Master CompuPhys - Python Numerical Project Scope statements

The Kuramoto model and superconducting quantum computers

Supervisor: D. Viennot, `mailto:david.viennot@utinam.cnrs.fr`

1 Description of the project

The Kuramoto model is a simple model describing a set of N nonlinear coupled phase oscillators. It is based on the following first order differential equations for $\{t \mapsto \theta^i(t)\}_{i\in\{0,...,N-1\}}$:

$$\dot{\theta}^i(t) = \omega_i + \frac{1}{N}\sum_{j=0}^{N-1} K_{ij} \sin\left(\theta^j(t-\tau_{ij}) - \theta^i(t) + \alpha_{ij}\right) + \eta_i(t) \tag{1}$$

where θ^i is the phase of the i-th oscillator (its "elongation" being $q^i(t) = q^i(0)\cos\left(\theta^i(t)\right)$), ω_i is its "natural" oscillation frequency, K is the coupling matrix, τ is the delay matrix, α is the dephasing matrix of the coupling and η_i represents the possible external noises.
Important remark: Note that θ^i is an angle, it is then defined modulo 2π! Remember this during the programming!
Generally, K_{ij} is inversely proportional to the "physical distance" between the oscillators i and j. In some cases, we can consider only nearest neighbour couplings ($K_{ij} = \kappa$ if i and j are nearest neighbour and otherwise $K_{ij} = 0$). We can also consider an open chain of oscillators coupled onto M nearest neighbours which is described by $K_{i,i+p} = K_{i,i-p} = \kappa$ for $p \in \{1, ..., M\}$ and $K_{ij} = 0$ if $j \notin \{i \pm p\}_{p\in\{1,...,M\}}$; for a closed chain, we consider the same relation but with the label i defined modulo N (i.e. $N \equiv 0$, $N+1 \equiv 1$,..., $-1 \equiv N-1$, $-2 \equiv N-2$, etc).
For $\eta = 0$ and $K = 0$, we refind a set of N independent harmonic oscillators.

The Kuramoto model presents some unusual phenomena, especially:

- for some parameters with large number of non-zero couplings, the Kuramoto model presents a self-synchronisation phenomenon: starting from random initial conditions $\{\theta^i(0)\}_i$, after a short time of transcient regime, the system is in a stable state where all the oscillators are synchroneous;
- for some parameters with large N, the Kuramoto model exhibits chimera states: starting from random initial conditions $\{\theta^i(0)\}_i$, after a short time of transcient regime, the system is in a metastable state where a part of the system synchroneously oscillates and an other part presents chaotic oscillations, without propagation of the chaos nor complete self-synchronisation. Such a state is metastable, i.e. it seems stable during a long time larger than the transcient regime time, but after that, the system collapses into a completely chaotic or synchroneous state. The duration of the chimera regime grows with N (chimera states are stable at the limit $N \to +\infty$).

Surprisingly the Kuramoto model is a very good description of a lot of different concrete (physical, chemical and biological) systems. In this project, we consider the following example:
A coupled array of Josephson junctions can be described by the Kuramoto model (see the article at *Physical Review E* in the supplementary files). A Josephson junction is an electronic device

constituted by an insulating layer in sandwich between two superconducting layers, where quantum effects occur as tunneling effects of Cooper pairs[1] from a superconducting layer to the other one. Arrays of Josephson junctions are the current architecture to build quantum computers (see `https://en.wikipedia.org/wiki/Superconducting_quantum_computing`)[2].

The goal of this project is the realisation of a simulator of an array of Josephson junctions based on the Kuramoto model.

2 Expected deliverables

2.1 Alpha version

Code developments:

- Code three simple integrators of first order differential equations based on respectively the Euler, the RK2 and the RK4 algorithms (see section 3).
- Code the variables, objects and functions needed to numerically represent the Kuramoto model eq. (1) in the simple case of an open or closed chain without noises, dephasing nor delay. Apply the integrators to simulate the dynamics of Kuramoto models.
- Code functions to compute the parameter orders R and Φ defined by:

$$Re^{\imath\Phi} = \frac{1}{N}\sum_{i=1}^{N} e^{\imath\theta^i} \tag{2}$$

  for a single configuration $\{\theta^i\}_i$ and code functions to draw the graphs of $t \mapsto R(t)$ and $t \mapsto \Phi(t)$.
- Code a function to compute the local Shannon entropy: $S_i^{q,n} = -\sum_{a=0}^{q-1} p_a \ln p_a$ where $p_a \in [0,1]$ is the fraction of the $2n+1$ oscillators around the i-th (oscillators labeled by $k \in \{i-n, i-n+1, ..., i-1, i, i+1, ..., i+n\}$) which have $\theta^k \in [\frac{2\pi a}{q}, \frac{2\pi(a+1)}{q}[$ (for a single configuration $\{\theta^i\}_i$).
  Code functions to draw the 1D graph of $i \mapsto S_i^{q,n}(t)$ at a time t and to draw a density graph of $(i,t) \mapsto S_i^{q,n}(t)$.
- Code a function to draw a graph representing the phases $\{\theta^i(t)\}_{i=0,...,N-1}$ at a time t on a circle; a function to draw in the same graph the N curves $t \mapsto \theta^i(t)$, and a function to draw a density graph of $(i,t) \mapsto \theta^i(t)$.

Physical study:

- Draw the different graphs associated with the dynamics of the Kuramoto model for different parameters N, κ, M, $\{\omega_i\}$ (case of the chain), by randomly choosing the initial conditions. Try to find interesting or remarkable cases. (Choose N at least equal to 100).
- Compare the three integrators for the numerical simulations.
- Try to interprete the quantities R, Φ and $S^{q,n}$ with respect to the obtained results.
- Try to illustrate the phenomena of self-synchronisation and of chimera states.

[1] A Cooper pair is a boson "formed by two electrons in interaction via the phonons" at ultra cold temperature

[2] In this project, for the sake of simplicity, we study only the classical regime of an array of Josephson junctions, and not the quantum regime.

2.2 Beta version

Code developments:

- Recode the different elements developed in the alpha version for the general case of the Kuramoto model (any coupling matrix K, any delay matrix τ, any dephasing matrix α and Gaussian white noises[3] η_i).
- For a 2D array of $N = N_r \times N_c$ oscillators, code functions permitting to pass from the raw-column coordinates (r, c) to a single number labelisation i, and reciproqually.
  Moroever you can code functions to built quickly matrices K, τ and α for special cases (for example with K_{ij} inversely proportional and τ_{ij} proportional to the distance between the nodes i and j in the 2D array). You can also code functions to build K, τ and α as random matrices. And finally you can code function to build matrices K, τ and α with a structure corresponding to your chosen physical application.
- Code a function to graphically reprensent the connectivity of the network defined by K, with an option parameter κ_{min} such that the graph presents an edge between the nodes i and j if only $K_{ij} > \kappa_{min}$ (the case $\kappa_{min} = 0$ being not excluded).
- Recode the function to compute $S_i^{q,n}$ for a 2D array of oscillators where we consider $S_i^{q,n}$ associated with the oscillators around the one labeled by i in a circle of radius of n nodes in the array.
- Recode the function to draw $(r, c) \mapsto S_{i_{r,d}}^{q,n}(t)$ at a time t as a density graph ($i_{r,d}$ is the label of the node of coordinates (r, d) in the array).
- Code a function to draw $(r, c) \mapsto \theta^{i_{r,d}}(t)$ at a time t as a density graph.

Physical study:

- Built a Kuramoto model corresponding to the application, by seeing the data in the associated article.
- Try to reproduce the numerical results of the article.
- Draw the different graphs for your model and illustrate with your model the phenomena of self-synchronisation and of chimera state.
- Try to write a general discussion concerning the simulation and the interpretation with a Kuramoto system.

2.3 Gold version

- Recode the functions to draw $(r, c, t) \mapsto S_{i_{r,c}}^{q,n}(t)$ and $(r, c, t) \mapsto \theta_{i_{r,c}}(t)$ as animated 2D density graphs, and $(i, t) \mapsto \theta^i(t)$ as an animated graph of points moving onto a circle.
- Recode the function to draw the graph connectivity defined by K in order to the value of K_{ij} is represented onto the edge between the nodes i and j by a coloration or by the thickness of the edge.
- Code an user graphical interface of your simulator as a dashboard displaying the different graphs and permitting to control directly all the parameters of the model.

[3]The value of $\eta_i(t)$ is at each time $t = n\Delta t$ ($n \in \mathbb{N}$) randomly choosen following the Normal distribution with mean value $\langle \eta_i \rangle = 0$ and variance $(\Delta\eta_i)^2 = \frac{\sigma_i^2}{\Delta t}$ for some σ_i

3 Algorithms to integrate a first order differential equation

Let $\dot{\theta} = F(\theta)$ be a first-order differential equation where $\theta \in \mathbb{R}^n$ and $F : \mathbb{R}^n \to \mathbb{R}^n$ (F is not a linear map). We denote by θ^i the i-th component of θ and by $F^i(\theta)$ the i-th component of the image of θ by F.
Let T be the time on which we want to simulate the system. Let $\Delta t = \frac{T}{N_t}$ be the step of the simulation ($N_t \in \mathbb{N}$ is total number of steps in the simulation). The time dependence is then modelized with the time partition $\{t_0, t_1, ..., t_{N_t}\}$ with $t_n = n\Delta t$.
Let $\theta_n \equiv \theta(t_n)$ be the value of the vector θ at time t_n. Attention to the confusion between θ^i (i-th component of θ at a certain time) and θ_n (the vector θ at time t_n), and note that $\theta^i_n = \theta^i(t_n)$.

The considered mumerical integrators of first order differential equations are based on the definition of the derivative:

$$\dot{\theta}(t_n) = \lim_{h \to 0} \frac{\theta(t_n + h) - \theta(t_n)}{h} \simeq \frac{\theta_{n+1} - \theta_n}{\Delta t} \tag{3}$$

with Δt sufficiently small (N_t sufficiently large, the time partition sufficiently thin).

3.1 Euler algorithm

The Euler algorithm is based on the direct use of eq. (3), its propagation scheme is:

$$\theta_{n+1} = \theta_n + F(\theta_n)\Delta t \tag{4}$$

3.2 RK2 algorithm

The Runge-Kutta algorithm is based on the double use of eq. (3) with an intermediate point at the middle of the step. Its propagation scheme is:

$$\theta_{n+1} = \theta_n + F\left(\theta_n + F(\theta_n)\frac{\Delta t}{2}\right)\Delta t \tag{5}$$

3.3 RK4 algorithm

The fourth order RK algorithm is based on the same principle than the RK2 algorithm, but with a refinement to estimate F at the middle point. Its propagation scheme is:

$$\theta_{n+1} = \theta_n + (K_{1,n} + 2K_{2,n} + 2K_{3,n} + K_{4,n})\frac{\Delta t}{6} \tag{6}$$

with

$$K_{1,n} = F(\theta_n) \tag{7}$$

$$K_{2,n} = F\left(\theta_n + K_{1,n}\frac{\Delta t}{2}\right) \tag{8}$$

$$K_{3,n} = F\left(\theta_n + K_{2,n}\frac{\Delta t}{2}\right) \tag{9}$$

$$K_{4,n} = F(\theta_n + K_{3,n}\Delta t) \tag{10}$$