The KuramotRon Quick Start Guide

Ronald D. Smith

March 2019

Introduction

The Kuramoto model is a simple and well-studied model of coupled oscillators. The standard form of the model is

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

where θ_i and ω_i are the phase and intrinsic frequency of the i-th oscillator, N is the total number of oscillators, and K is the coupling strength.

In the absence of coupling ($K = 0$), the oscillators simply change phase, linearly with time, according to their intrinsic frequency. As the coupling strength increases, the oscillators transition from the aforementioned incoherent state to partial, and then full, phase locking. Alternatively, if the coupling strength is negative, the oscillators will tend to distribute themselves evenly throughout phase space. An interactive demonstration of this model can be found by following this link.

The KuramotRon

The simulation is an extension of the Kuramoto model which includes spatial coupling. The oscillators are organized on a square grid, and their dynamics are governed by

$$\frac{d\theta_i}{dt} = \omega_i + \xi + \frac{K}{N}\sum_{j=1}^{N}\sin(\theta_j - \theta_i)\cdot f(D_{ij}^p) \tag{2}$$

Compared to (1), the additional terms in (2) are:

- D_{ij}^p is the Minkowski distance with parameter p. That is,

$$D_{ij}^p = \left(|x_j - x_i|^{1/p} + |y_j - y_i|^{1/p}\right)^p.$$

  For $p = 2$, this is the Euclidean distance. For $p = 1$, it's the taxicab distance. As $p \to \infty$, it approaches the Chebyshev distance (maximum

difference between coordinates). For $p < 1$ this is not a distance as it violates the triangle inequality.

- f is called the 'influence' function which maps distances to a real number. There are several built-in functions, and instructions about how to add your own are in the next section.

- ξ is a noise term, drawn from a normal distribution, the variance of which can be modulated in the simulation.

Running/modifying the simulation

You run the simulation by calling the script `Kuramoto_Spatial`. Two windows will open. One window contains the simulation, and the other has the parameter controls.

The system is simulated using Euler's method (this choice was made to favor speed over accuracy!).

The parameter control window options are (from top to bottom):

- C_{mode}: Controls how the oscillators' phases are displayed. Mode 1 maps the interval $(0, 2\pi)$ equally across the spectrum using MATLAB's hsv colormap. The rest of the modes map a narrow range of phases to one color, and are meant to represent things like neurons firing, frogs croaking, fireflies lighting up, etc. **You can add your own color schemes** by editing the 'GetColors.m' file. The code will automatically detect new options (it does this by counting the number of times the word 'case' appears in 'GetColors.m').

- f: This will say something like 'inverse' or 'sin'. This is the influence function being used. **You can add your own influence functions** by editing the file 'SpatialInfluence.m'. The code will automatically detect any new options the next time you run it. As you move the slider bar left-to-right, the options go in the same order they are defined in the 'SpatialInfluence.m' file. To see a list of all available options, enter `GetSpatialFunctions` into the command window when the simulation is not running.

- α: This is a parameter that is passed to the spatial influence function. To see how the parameter is being used, you'll need to read the code in the file 'SpatialInfluence.m'. For example, if you have selected f = inverse, then $\alpha = 2$ is an inverse square law. Currently, it's only supported for an influence function to have a single parameter.

- p: This is the parameter for the Minkowski distance. $p = 1$ is the taxicab metric, and $p = 2$ is the Euclidean distance. Any values $p \geq 1$ are valid distance metrics, although you can select $0 < p < 1$ and see what happens. See the wikipedia page for more information.

- $|\xi|$: Controls the magnitude of the noise. Specifically, the noise is drawn from a $N(0, |\xi|^2)$ distribution on each iteration.
- dt: The time step used in Euler's method. If the simulation appears to be 'flickering', you can be pretty sure that your time step is too large. Set this as low as possible, but not so low that the simulation appears to be frozen.
- k: The coupling parameter from (2).

The control window also shows a plot of the current influence function. Whenever you change a parameter related to the spatial coupling, the code might slow down for a second while it recomputes this plot.

You can end the simulation by simply closing the simulation window.

Saving and loading settings

If you find some settings that you really like, then quit the simulation (by closing the simulation window) and call:

```
SaveSettings(s,tStamp)
```

This will save your settings to a folder called Output that is one directory up from where the simulation code exists. This directory must exist or you will get an error. For example, if you have saved all the code in /Documents/KuramotRon, then there should also be a directory /Documents/Output. The file will be named something like 'Settings_737490.574866323615.txt' which you can rename to something more descriptive if you like.

You can then import these settings by editing the line in 'Kuramoto_Spatial.m' that looks like:

```
s = table2struct(readtable('../Output/MySettings.txt'));
```

and making sure that the line above it is `if 1`.

That's it! If you find any strange behavior or manage to break it, let me know!