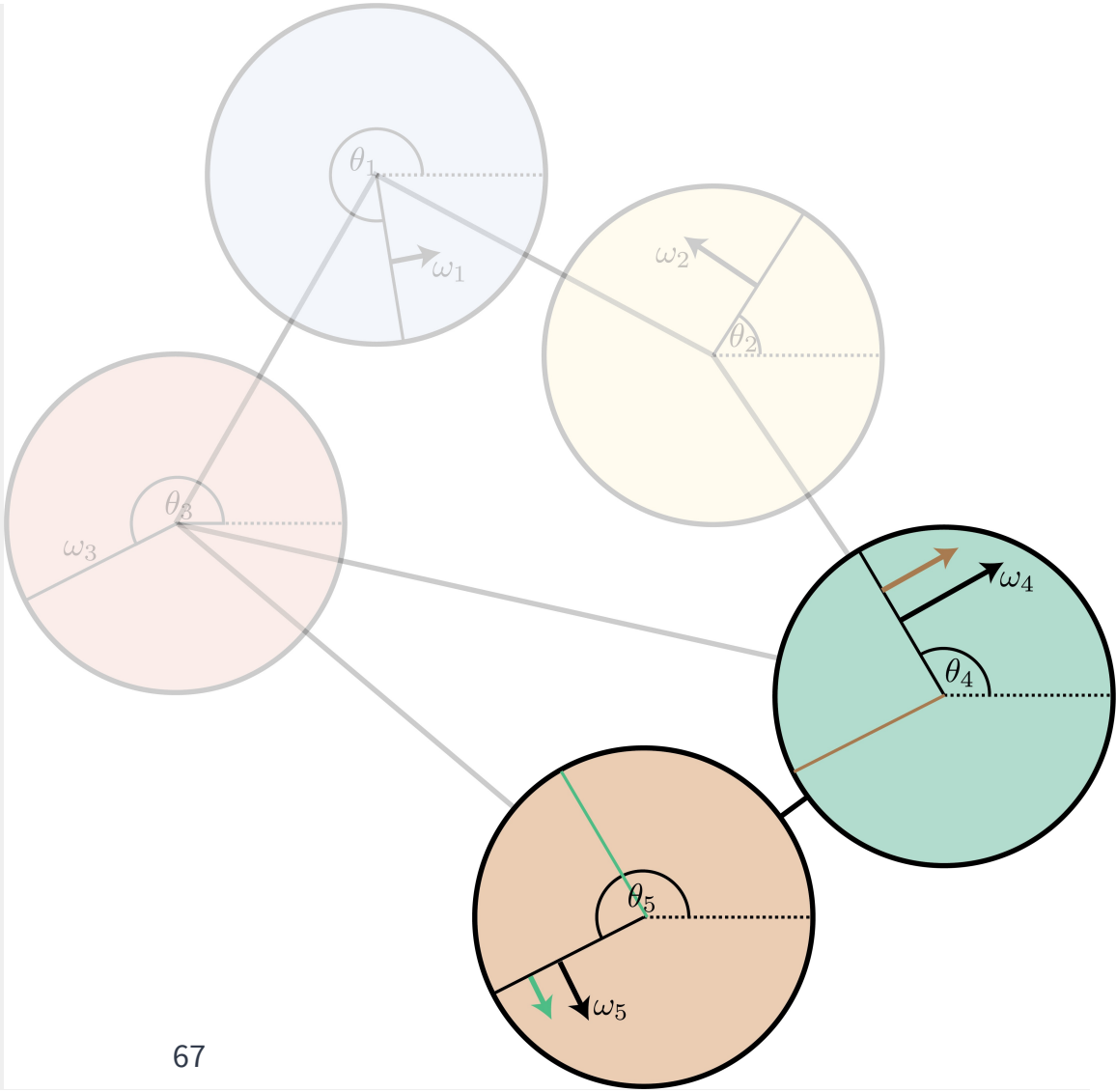
Kuramoto Oscillators

- Looking at the 5^{th} oscillator in particular:
- It is connected to both the 3^{rd} and the 4^{th} oscillators.
- The angle difference between the 5^{th} and 3^{rd} oscillators is 0°, so neither frequency will be affected.
- The angle difference between the 5^{th} and 4^{th} oscillators is larger, so both frequencies will be pulled towards each other.



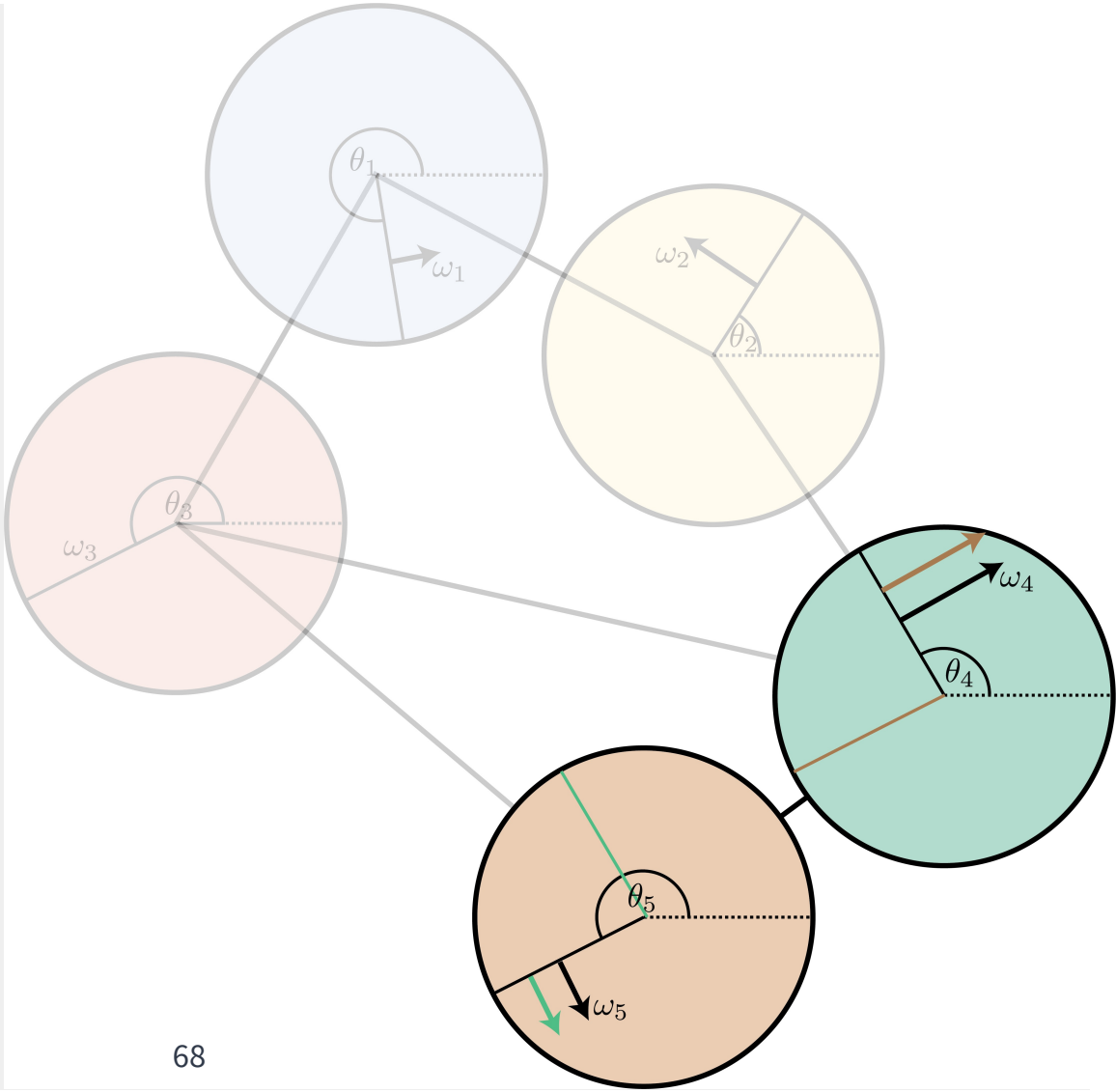
Kuramoto Oscillators

- Looking at the 5^{th} oscillator in particular:
- It is connected to both the 3^{rd} and the 4^{th} oscillators.
- The angle difference between the 5^{th} and 3^{rd} oscillators is 0°, so neither frequency will be affected.
- The angle difference between the 5^{th} and 4^{th} oscillators is larger, so both frequencies will be pulled towards each other.



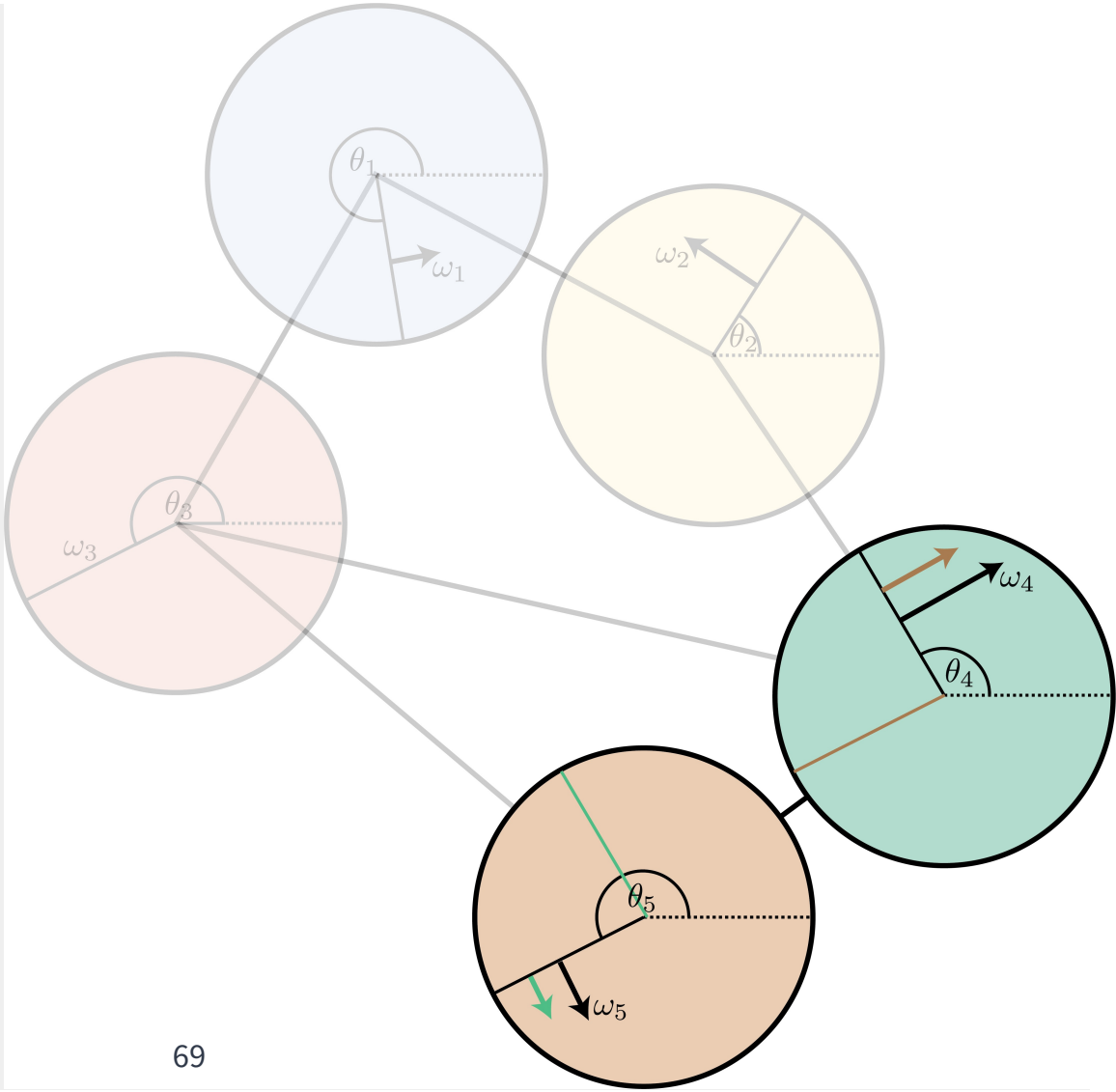
Kuramoto Oscillators

- Looking at the 5^{th} oscillator in particular:
- It is connected to both the 3^{rd} and the 4^{th} oscillators.
- The angle difference between the 5^{th} and 3^{rd} oscillators is 0°, so neither frequency will be affected.
- The angle difference between the 5^{th} and 4^{th} oscillators is larger, so both frequencies will be pulled towards each other.
- How far the frequency is pulled depends on the coupling constant.
  - 0, small, large



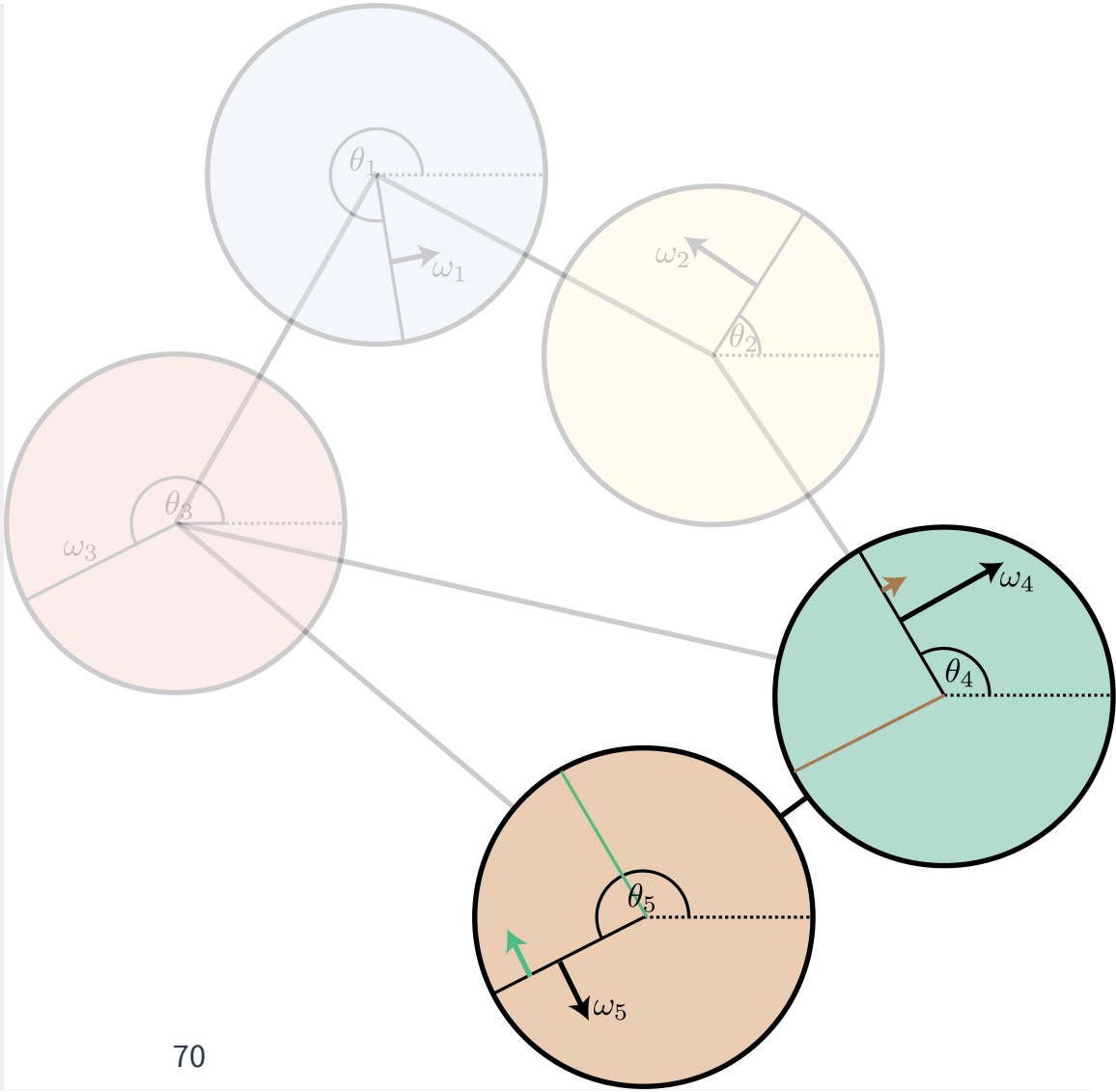
Kuramoto Oscillators

- Looking at the 5^{th} oscillator in particular:
- It is connected to both the 3^{rd} and the 4^{th} oscillators.
- The angle difference between the 5^{th} and 3^{rd} oscillators is 0°, so neither frequency will be affected.
- The angle difference between the 5^{th} and 4^{th} oscillators is larger, so both frequencies will be pulled towards each other.
- How far the frequency is pulled depends on the coupling constant.
  - 0, small, large



Kuramoto Oscillators

- Looking at the 5^{th} oscillator in particular:
- It is connected to both the 3^{rd} and the 4^{th} oscillators.
- The angle difference between the 5^{th} and 3^{rd} oscillators is 0°, so neither frequency will be affected.
- The angle difference between the 5^{th} and 4^{th} oscillators is larger, so both frequencies will be pulled towards each other.
- How far the frequency is pulled depends on the coupling constant.
  - 0, small, large

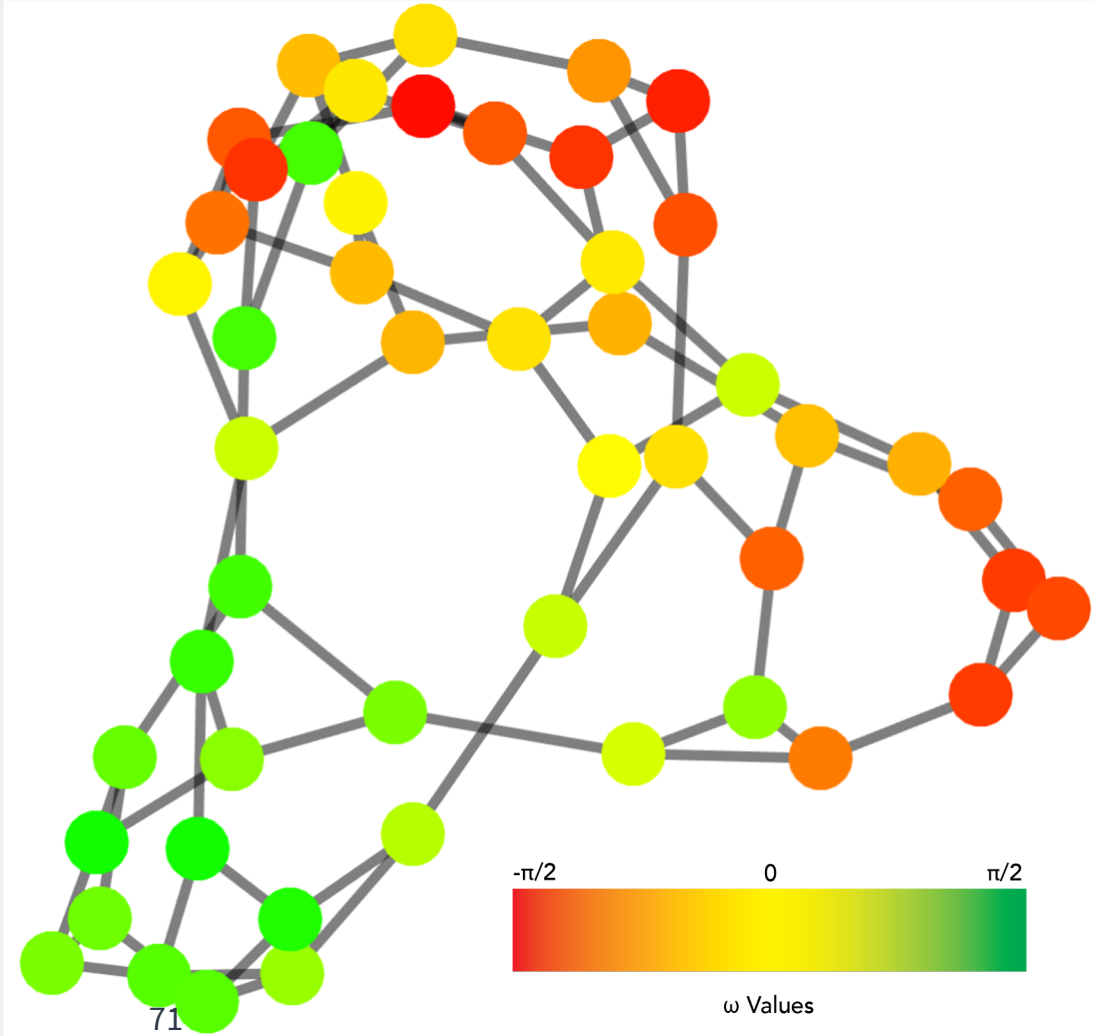


Kuramoto Oscillators

- **System of 50 oscillators described by their phase angles.**

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

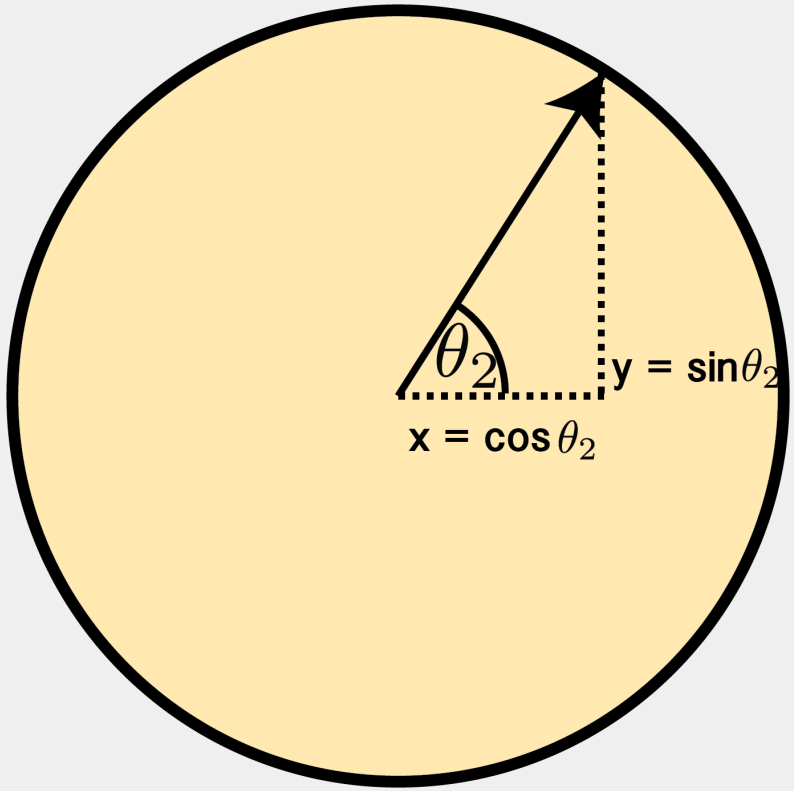
- ω_i: frequencies between ±pi
- κ: 0.5
- A: connectivity matrix — **frequency assortative**



Kuramoto Oscillators

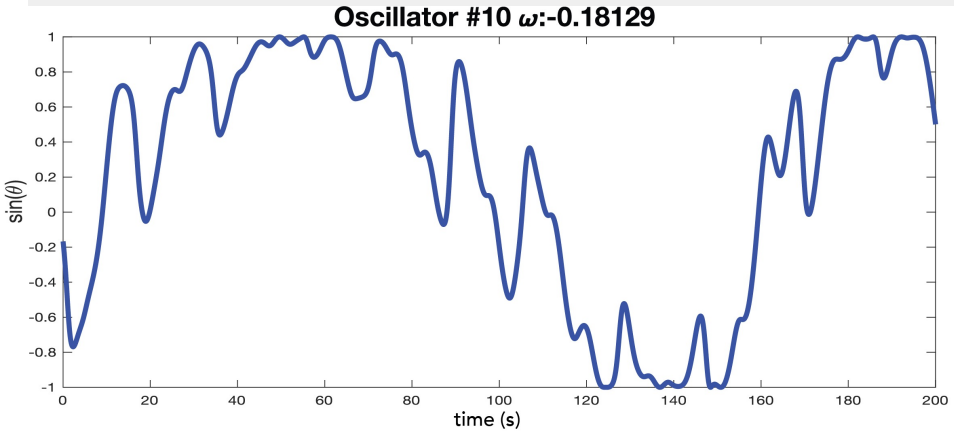
- Kuramoto oscillators are “spinning tops” which means their angles, although oscillatory, are not bounded

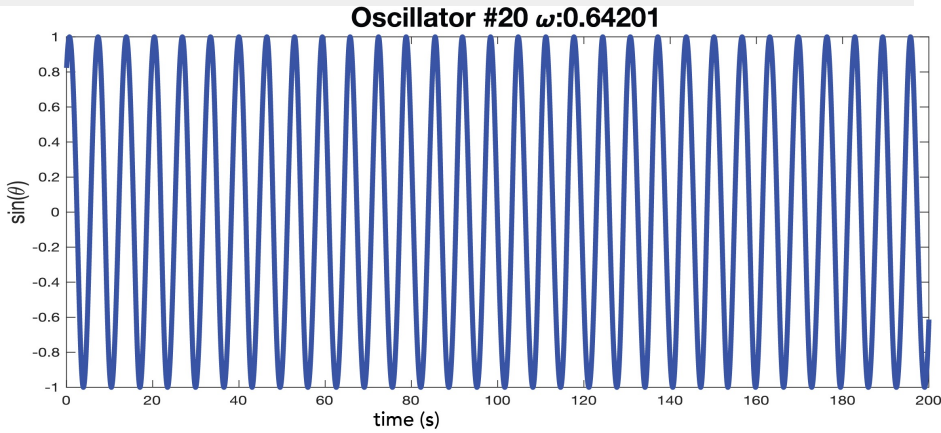
- We will not predict the angle, but rather the cosine and sine of the angle (x and y coordinates)

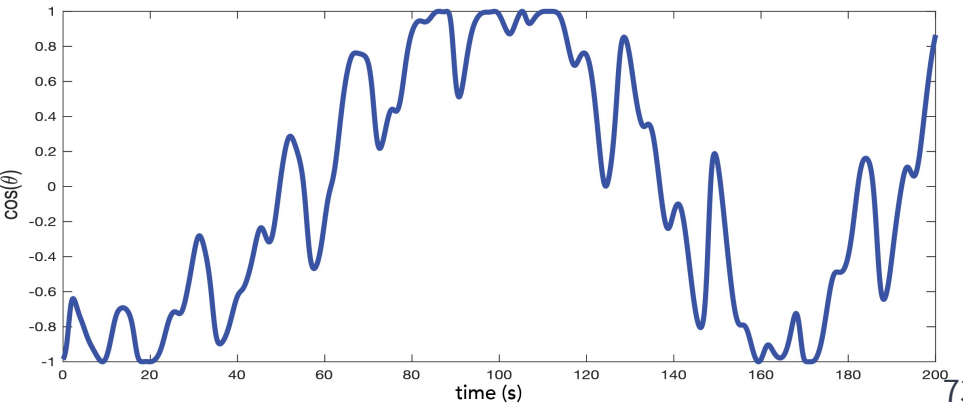


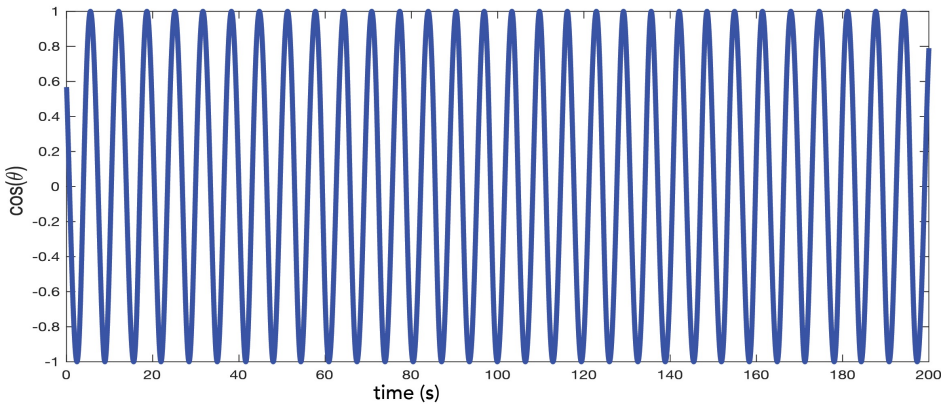
Kuramoto Oscillators

- Below are sine/cosine plots for two of the oscillators in our 50 node Kuramoto oscillator system





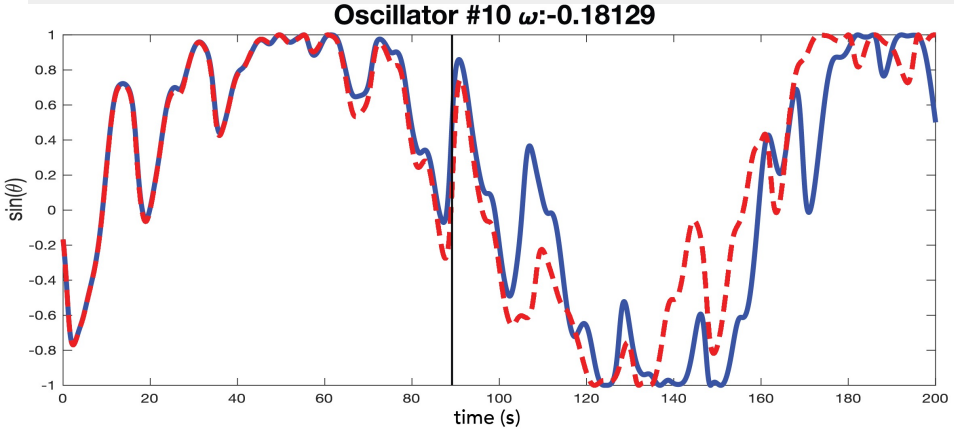


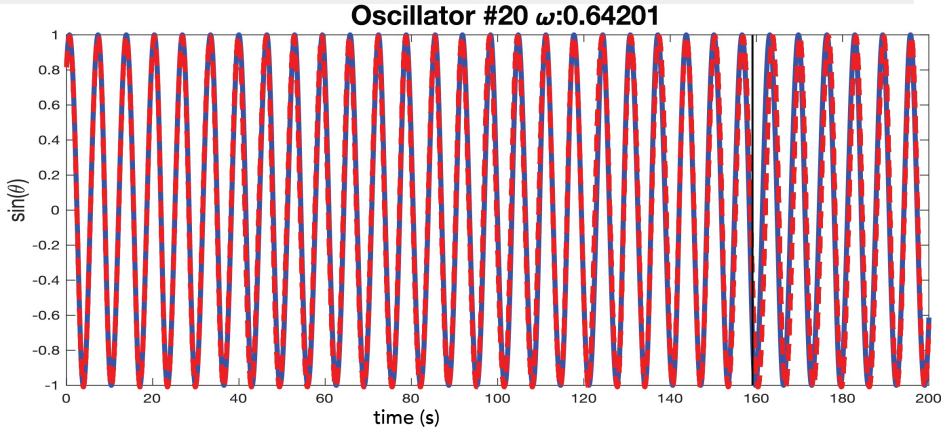


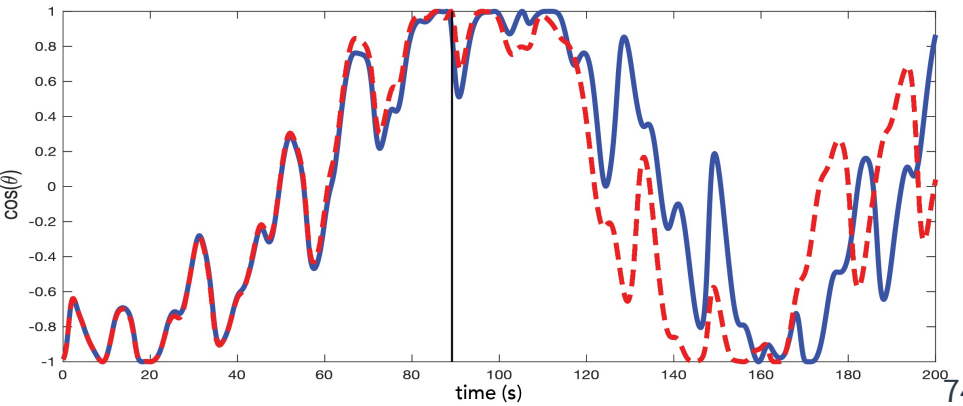
Blue: True State

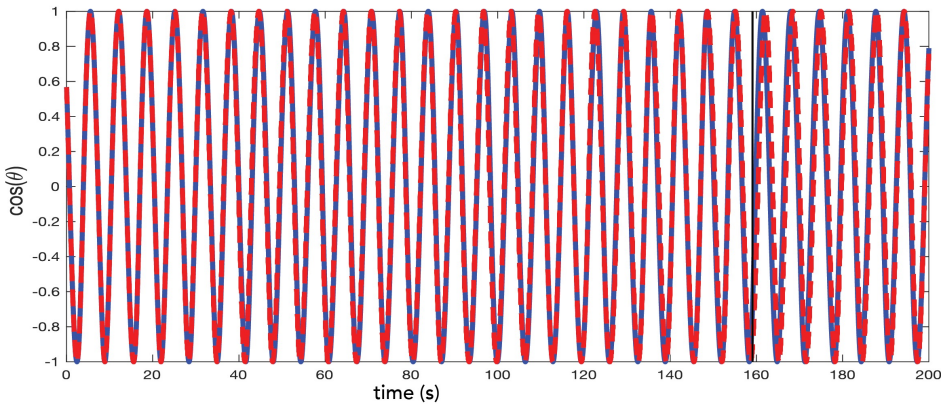
Kuramoto Oscillators

- Sine/cosine predictions for these two oscillators (training phase not shown)









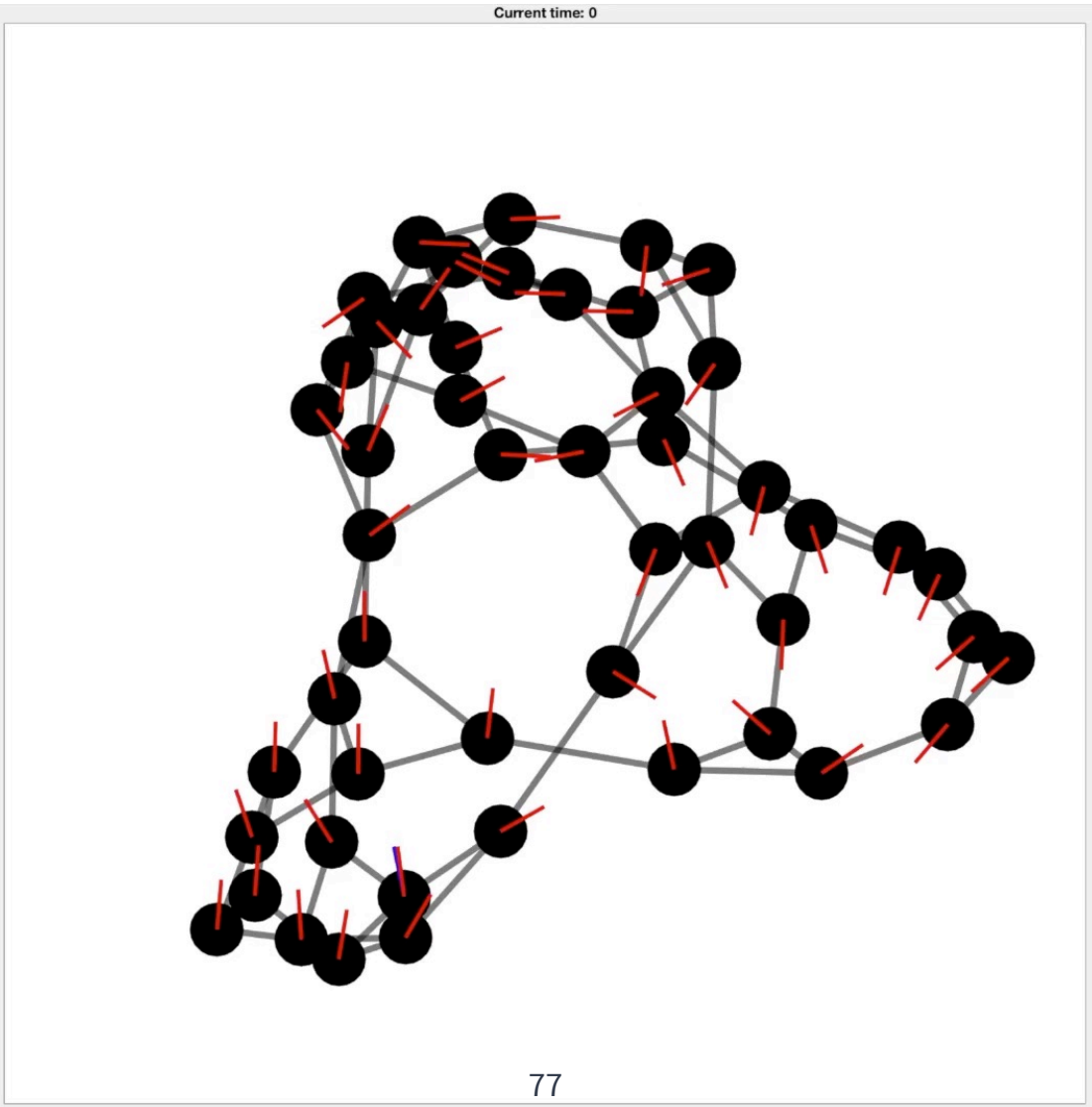
Blue: True State Red: Prediction

What do these results show?

- The parallel prediction scheme is not perfect, but it works well (captures the essence of the model)
- The single reservoir scheme **does not work** for any reasonably sized reservoir, so this is a clear improvement

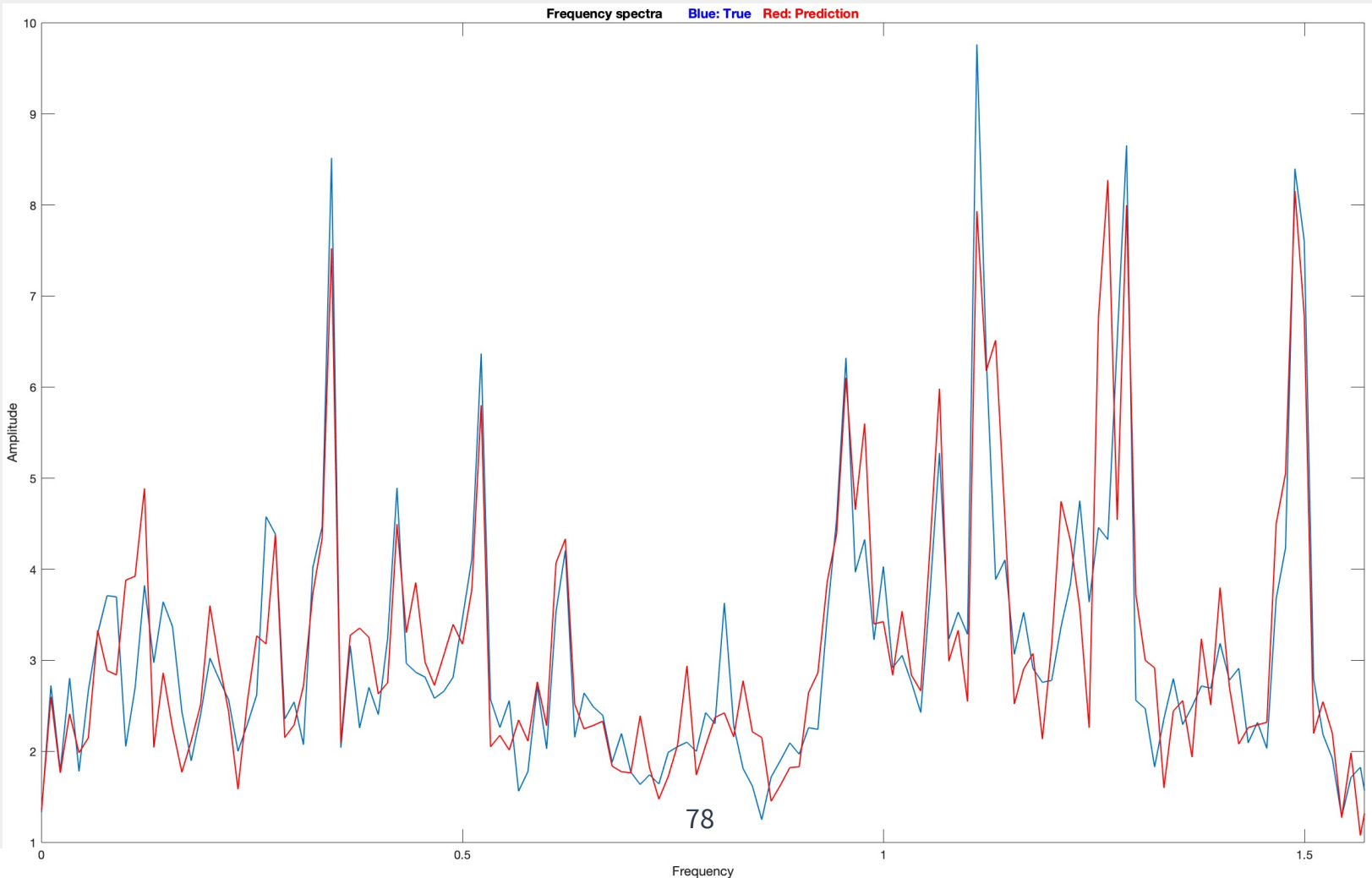
Two more slides showing the accuracy of the predictions…

Kuramoto Oscillators



Kuramoto Oscillators

- Frequency spectrum for the true vs. predicted 50 node oscillator model



Why are these good?

- Many applications of network prediction care about **global** accuracy and **local** accuracy
- The frequency spectrum proves that although the local predictions aren't as good, the global prediction is better
- Ex. Weather prediction

Next Steps

- Improve **local** accuracy
- Predict **more than one** node with each reservoir
- Develop prediction scheme when we don't know the next connections (this is already being done)

Acknowledgements

- TREND Program at UMD
- Chaos Lab at UMD: Michelle Girvan, Ed Ott, Thomas Antonsen
- Keshav Srinivasan