

Cycles et boucles

Synchronisation d'oscillations neuronales

Du modèle de Kuramoto au cycle limite
d'un système linéaire complexe

Ibrahim Khalil Joubairi

Fil conducteur

Idée générale

Des oscillateurs ayant des fréquences propres différentes peuvent, grâce au couplage, adopter une fréquence collective.

Objet d'étude

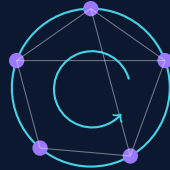
On suit l'évolution des phases et on cherche des conditions de convergence vers un régime périodique stable.

Fil directeur

Hétérogénéité individuelle



Synchronisation collective



Objectif : expliquer l'émergence d'un rythme collectif stable.

Contexte scientifique

Phénomènes oscillatoires

Les oscillations sont omniprésentes dans les sciences :

- ▶ oscillations mécaniques ;
- ▶ oscillations électriques ;
- ▶ oscillations biologiques.

Dans le cerveau humain

Les signaux EEG montrent que l'activité cérébrale n'est pas aléatoire, mais organisée en rythmes collectifs.

Rythmes cérébraux

Les oscillations neuronales apparaissent sous forme de bandes de fréquences :

$$\delta, \theta, \alpha, \beta, \gamma$$

Question naturelle

Comment des neurones ou populations neuronales, ayant chacun leur propre dynamique, peuvent-ils produire une activité collective organisée ?

Enjeu : passer d'oscillateurs individuels hétérogènes à un rythme collectif stable.

Problématique retenue

Comment un ensemble d'oscillateurs hétérogènes peut-il converger vers un régime oscillant collectif stable, et en quoi cette synchronisation peut-elle être interprétée mathématiquement comme l'attraction vers un cycle limite, notamment dans le cadre des oscillations neuronales ?

Difficulté

Les oscillateurs possèdent des fréquences propres différentes :

$$\omega_1, \omega_2, \dots, \omega_N$$

Cette hétérogénéité tend à désorganiser les phases.

Mécanisme étudié

Le couplage entre oscillateurs peut compenser cette dispersion et produire un mouvement collectif :

$$\theta_i(t) - \theta_j(t) = \text{constante}$$

Question : comment démontrer l'apparition d'un régime stable ?

Objectifs et démarche scientifique

Objectifs théoriques

- ▶ Modéliser les oscillations neuronales par des oscillateurs de phase couplés.
- ▶ Étudier le modèle de Kuramoto.
- ▶ Construire un système linéaire complexe associé.
- ▶ Montrer la convergence vers un régime oscillant collectif stable.

Objectifs numériques et expérimentaux

- ▶ Réaliser des simulations numériques sous Python.
- ▶ Observer le verrouillage de phase.
- ▶ Mettre en évidence une fréquence collective.
- ▶ Confronter le modèle à une implémentation Arduino.



Démarche : modéliser, analyser, simuler, confronter au réel.

Notions fondamentales

Système dynamique

Un système dynamique décrit l'évolution temporelle d'un état :

$$\dot{x} = f(x).$$

Attracteur

Un attracteur est un ensemble vers lequel les trajectoires évoluent après un régime transitoire.

Cycle limite

Un cycle limite est une trajectoire périodique isolée qui attire les trajectoires voisines.

Interprétation retenue

La synchronisation sera interprétée comme la convergence vers un régime oscillant collectif stable.

Idée clé : un comportement collectif stable peut émerger d'une dynamique individuelle hétérogène.

Le modèle de Kuramoto

On modélise un réseau de N oscillateurs couplés. Chaque oscillateur est décrit uniquement par sa phase $\theta_i(t)$ et possède une fréquence propre ω_i .

Équation du modèle

$$\dot{\theta}_i(t) = \omega_i + \frac{K}{N} \sum_{j=1}^N \sin(\theta_j(t) - \theta_i(t)), \quad i = 1, \dots, N.$$

- ▶ ω_i : fréquence naturelle de l'oscillateur i .
- ▶ K : intensité du couplage global.
- ▶ $\sin(\theta_j - \theta_i)$: interaction qui tend à réduire les écarts de phase.
- ▶ $\frac{1}{N}$: normalisation lorsque le nombre d'oscillateurs augmente.

Idée essentielle : si le couplage est assez fort, les phases peuvent se verrouiller.

Paramètre d'ordre : mesurer la synchronisation

Pour étudier la synchronisation, il ne suffit pas de suivre séparément toutes les phases $\theta_i(t)$. On introduit donc un indicateur global appelé **paramètre d'ordre**.

Définition

$$re^{i\psi} = \frac{1}{N} \sum_{j=1}^N e^{i\theta_j}$$

- ▶ $r \in [0, 1]$ mesure le degré de synchronisation.
- ▶ $r \simeq 0$: les phases sont dispersées sur le cercle.
- ▶ $r \simeq 1$: les phases sont presque alignées.
- ▶ ψ représente la phase moyenne du groupe.

Idée clé : plus $r(t)$ augmente, plus le réseau devient synchronisé.

Réécriture du modèle : effet de champ moyen

Le paramètre d'ordre permet de remplacer l'influence de tous les oscillateurs par une action collective moyenne.

Identité fondamentale

$$\frac{1}{N} \sum_{j=1}^N \sin(\theta_j - \theta_i) = r \sin(\psi - \theta_i)$$

Forme réduite du modèle de Kuramoto

$$\dot{\theta}_i = \omega_i + Kr \sin(\psi - \theta_i)$$

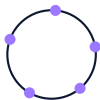
- ▶ ψ représente la direction moyenne du groupe.
- ▶ r mesure la cohérence des phases.

Interprétation : chaque oscillateur est attiré par le groupe moyen.

De la désynchronisation au verrouillage de phase

Le modèle de Kuramoto décrit une transition progressive lorsque le couplage K augmente.

Désynchronisation



K faible
 $r(t) \simeq 0$



Synchronisation



K suffisant
 $r(t) > 0$

Verrouillage de phase

$$\theta_i(t) = \Omega t + \varphi_i \quad \Longrightarrow \quad \theta_i(t) - \theta_j(t) = \text{constante}$$

Condition de synchronisation et limites de Kuramoto

Le verrouillage de phase exige que le couplage compense l'écart entre les fréquences propres.

Condition qualitative

$$|\omega_i - \Omega| \leq Kr_\infty$$

Dispersion des fréquences

Si les ω_i sont trop éloignées, les oscillateurs gardent des rythmes différents.

$$|\omega_i - \Omega| \text{ grand}$$

Effet du couplage

Si K est suffisant, le couplage peut imposer une fréquence collective.

$$Kr_\infty \text{ grand}$$

Limites analytiques

Modèle non linéaire, stabilité globale difficile à démontrer directement, cycle limite non explicite.

Transition : utiliser un système linéaire complexe pour étudier la convergence.

Pourquoi introduire un modèle linéaire complexe ?

Le modèle de Kuramoto décrit la synchronisation, mais son analyse directe est difficile à cause du terme non linéaire $\sin(\theta_j - \theta_i)$.

Limites de Kuramoto

- ▶ non-linéarité ;
- ▶ stabilité globale délicate ;
- ▶ cycle limite non explicite.

Méthode utilisée

- ▶ variables complexes ;
- ▶ système linéaire ;
- ▶ étude par le spectre.

Idée centrale

Construire un système linéaire dont le comportement asymptotique reproduit la synchronisation de phase de Kuramoto.

Ce n'est pas une linéarisation locale de Taylor, mais une représentation auxiliaire.

À retenir : le modèle complexe est un outil auxiliaire pour rendre la convergence accessible par le spectre.

Construction du modèle linéaire complexe

On associe à chaque oscillateur une variable complexe contenant à la fois l'amplitude et la phase.

Variable complexe

$$x_i(t) = \rho_i(t)e^{i\theta_i(t)}, \quad \rho_i(t) = |x_i(t)|, \quad \theta_i(t) = \arg(x_i(t)).$$

Système associé

$$\dot{x}_i = (i\omega_i - \mu_i)x_i + \frac{K}{N} \sum_{j=1}^N x_j, \quad i = 1, \dots, N.$$

Le terme $i\omega_i x_i$ donne la rotation propre, $-\mu_i x_i$ règle la dissipation, et la somme modélise le couplage global.

But : créer un cadre linéaire permettant d'étudier la convergence.

À retenir : la variable complexe contient simultanément amplitude et phase.

Écriture matricielle du modèle complexe

L'écriture matricielle permet de remplacer l'étude de la dynamique par l'étude des valeurs propres de la matrice du système.

Notations

$$x = (x_1, \dots, x_N)^T, \quad \Omega = \text{diag}(\omega_i), \quad M = \text{diag}(\mu_i).$$

Forme matricielle

$$\dot{x} = Ax, \quad A = \frac{K}{N} J_N + i\Omega - M.$$

Couplage global

$$J_N = \begin{pmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{pmatrix}$$

- ▶ J_N : interactions globales ;
- ▶ $i\Omega$: rotations propres ;
- ▶ $-M$: dissipation.

Objectif : faire émerger un mode collectif dominant.

À retenir : l'écriture matricielle transforme la question de convergence en question spectrale.

Lien avec Kuramoto : amplitude et phase

En écrivant $x_i = \rho_i e^{i\theta_i}$ et en séparant les parties réelle et imaginaire, le système complexe donne :

Amplitude

$$\dot{\rho}_i = -\mu_i \rho_i + \frac{K}{N} \sum_{j=1}^N \rho_j \cos(\theta_j - \theta_i)$$

Phase

$$\dot{\theta}_i = \omega_i + \frac{K}{N} \sum_{j=1}^N \frac{\rho_j}{\rho_i} \sin(\theta_j - \theta_i)$$

Point clé

Si les amplitudes deviennent asymptotiquement égales, alors

$$\frac{\rho_j(t)}{\rho_i(t)} \rightarrow 1$$

et l'équation de phase redonne le modèle de Kuramoto.

Ainsi, le choix des μ_i doit stabiliser les amplitudes.

Choix des μ_i : idée spectrale

Le rôle des coefficients μ_i est de sélectionner un régime collectif stable.

Matrice du système

$$A = \frac{K}{N} J_N + i\Omega - M$$

Objectif spectral

$$\lambda_1 = i\omega_m, \quad \text{Re}(\lambda_k) < 0 \quad (k \geq 2)$$

$$\omega_m = \frac{1}{N} \sum_{i=1}^N \omega_i$$

un mode collectif conservé + modes parasites amortis

Donc : à long terme, le système est gouverné par une rotation collective.

À retenir : les μ_i règlent la dissipation pour isoler le mode collectif.

Perspective spectrale : décomposition en modes

Pour un système linéaire

$$\dot{x} = Ax,$$

l'évolution peut se lire à travers les valeurs propres de la matrice A .

Décomposition en modes

Si A est diagonalisable, alors :

$$x(t) = \sum_{k=1}^N c_k e^{\lambda_k t} v_k$$

Chaque valeur propre λ_k gouverne l'évolution du mode propre v_k .

Ainsi, l'évolution globale du système est une superposition de plusieurs comportements élémentaires.

Idée : identifier quel mode disparaît et quel mode reste dominant à long terme.

À retenir : chaque valeur propre décrit l'évolution d'un mode propre.

Perspective spectrale : conditions de convergence

Le but est qu'à long terme il ne reste qu'un seul mode oscillant collectif.

Mode recherché

On veut obtenir :

$$x(t) \sim c_1 e^{i\omega_m t} v_1.$$

Ce terme décrit une rotation périodique à la fréquence moyenne ω_m .

Ainsi, on impose :

$$\lambda_1 = i\omega_m \quad \text{et} \quad \text{Re}(\lambda_k) < 0 \quad (k \geq 2).$$

Interprétation

Le premier mode est conservé, car il correspond à une rotation pure. Tous les autres modes sont amortis, car leur partie réelle est négative.

Conséquence : convergence vers un régime oscillant collectif stable.

À retenir : un mode conservé et les autres amortis donnent un régime collectif stable.

Imposer le mode collectif

On impose à la matrice A un vecteur propre représentant une rotation commune.

Vecteur propre recherché

$$v_1 = \begin{pmatrix} e^{i\varphi_1} \\ \vdots \\ e^{i\varphi_N} \end{pmatrix}$$

Condition imposée

$$Av_1 = i\omega_m v_1$$

Cette condition signifie que tous les oscillateurs tournent à la fréquence moyenne ω_m , avec des décalages de phase fixes φ_i .

Étape suivante : cette contrainte permettra de déterminer les μ_i .

À retenir : le vecteur propre impose une fréquence commune avec des décalages fixes.

Choix explicite des coefficients μ_i

La condition $A v_1 = i \omega_m v_1$ fixe les coefficients de dissipation. Le calcul détaillé est donné en annexe.

Notations utilisées

$$\omega_m = \frac{1}{N} \sum_{i=1}^N \omega_i, \quad \tilde{\omega}_i = \omega_i - \omega_m, \quad r_\infty = \frac{\gamma}{N}.$$

Résultat obtenu après calcul

$$\mu_i = \sqrt{(K r_\infty)^2 - \tilde{\omega}_i^2}$$

avec la condition d'existence :

$$|\tilde{\omega}_i| \leq K r_\infty.$$

Cette formule adapte la dissipation à l'écart de chaque oscillateur par rapport à la fréquence moyenne.

Rôle des μ_i : conserver le mode collectif et amortir les modes parasites.

À retenir : la dissipation dépend de l'écart à la fréquence moyenne.

Interprétation : vers un cycle limite collectif

Le choix des μ_i permet de conserver un mode oscillant collectif et d'amortir les autres modes.

Décomposition puis régime asymptotique

La solution se décompose sous la forme :

$$x(t) = c_1 e^{i\omega_m t} v_1 + \sum_{k=2}^N c_k e^{\lambda_k t} v_k.$$

Comme $\text{Re}(\lambda_k) < 0$ pour $k \geq 2$, les modes parasites disparaissent :

$$x(t) \sim c_1 e^{i\omega_m t} v_1.$$

Interprétation

Chaque composante vérifie :

$$x_i(t) \sim c_1 e^{i(\omega_m t + \varphi_i)}.$$

Même fréquence, décalages de phase constants.

Nuance : ici, « cycle limite » désigne un régime oscillant stable, pas un cycle limite non linéaire strictement isolé.



rotation stable

Le mouvement devient périodique dans le plan complexe : c'est le régime collectif recherché.

Théorème de convergence

On formalise maintenant le mécanisme spectral présenté précédemment.

Hypothèses

On suppose que :

$$\lambda_1 = i\omega_m$$

est une valeur propre simple de A , et que les autres valeurs propres vérifient :

$$\operatorname{Re}(\lambda_k) \leq -\alpha < 0, \quad k \geq 2.$$

Résultat

Pour toute condition initiale $x(0)$, la solution vérifie :

$$x(t) = c_1 e^{i\omega_m t} v_1 + \sum_{k=2}^N c_k e^{\lambda_k t} v_k.$$

Comme les modes $k \geq 2$ sont amortis :

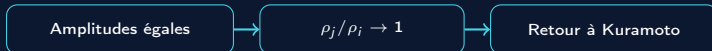
$$x(t) - c_1 e^{i\omega_m t} v_1 \longrightarrow 0.$$

Conclusion : le système converge vers un régime oscillant collectif stable.

Retour final vers Kuramoto

D'après la slide précédente, les amplitudes deviennent asymptotiquement égales :

$$\rho_i(t) \sim |c_1| \quad \Rightarrow \quad \frac{\rho_j(t)}{\rho_i(t)} \longrightarrow 1.$$



Équation de phase du modèle complexe

$$\dot{\theta}_i = \omega_i + \frac{K}{N} \sum_{j=1}^N \frac{\rho_j}{\rho_i} \sin(\theta_j - \theta_i)$$

Limite asymptotique

Comme $\rho_j / \rho_i \rightarrow 1$, on retrouve :

$$\dot{\theta}_i = \omega_i + \frac{K}{N} \sum_{j=1}^N \sin(\theta_j - \theta_i).$$

Conclusion : le modèle complexe redonne asymptotiquement le modèle de Kuramoto.

Simulations numériques : rôle et objectifs

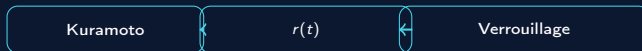
Après l'étude théorique, les simulations permettent de visualiser les mécanismes de synchronisation.

Ce que la simulation illustre

- ▶ la transition désynchronisation $\rightarrow$ synchronisation ;
- ▶ l'évolution du paramètre d'ordre $r(t)$;
- ▶ le verrouillage des différences de phase ;
- ▶ la convergence des amplitudes dans le modèle complexe.

Attention

Une simulation ne constitue pas une preuve générale : elle valide qualitativement un comportement pour un choix de paramètres.

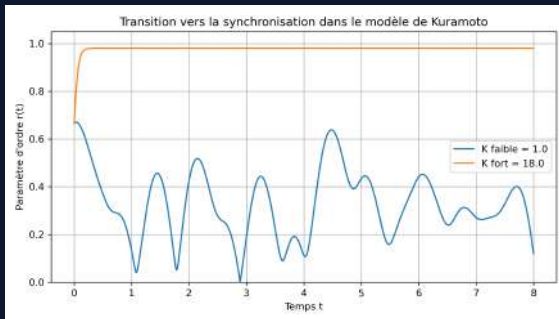


But : confronter visuellement la théorie au comportement numérique.

À retenir : les simulations illustrent la théorie sans la remplacer.

Simulation 1 : influence du couplage K

On compare l'évolution du paramètre d'ordre $r(t)$ pour deux valeurs du couplage.

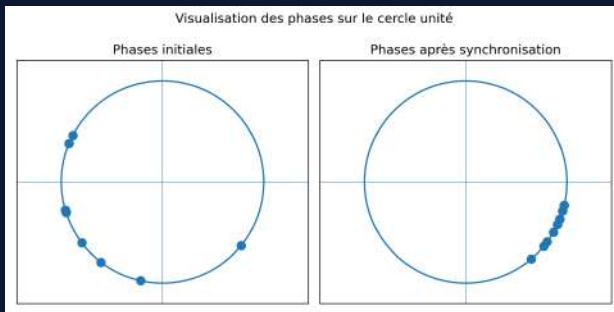


- ▶ K faible : $r(t)$ reste faible $\Rightarrow$ phases dispersées.
- ▶ K fort : $r(t)$ augmente puis se stabilise $\Rightarrow$ synchronisation.

Conclusion : le couplage déclenche la transition vers un état collectif synchronisé.

Simulation 1 : phases sur le cercle unité

Le paramètre d'ordre $r(t)$ peut être interprété géométriquement : chaque phase est un point du cercle unité.



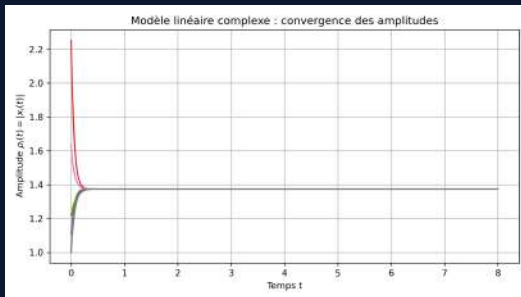
- **Avant** : les phases sont dispersées, donc leur moyenne complexe est faible.
- **Après** : les phases se regroupent, donc $r(t)$ augmente.

Conclusion : la synchronisation correspond à une concentration des phases.

Simulation 2 : convergence des amplitudes

On simule le modèle linéaire complexe :

$$\dot{x}_i = (i\omega_i - \mu_i)x_i + \frac{K}{N} \sum_{j=1}^N x_j.$$



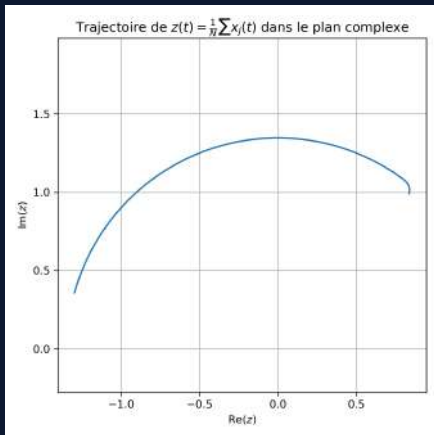
- Les amplitudes $\rho_i(t) = |x_i(t)|$ évoluent d'abord différemment.
- Après le régime transitoire, elles convergent vers une valeur commune.

Conclusion : $\rho_j(t)/\rho_i(t) \rightarrow 1$, ce qui permet le retour vers Kuramoto.

Simulation 2 : trajectoire collective dans le plan complexe

On observe l'état moyen du système :

$$z(t) = \frac{1}{N} \sum_{j=1}^N x_j(t).$$



Observation

La trajectoire de $z(t)$ devient régulière dans le plan complexe.

Interprétation

Cette trajectoire traduit l'apparition d'une rotation collective stable.

Conclusion :

La simulation illustre la convergence vers un régime oscillant collectif.

Discussion : validité, limites et cohérence

Les simulations numériques illustrent les mécanismes établis théoriquement, mais elles ne constituent pas une preuve générale.

Cohérence avec la théorie

- ▶ $r(t)$ augmente lorsque le couplage K est suffisant.
- ▶ Les phases se regroupent sur le cercle unité.
- ▶ Les amplitudes $\rho_i(t)$ se stabilisent.
- ▶ L'observable $z(t)$ tend vers une rotation collective.

Limites des simulations

- ▶ Elles testent un choix particulier de paramètres.
- ▶ Elles ne prouvent pas la convergence pour tout système.
- ▶ Les résultats dépendent de K , des ω_i et des conditions initiales.

Bilan : les simulations confirment qualitativement l'amortissement des modes parasites, la stabilisation des amplitudes et l'apparition d'un régime oscillant collectif.

Expérience Arduino : principe

L'objectif est de visualiser expérimentalement le verrouillage de phase à partir d'une version numérique simplifiée du modèle de Kuramoto.

Modèle implémenté sur Arduino

$$\dot{\theta}_i = \omega_i + \frac{K}{N} \sum_{j=1}^N \sin(\theta_j - \theta_i), \quad N = 4.$$

Principe

L'Arduino calcule quatre phases internes $\theta_i(t)$.

Chaque phase est ensuite transformée en signal carré observable sur les sorties :

$D8, D9, D10, D11$.

But : observer l'effet du couplage K sur la synchronisation des signaux.

À retenir : Arduino donne une visualisation numérique embarquée du verrouillage de phase.

Montage et protocole expérimental

Montage réel

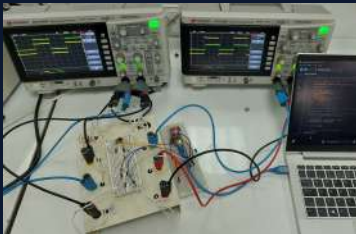
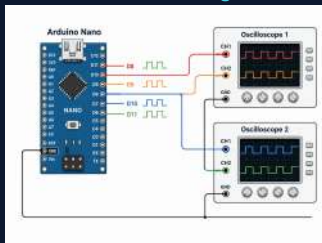


Schéma de câblage



Montage

- ▶ Arduino Nano ;
- ▶ sorties : D8–D11 ;
- ▶ 2 oscilloscopes à 2 voies ;
- ▶ masse commune : GND.

Protocole

$$K = 0$$

oscillateurs indépendants.

$$K > 0$$

oscillateurs couplés.

But : comparer les décalages de phase.

Résultat expérimental : cas $K = 0$



Sans couplage

$$K = 0$$

Les oscillateurs évoluent indépendamment :

$$\dot{\theta}_i = \omega_i.$$

Observation

Les signaux dérivent les uns par rapport aux autres.
Les différences de phase ne restent pas constantes.

Conclusion : pas de verrouillage de phase.

Résultat expérimental : cas $K > 0$



Avec couplage

$$K > 0$$

Les phases interagissent selon le modèle de Kuramoto :

$$\dot{\theta}_i = \omega_i + \frac{K}{N} \sum_{j=1}^N \sin(\theta_j - \theta_i).$$

Observation

Les décalages deviennent plus stables.

Les signaux tendent vers un verrouillage de phase.

Conclusion : le couplage favorise la synchronisation.

Discussion de l'expérience Arduino

L'expérience Arduino permet de visualiser le mécanisme de verrouillage de phase, mais elle reste une version simplifiée du phénomène réel.

Ce que l'expérience valide

- ▶ Le rôle du couplage K .
- ▶ Le passage d'une dérive des phases à des décalages plus stables.
- ▶ Le lien entre l'équation de Kuramoto et un signal observable.

Limites

- ▶ Les oscillateurs sont numériques, non biologiques.
- ▶ Le couplage est calculé par programme.
- ▶ L'expérience illustre le modèle, mais ne remplace pas la preuve théorique.

Bilan : l'Arduino donne une visualisation expérimentale simple de la synchronisation étudiée dans le modèle de Kuramoto.

Pourquoi contrôler la synchronisation neuronale ?

Synchroniser

Certains rythmes cérébraux coordonnés sont utiles au fonctionnement du cerveau : attention, mémoire, sommeil ou traitement de l'information. Dans ce cas, favoriser la synchronisation peut aider à renforcer une activité collective efficace.

Désynchroniser

À l'inverse, une synchronisation trop forte peut devenir pathologique. Elle intervient notamment dans des troubles comme l'épilepsie, la maladie de Parkinson ou certains tremblements. L'objectif est alors de réduire cette synchronisation excessive.

Intérêt du modèle

Le modèle étudié permet de comprendre le rôle du couplage entre oscillateurs. En agissant sur ce couplage, on peut interpréter comment un réseau neuronal peut être amené vers un état synchronisé ou désynchronisé.

rythme utile $\Rightarrow$ synchroniser

rythme pathologique $\Rightarrow$ désynchroniser

Annexe 1 — Calcul détaillé des coefficients μ_i

Condition imposée

$$A v_1 = i \omega_m v_1, \quad A = \frac{K}{N} J_N + i \Omega - M.$$

On pose $\gamma = \sum_{j=1}^N e^{i\varphi_j}$. Comme $J_N v_1 = \gamma(1, \dots, 1)^T$, la coordonnée i donne :

$$\frac{K\gamma}{N} + i\omega_i e^{i\varphi_i} - \mu_i e^{i\varphi_i} = i\omega_m e^{i\varphi_i}.$$

Relation utile

Avec $\tilde{\omega}_i = \omega_i - \omega_m$,

$$(\mu_i - i\tilde{\omega}_i) e^{i\varphi_i} = \frac{K\gamma}{N}.$$

En multipliant par $e^{-i\varphi_i}$:

$$\mu_i - i\tilde{\omega}_i = \frac{K\gamma}{N} (\cos \varphi_i - i \sin \varphi_i).$$

Donc :

$$\mu_i = \frac{K\gamma}{N} \cos \varphi_i, \quad \tilde{\omega}_i = \frac{K\gamma}{N} \sin \varphi_i.$$

Annexe 1 — Fin du calcul des μ_i

Passage à r_∞

On choisit l'origine des phases de sorte que γ soit réel positif et on pose

$$r_\infty = \frac{\gamma}{N}.$$

La relation imaginaire devient :

$$\tilde{\omega}_i = Kr_\infty \sin \varphi_i, \quad \sin \varphi_i = \frac{\tilde{\omega}_i}{Kr_\infty}.$$

Condition d'existence :

$$|\tilde{\omega}_i| \leq Kr_\infty.$$

Formule finale

À partir de $\mu_i = Kr_\infty \cos \varphi_i$ et $\cos^2 \varphi_i = 1 - \sin^2 \varphi_i$,

$$\mu_i = Kr_\infty \sqrt{1 - \left(\frac{\tilde{\omega}_i}{Kr_\infty} \right)^2}$$

donc :

$$\mu_i = \sqrt{(Kr_\infty)^2 - \tilde{\omega}_i^2}.$$

On retient la racine positive pour obtenir une dissipation stabilisante.

Annexe 2 — Pourquoi une valeur propre $i\omega_m$?

Régime périodique recherché

Pour le système linéaire

$$\dot{x} = Ax,$$

on cherche un mode collectif de la forme

$$x(t) = e^{i\omega t} v.$$

Calcul direct

En dérivant :

$$\dot{x}(t) = i\omega e^{i\omega t} v.$$

D'autre part :

$$Ax(t) = e^{i\omega t} Av.$$

Comme $\dot{x} = Ax$, on obtient :

$$Av = i\omega v.$$

Ainsi, une rotation périodique non amortie correspond à une valeur propre imaginaire pure. Pour une rotation collective à la fréquence moyenne, on impose donc :

$$\lambda_1 = i\omega_m.$$

Annexe 3 — Code Python des simulations I

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 # Modele de Kuramoto
5 def order_parameter(theta):
6     z = np.mean(np.exp(1j * theta))
7     return np.abs(z), np.angle(z)
8
9 def kuramoto_rhs(theta, omega, K):
10     N = len(theta)
11     diff = theta[None, :] - theta[:, None]
12     return omega + (K / N) * np.sum(np.sin(diff), axis=1)
13
14 def simulate_kuramoto(omega, K, theta0, T=8.0, dt=1e-3):
15     steps = int(T / dt)
16     t = np.linspace(0, T, steps)
17     theta = theta0.copy()
18     R = np.zeros(steps)
19     THETA = np.zeros((steps, len(theta0)))
20
21     for k in range(steps):
22         R[k], _ = order_parameter(theta)
23         THETA[k] = theta
24
25         k1 = kuramoto_rhs(theta, omega, K)
26         k2 = kuramoto_rhs(theta + 0.5*dt*k1, omega, K)
27         k3 = kuramoto_rhs(theta + 0.5*dt*k2, omega, K)
28         k4 = kuramoto_rhs(theta + dt*k3, omega, K)
29
30         theta += (dt/6.0)*(k1 + 2*k2 + 2*k3 + k4)
31
32     return t, R, THETA
33
34 # Modele lineaire complexe
35 def consistency_r(r, omega_centered, K):
36     if r <= 1e-12:
37         return np.nan
38     s = omega_centered / (K*r)
39     if np.any(np.abs(s) > 1):
40         return np.nan
```

Annexe 3 — Code Python des simulations II

```
1 return np.mean(np.sqrt(1 - s**2))
2
3 def solve_r_infty(omega, K, r0=0.8, itmax=5000, tol=1e-12):
4     omega_m = np.mean(omega)
5     omega_centered = omega - omega_m
6     r = r0
7
8     for _ in range(itmax):
9         r_new = consistency_r(r, omega_centered, K)
10        if np.isnan(r_new):
11            return np.nan
12        if abs(r_new-r) < tol:
13            return r_new
14        r = 0.5*r + 0.5*r_new
15
16    return r
17
18 def complex_rhs(x, omega, mu, K):
19     N = len(x)
20     return (1j*omega - mu)*x + (K/N)*np.sum(x)
21
22 def simulate_complex_model(omega, K, T=8.0, dt=1e-3):
23     steps = int(T/dt)
24     t = np.linspace(0, T, steps)
25
26     omega_m = np.mean(omega)
27     omega_centered = omega - omega_m
28
29     r_infty = solve_r_infty(omega, K)
30     mu = np.sqrt((K*r_infty)**2 - omega_centered**2)
31
32     x = (np.random.randn(len(omega)) + 1j*np.random.randn(len(omega)))*0.3 + (1+1j)
33
34     RHO = np.zeros((steps, len(omega)))
35     Z = np.zeros(steps, dtype=complex)
36
37     for k in range(steps):
38         RHO[k] = np.abs(x)
39         Z[k] = np.mean(x)
40
41         k1 = complex_rhs(x, omega, mu, K)
```

Annexe 3 — Code Python des simulations III

```
2     k2 = complex_rhs(x + 0.5*dt*k1, omega, mu, K)
3     k3 = complex_rhs(x + 0.5*dt*k2, omega, mu, K)
4     k4 = complex_rhs(x + dt*k3, omega, mu, K)
5
6     x += (dt/6.0)*(k1 + 2*k2 + 2*k3 + k4)
7
8     return t, RH0, Z, r_infty, mu
9
10 # Parametres
11 np.random.seed(0)
12 N = 8
13 omega = np.array([-6.0, -3.0, -2.0, 0.0, 1.5, 2.5, 4.0, 5.0])
14 theta0 = 2*np.pi*np.random.rand(N)
15
16 K_weak = 1.0
17 K_strong = 18.0
18
19 # Simulation Kuramoto
20 t, R_weak, THETA_weak = simulate_kuramoto(omega, K_weak, theta0)
21 _, R_strong, THETA_strong = simulate_kuramoto(omega, K_strong, theta0)
22
23 plt.figure()
24 plt.plot(t, R_weak, label="K faible")
25 plt.plot(t, R_strong, label="K fort")
26 plt.xlabel("Temps")
27 plt.ylabel("r(t)")
28 plt.title("Transition vers la synchronisation")
29 plt.legend()
30 plt.grid()
31 plt.savefig("fig1_kuramoto_comparaison_r.png", dpi=300)
32 plt.show()
33
34 # Cercle unite
35 theta_initial = THETA_strong[0]
36 theta_final = THETA_strong[-1]
37
38 angles = np.linspace(0, 2*np.pi, 400)
39 plt.figure(figsize=(8,4))
40
41 plt.subplot(1,2,1)
42 plt.plot(np.cos(angles), np.sin(angles))
```

Annexe 3 — Code Python des simulations IV

```
3 plt.scatter(np.cos(theta_initial), np.sin(theta_initial))
4 plt.axis("equal")
5 plt.title("Phases initiales")
6
7 plt.subplot(1,2,2)
8 plt.plot(np.cos(angles), np.sin(angles))
9 plt.scatter(np.cos(theta_final), np.sin(theta_final))
0 plt.axis("equal")
1 plt.title("Après synchronisation")
2
3 plt.savefig("fig2_kuramoto_cercle_phases.png", dpi=300)
4 plt.show()
5
6 # Simulation complexe
7 t_complex, RHO, Z, r_infty, mu = simulate_complex_model(omega, K_strong)
8
9 plt.figure()
0 for i in range(N):
1     plt.plot(t_complex, RHO[:, i])
2 plt.xlabel("Temps")
3 plt.ylabel("Amplitude rho_i(t)")
4 plt.title("Convergence des amplitudes")
5 plt.grid()
6 plt.savefig("fig3_complexe_amplitudes.png", dpi=300)
7 plt.show()
8
9 plt.figure()
0 plt.plot(np.real(Z), np.imag(Z))
1 plt.xlabel("Re(z)")
2 plt.ylabel("Im(z)")
3 plt.title("Trajectoire collective z(t)")
4 plt.axis("equal")
5 plt.grid()
6 plt.savefig("fig4_complexe_z_plan.png", dpi=300)
7 plt.show()
```

Annexe 4 — Code Arduino de l'expérience I

```
1 // 4 oscillateurs de Kuramoto avec Arduino Nano
2 // Sorties : D8, D9, D10, D11
3
4 const int N = 4;
5 const int pins[N] = {8, 9, 10, 11};
6
7 float theta[N] = {0.0, 1.2, 2.3, 4.0};
8
9 float omega[N] = {
10     2.0 * PI * 1.00,
11     2.0 * PI * 1.07,
12     2.0 * PI * 0.94,
13     2.0 * PI * 1.12
14 };
15
16 float K = 0.0;
17 float dt = 0.01;
18
19 unsigned long previousTime = 0;
20 int printCounter = 0;
21
22 void setup() {
23     Serial.begin(9600);
24
25     for (int i = 0; i < N; i++) {
26         pinMode(pins[i], OUTPUT);
27     }
28
29     Serial.println("4 oscillateurs de Kuramoto");
30     Serial.println("Envoyer K : 0, 3, 5 ou 8");
31 }
32
33 void loop() {
34
35     if (Serial.available() > 0) {
36         String input = Serial.readStringUntil('\n');
37         input.trim();
38
39         if (input.length() > 0) {
40             K = input.toFloat();
41         }
42     }
43 }
```

Annexe 4 — Code Arduino de l'expérience II

```
1     Serial.print("Nouveau K = ");
2     Serial.println(K);
3 }
4 }
5
6 unsigned long currentTime = millis();
7
8 if (currentTime - previousTime >= 10) {
9     previousTime = currentTime;
10
11     float dtheta[N];
12
13     for (int i = 0; i < N; i++) {
14         float coupling = 0.0;
15
16         for (int j = 0; j < N; j++) {
17             coupling += sin(theta[j] - theta[i]);
18         }
19
20         dtheta[i] = omega[i] + (K / N) * coupling;
21     }
22
23     for (int i = 0; i < N; i++) {
24         theta[i] += dtheta[i] * dt;
25
26         while (theta[i] >= 2.0 * PI) {
27             theta[i] -= 2.0 * PI;
28         }
29
30         while (theta[i] < 0.0) {
31             theta[i] += 2.0 * PI;
32         }
33     }
34
35     for (int i = 0; i < N; i++) {
36         if (sin(theta[i]) >= 0) {
37             digitalWrite(pins[i], HIGH);
38         } else {
39             digitalWrite(pins[i], LOW);
40         }
41     }
42 }
```

Annexe 4 — Code Arduino de l'expérience III

```
2   printCounter++;
3
4
5   if (printCounter >= 20) {
6       printCounter = 0;
7
8       Serial.print("K = ");
9       Serial.print(K);
10      Serial.print(" | theta = ");
11
12      for (int i = 0; i < N; i++) {
13          Serial.print(theta[i], 3);
14          Serial.print(" ");
15      }
16
17      Serial.println();
18  }
19  }
20 }
```