Synchronisation

Simulation numérique de la synchronisation d'oscillateurs

Stage au Laboratoire de Physique Théorique et Modélisation.

Ferdinand Tixidre

Cergy Paris Université - Institut des sciences et techniques

CERGY PARIS UNIVERSITÉ

LPTM
laboratoire de physique théorique et modélisation

Synchronisation

Simulation numérique de la synchronisation d'oscillateurs

par

Ferdinand Tixidre

Numéro étudiant
22112383

Responsables :	Alessandro Torcini
	Matteo Di Volo
Institution :	Cergy Paris Université Institut des Sciences et Techniques
Structure :	Laboratoire de Physique Théorique et Modélisation
Durée du stage :	26 avril - 25 juin 2021

Image de couverture : Un banc de poisson nageant de manière coordonnée. Brent Storm / Unsplash.

CERGY PARIS UNIVERSITÉ

Préface

Tout d'abord j'adresse mes remerciements à François Germinet, président de CY Cergy Paris Université, et Monsieur Jean Avan, directeur du Laboroire de Physique Théorique et Modélisation, d'avoir rendu ce stage possible.

J'aimerais aussi remercier Monsieur Guy Trambly de Laissardière, responsable du master de physique, et toute l'équipe pédagogique pour la qualité des enseignements en cette année particulière, pour leur écoute et leurs conseils tout au long de l'année.

Enfin je tiens aussi à remercier vivement les responsables de ce stage, Alessandro Toricini et Matteo Di Volo pour leur accueil et le partage de leur expertise. Grâce à eux j'ai pu apprendre beaucoup pendant ces deux mois.

Ce stage de fin de Master 1 s'est déroulé dans le cadre de restrictions liées au contexte sanitaire, ainsi le travail a été réalisé à distance. L'objet du stage était d'étudier les phénomènes de synchronisation avec le modèle de Kuramoto. J'ai découvert l'existence de ces phénomènes en cours et lors de conférence de vulgarisation, lorsque l'opportunité d'étudier le sujet en profondeur s'est présentée j'ai immédiatement été intéressé. Ce stage m'a permis de développer mes compétences en programmation, dans les languages C++ et python, et a constitué une excellente introduction aux systèmes chaotiques.

Ferdinand Tixidre
Paris, Juin 2021

Sommaire

Les phénomènes de synchronisation sont des cas où un ordre semble émerger du chaos. Ils apparaissent dès que des oscillateurs sont en interaction. Chaque oscillateurs au sein d'une population est doté d'une fréquence naturelle et est en interaction avec les autres selon une valeur de couplage.

Ce rapport s'organise de façon chronologique en partant du point de vue d'une personne qui découvre le sujet. On commencera d'abord par une vulgarisation du phénomène de synchronisation en donnant quelques exemples. Puis dans la partie 1.2 on détaillera le cadre mathématique où l'on introduira un modèle pour décrire le phénomène de synchronisation. Ensuite nous détaillerons la phénoménologie de ce modèle dans le cas de la distribution de fréquence la plus simple 2.1, évolution du système en fonction du temps 2.2, mise en évidence des différents régimes 2.3 et influence de la taille de la population 2.4.

Dans la seconde moitié du rapport on s'intéresse au cas d'une distribution de fréquence bimodale 3.1, nous réaliserons des simulations numériques permettant de retrouver les différentes transitions de phases 3.2, 3.3 puis nous tracerons le diagramme de phase du système 3.4 et mettrons en évidence une région de bistabilité 3.5.

Table des matières

Table des figures

1

Introduction

1.1. Motivation

En 1656 le physicien néerlandais Christian Huygens crée la première horloge à pendule, dans le but de faciliter la navigation en mer. Ces horloges ne se décalaient que de quelques secondes par jours et permettaient donc de mesurer la longitude d'un navire assez précisément. Son idée était de fixer deux horloges, à une lourde masse suspendue pour amortir le roulis de la mer. Mais en testant son dispositif il remarqua que les deux horloges se synchronisaient spontanément et compris que ce phénomène se produisait parce que les horloges étaient suspendus à la même poutre.

Ce phénomène de synchronisation observé par Huygens est l'un des plus universel de la nature. Il s'étend de l'échelle subatomique à l'échelle cosmique et traduit une tendance profonde vers l'ordre de la nature, qui peut sembler s'opposer à l'entropie.

Quelques belles illustrations de ce phénomène dans la nature sont par exemples les murmurations d'étourneaux, les bancs de poissons ou les vers luisant d'Asie du Sud-Est qui clignotent en phase [4]. Mais des objets inanimés peuvent aussi se synchroniser, les laser sont le résultat de synchronisation à l'échelle atomique, un exemple à l'échelle de notre système solaire est la division de Cassini dans les anneaux de Saturne, causé par l'interaction gravitationnelle avec la lune Mimas.

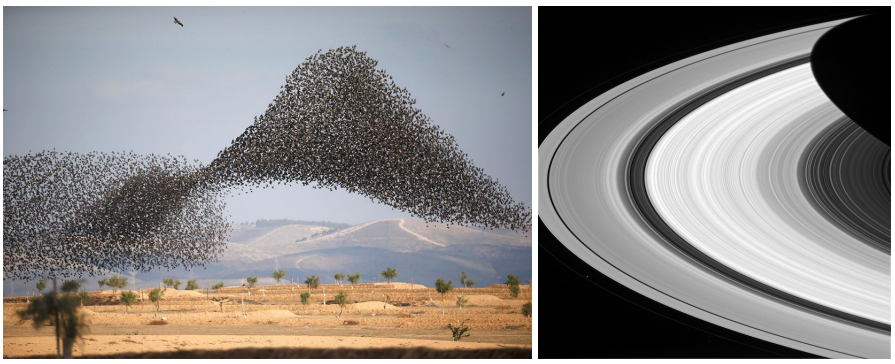


Figure 1.1 : Deux illustrations du phénomène de synchronisation. À gauche : murmuration d'étourneaux, à droite : les anneaux de Saturne et la division de Cassini.

L'omniprésence et la complexité de ce phénomène a incité les scientifiques à développer une approche mathématique pour en comprendre les principes fondamentaux. Les premiers travaux majeurs vinrent de Arthur Taylor Winfree [6] qui comprit l'effet de seuil du phénomène : la synchronisation ne se produit que si le couplage entre les oscillateurs est suffisamment fort. Le modèle proposé par Winfree était difficile à résoudre analytiquement mais inspira Yoshiki Kuramoto qui établit en 1975 un modèle plus facilement utilisable [1]. Ce modèle est aujourd'hui utilisé dans de nombreux domaines, nous étudierons, par simulations numériques, les différents phénomènes qui se produisent en fonction des paramètres du système.

1.2. Le modèle de Kuramoto

Considérons un grand nombre d'oscillateurs N, disposés sur un cercle unitaire. Sans interaction entre-eux les oscillateurs tourne autour du cercle, de façon indépendantes, à leurs fréquences naturelles :

$$\dot{\varphi}_i(t) = \omega_i, \;\; i = 1, ..., N \tag{1.1}$$

À chaque oscillateur i on associe la phase φ_i et la fréquence naturelle ω_i. Les fréquences ω_i sont distribués selon une densité de probabilité $\omega \mapsto g(\omega)$.

Pour que la synchronisation se produise il faut que les oscillateurs soient couplés d'une façon ou d'une autre. Kuramoto comprit qu'un champ moyen de couplage serait plus simple à modeliser mathématiquement. Dans ce cas tous les oscillateurs sont en interactions entre eux, avec une intensité qui est la même pour chaque paire d'oscillateurs. À partir de cela Kuramoto proposa l'équation suivante qui gouverne la dynamique du système [1] [2] :

$$\dot{\varphi}_i(t) = \omega_i + \frac{K}{N} \sum_{j=1}^{N} sin(\varphi_j(t) - \varphi_i(t)) \tag{1.2}$$

K est la constante de couplage et $1/N$ est le facteur de normalisation.

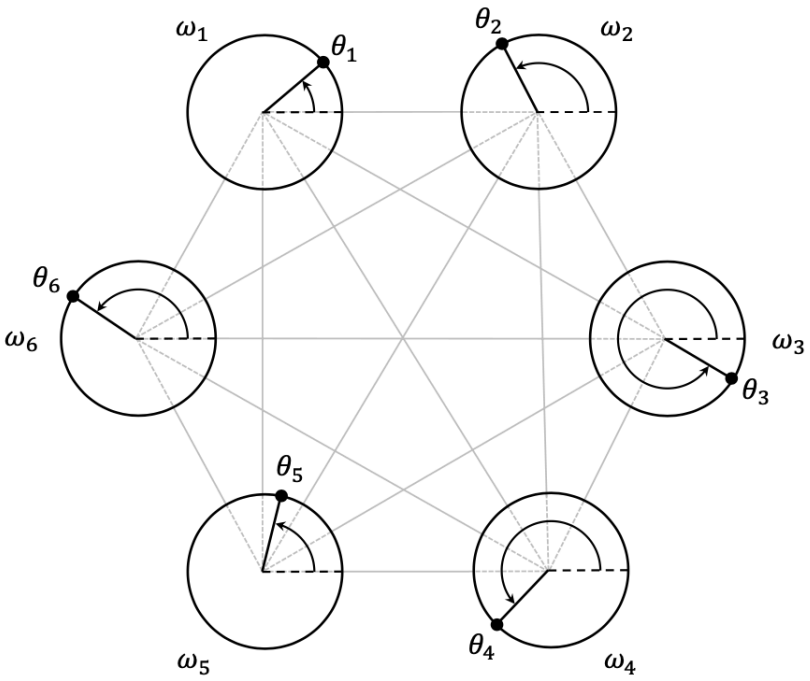


Figure 1.2 : Illustration du couplage entre tous les oscillateurs dans le modèle de Kuramoto pour $N = 6$.

Dans l'équation (1.2) on voit que les oscillateurs ont tendance à osciller à leur fréquence naturelle mais le terme d'interaction tends à les faire osciller ensemble, quand $K > 0$.

Il est naturel de s'attendre à ce que la synchronisation entre oscillateurs dépende de la distributions des fréquences naturelles. Dans le cas d'une distribution de fréquence symétrique et unimodale (par exemple une gaussienne) la synchronisation n'a pas lieu si l'écart type est trop grand comparé à K. Dans ces cas les oscillateurs sont trop différents les uns des autres, la synchronisation est difficiles à obtenir. Les oscillateurs tournent à leur fréquence naturelle. En augmentant la valeur de K cela reste vrai jusqu'à une valeur critique K_c, à partir de ce seuil une portion de la population d'oscillateurs se synchronise (ce dont la fréquence naturelle est proche de la moyenne de la distribution). En augmentant encore K on augmente la proportion d'oscillateurs synchronisés. La valeur critique de couplage sépare donc deux régimes dans lesquels le système est soit incohérent soit partiellement synchronisé.

Pour quantifier le degré de synchronisation du système nous pouvons introduire le paramètre d'ordre

$$Re^{i\theta} = \frac{1}{N}\sum_{j=1}^{N} e^{i\varphi_j} \tag{1.3}$$

qui est la moyenne empirique qui décrit la dynamique collective de la population. $R(t)$ est comprit entre 0 et 1 mesure la cohérence de phase des oscillateurs et θ la phase moyenne. Par exemple quand les phases sont plus ou moins uniformément répartis on a $R \simeq 0$ alors que quand $R \simeq 1$ la population est regroupée (voir figure 1.3). Le système est parfaitement synchronisé quand $R \equiv 1$.

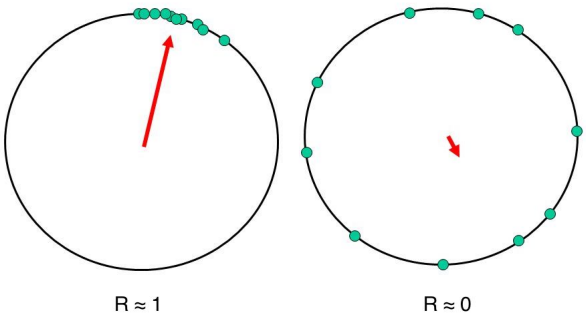


Figure 1.3 : Illustration du paramètre d'ordre R dans le cas d'un état partiellement synchronisé et d'un état incohérent. $N = 10$

Ce paramètre d'ordre permet de réécrire (1.2) d'une façon qui sera plus pratique. Pour cela on va multiplier (1.3) par $e^{i\varphi_i}$ $(i = 1, ..., N)$ pour obtenir

$$Re^{i(\theta-\varphi_i)} = \frac{1}{N}\sum_{j=1}^{N} e^{i(\varphi_j-\varphi_i)}, \;\; i = 1, ..., N. \tag{1.4}$$

La partie imaginaire de (1.4) est

$$Rsin(\theta - \varphi_i) = \frac{1}{N}\sum_{j=1}^{N} sin(\varphi_j - \varphi_i), \;\; i = 1, ..., N. \tag{1.5}$$

on va injecter ce résultat dans (1.2). Ainsi (1.2) devient

$$\dot{\varphi}_i = \omega_i + KRsin(\theta - \varphi_i), \;\; i = 1, ..., N. \tag{1.6}$$

qui met en évidence le fait que le terme d'interaction dans (1.6) tends à rapprocher les phases ϕ_i de θ. Les phénomènes de synchronisation sont donc le résultat d'un effet boule de neige : quand la cohérence augmente, la valeur de R augmente aussi ce qui conduit à une augmentation du facteur KR dans (1.6) et donc à une augmentation de la cohérence.

2

Phénoménologie dans le modèle de Kuramoto

Dans ce chapitre nous allons passer en revue différents phénomènes prédits par le modèle de Kuramoto. Les fréquences naturelles des oscillateurs seront distribués selon une fonction gaussienne. Après avoir définis les conditions initiales dans la partie 2.1, nous étudierons l'évolution de R en fonction du temps en section 2.2. Dans la partie 2.3 nous chercherons à déterminer la valeur de couplage critique puis dans la dernière partie 2.4 nous observerons l'influence du nombre d'oscillateurs.

2.1. Conditions initiales

Tout comme Kuramoto nous allons utiliser une distribution g symétrique, unimodale et centrée en zéro. Plus précisement les fréquences naturelles des N oscillateurs seront distribués selon une loi gaussienne de moyenne ω_0 et d'écart-type σ.

$$g(\omega) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\omega-\omega_0)^2}{2\sigma^2}} \tag{2.1}$$

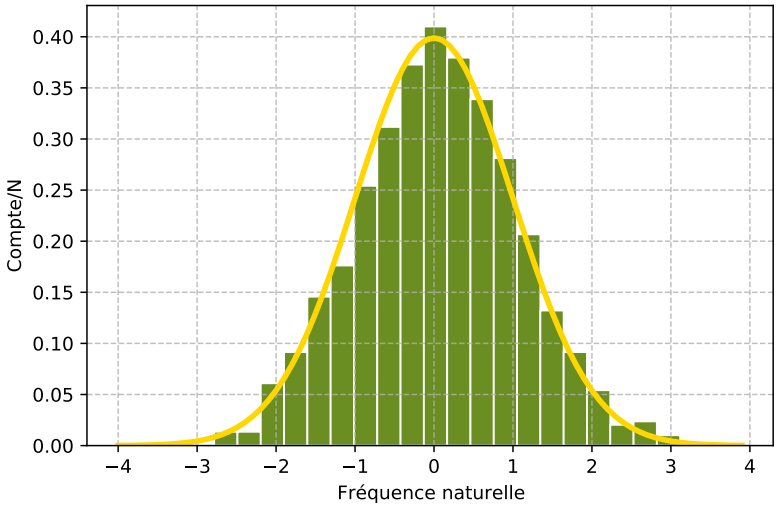


Figure 2.1 : Les fréquences de $N = 1000$ oscillateurs distribuées selon une loi gaussienne avec $\omega_0 = 0$ et $\sigma = 1$. En histogramme les valeurs obtenues dans la simulation, comparées à la distribution théorique.

Dans la suite, sauf mention contraire on utilisera $\omega_0 = 0$ et $\sigma = 1$. Les phases initiales φ sont distribuées aléatoirement entre 0 et 2π, selon une distribution uniforme.

$$\varphi(t = 0) \sim U(0, 2\pi) \tag{2.2}$$

2.2. Paramètre d'ordre en fonction du temps

Regardons maintenant avec plus de détails comment la valeur de R révèle le processus de synchronisation du système. Dans la simulation 2.2 la dynamique du système est calculé en intégrant numériquement l'équation (1.6) avec la méthode d'Euler. On utilise un pas d'intégration $\delta t = 10^-3$ et on simule pendant au moins un temps $T = 50$.

Pour des valeurs de K inférieures à un seuil K_c les oscillateurs ne semblent pas ressentir les interactions mutuelles, ils oscillent à leurs fréquences naturelles et le système est incohérent. À chaque distribution initiale des phases, elles sont uniformément réparties entre 0 et 2π donc la phase moyenne θ est presque nulle.

Pour des valeurs de K supérieures à K_c on voit que R augmente rapidement et se stabilise vers une valeur supérieure à zéro. Deux groupes se forment, les oscillateurs dont la fréquence naturelle est suffisamment éloignée de ω_0 continuent d'osciller à un rythme proche de leur fréquence naturelle, mais ceux avec une fréquence naturelle proche de ω_0 vont verrouillés leurs phases avec θ

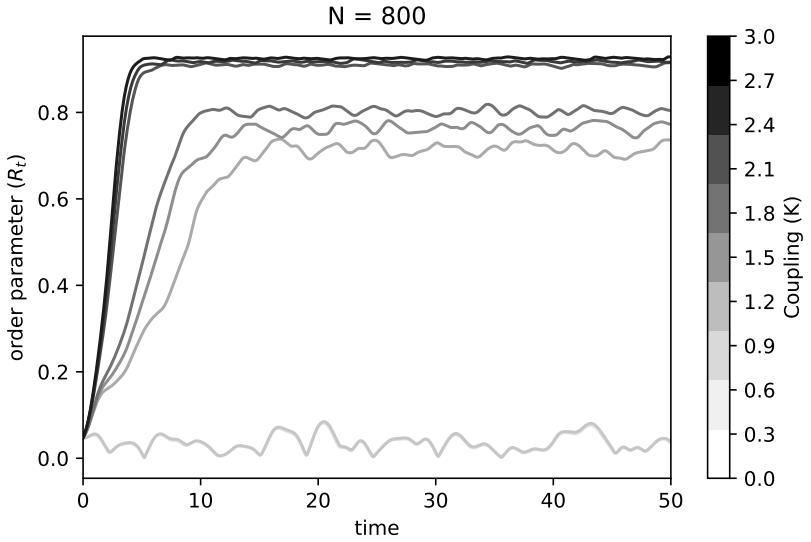


Figure 2.2 : Évolution du paramètre d'ordre en fonction du temps, pour différentes valeurs de couplage. On utilise un temps d'évolution $T = 50$, un pas $\delta t = 10^{-3}$. Par soucis de lisibilité on ne représente pas toutes les courbes, $\delta K = 0.3$.

La valeur autour de laquelle R se stabilise augmente avec K et tend vers 1 quand $K \to +\infty$.

2.3. Couplage critique

Comme démontré dans [5] on peut prédire analytiquement K_c pour cette distribution gaussienne particulière :

$$K_c = \frac{2}{\pi g(\omega_0)}, \quad g(\omega_0) = \frac{1}{\sqrt{2\pi}\sigma} \tag{2.3}$$

Avec notre distribution (voir 2.1) on obtiens $K_c = \sqrt{\frac{8}{\pi}} \simeq 1.6$. Pour chaque K on va donc calculer la valeur de R moyenné sur un intervalle de temps où le système est stable. Sur la simulation 2.3 on voit que R n'augmente qu'à partir d'une valeur seuil qui correspond à notre prédiction analytique de K_c (2.3).

On peut ensuite montrer que R varie avec $\sqrt{K - K_c}$ pour $K \geq K_c$ (voir figure 2.4)

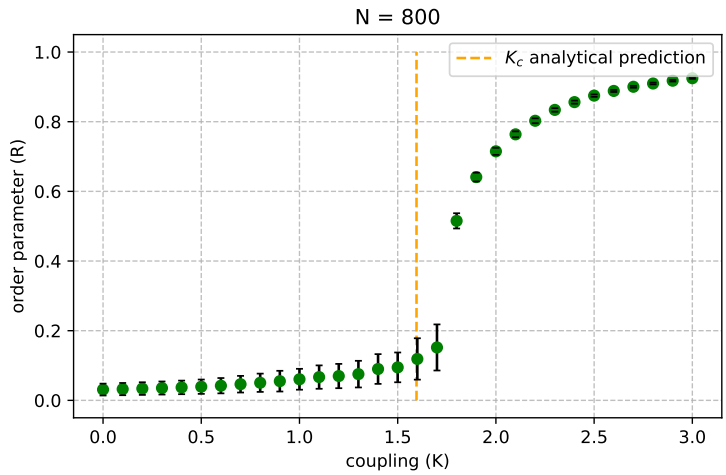


Figure 2.3 : Évolution de R moyen en fonction du couplage K, avec en pointillé la prédiction analytique obtenues en (2.3). Pour cette simulation et les suivantes (sauf mention contraire) on utilisera un temps de simulation $T = 100$ et $\delta t = 10^{-3}$. On calcule R en moyennant les valeurs une fois le système stabilisé, c'est à dire après un temps $t_{transient} = 20$. $R_{mean} = 1/((T - t_{transient})/\delta t) \cdot \sum R$. Les barres d'erreur représente l'écart-type par rapport au temps. Le code C++ correspondant est fournie en annexe B.

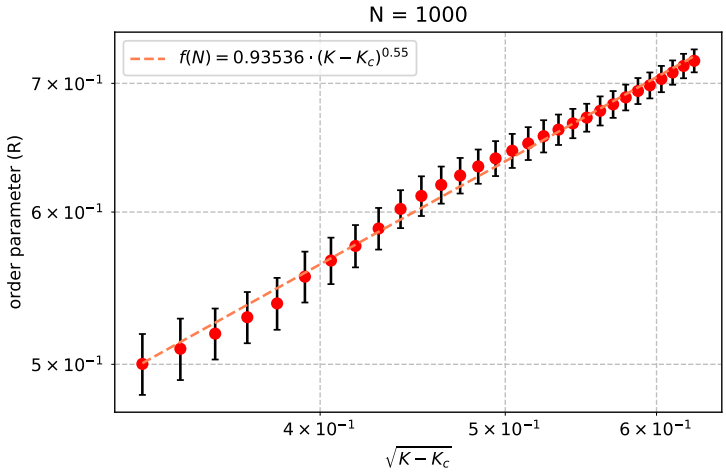


Figure 2.4 : Paramètre R en fonction de $\sqrt{K - K_c}$, échelle logarithmique. Les données sont ajustées avec une fonction de la forme $a(K - K_c)^b$ en utilisant la valeur analytique de K_c, tracée en pointillé.

2.4. Influence du nombre d'oscillateurs

On peut montrer avec le théorème central limite que quand N est fini, $R \propto 1/\sqrt{N}$. En effet si on considère une suite de variables aléatoires indépendantes $X_1, X_2, ..., X_n$ d'espérance μ et d'écart-type σ. La moyenne de cette série de variable est

$$S = \frac{1}{N} \sum_{j=1}^{N} X_j \tag{2.4}$$

et pour de grandes valeurs de N, S suit une loi normale $\mathcal{N}(\mu, \frac{\sigma}{\sqrt{N}})$. Dans notre cas, selon (1.3) R est égal à la racine carrée de deux moyennes.

$$R = \frac{1}{N} \sqrt{(\sum_{j=1}^{N} cos\varphi_j)^2 + (\sum_{j=1}^{N} sin\varphi_j)^2} \tag{2.5}$$

Ces deux quantités $\frac{1}{N}\sum_{j=1}^{N} cos\varphi_j$ et $\frac{1}{N}\sum_{j=1}^{N} sin\varphi_j$ sont distribuées selon une gaussienne centrée en 0 avec un écart-type proportionnel à $1/\sqrt{N}$. D'où

$$R \approx \sqrt{(0+\frac{\sigma}{\sqrt{N}})^2 + (0+\frac{\sigma}{\sqrt{N}})^2} \approx \sqrt{\frac{2\sigma}{N}} \tag{2.6}$$

$$R \propto \frac{1}{\sqrt{N}}$$

Ainsi si le nombre d'oscillateurs est multiplié par 4 on devrait trouver un paramètre d'ordre R divisé par 2. Et effectivement sur la figure 2.5 on peut voir que dans le régime asynchrone les valeurs de R diminue quand N augmente. Alors qu'elles ne dépendent pas de N quand $K > K_c$.

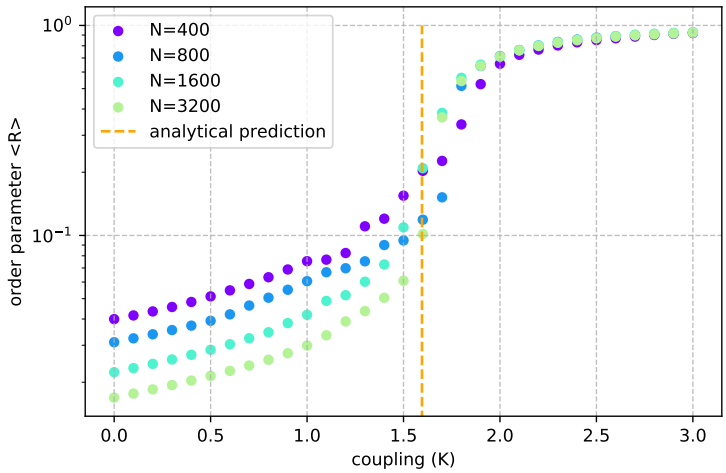


Figure 2.5 : R en fonction de du couplage K pour différentes valeurs de N. On utilise les mêmes paramètres que précédemment pour calculer R. En pointillé, la valeur analytique de K_c.

Nous pouvons ensuite tracer R en fonction de du nombre d'oscillateurs N, avec une valeur de couplage loin du seuil critique ce qui est fait sur la figure 2.6. Nous pouvons clairement voir que R décroit vers zéro comme $R \propto \frac{1}{\sqrt{N}}$.

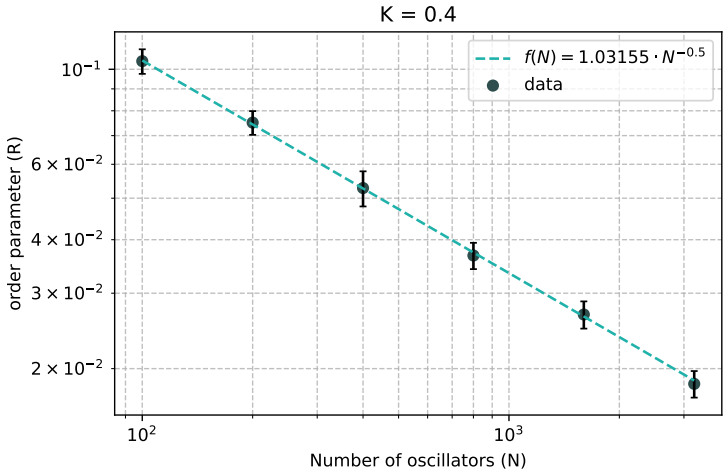


Figure 2.6 : Paramètre d'ordre R en fonction du nombre d'oscillateurs N. Pour une valeur de couplage $K = 0.4 < K_c$. On ajuste les données avec une fonction de la forme $a \cdot N^b$

3

Cas d'une distribution bimodale

Dans cette partie nous allons changer de distribution de fréquence et voir comment cette simple modification enrichit la diversité des phénomènes que l'on observe. Après avoir définis notre nouvelle distribution 3.1 on fera plusieurs simulations avec différents paramètres 3.2, 3.3 que l'on compilera dans un diagramme de phase de notre système 3.4. Enfin nous montrerons l'existence d'un régime de bi-stabilité dans la partie 3.5.

3.1. Distribution bimodale

Jusqu'à présent nous avons étudié le cas d'une distribution unimodale et symétrique. On peut maintenant se demander si ces résultats sont encore valides en changeant la distribution g. La suite logique est de considérer une distribution bimodale. Le cas d'une lorentzienne bimodale a été largement étudié dans la littérature (voir [3]) mais l'analyse s'avère plus complexe que dans le cas unimodal. Nous explorerons le cas d'une distribution g égale à la somme de deux gaussiennes identiques.

$$g(\omega) = \frac{1}{2\sigma\sqrt{2\pi}}(e^{-(\omega-\omega_0)^2/2\sigma^2} + e^{-(\omega+\omega_0)^2/2\sigma^2}) \tag{3.1}$$

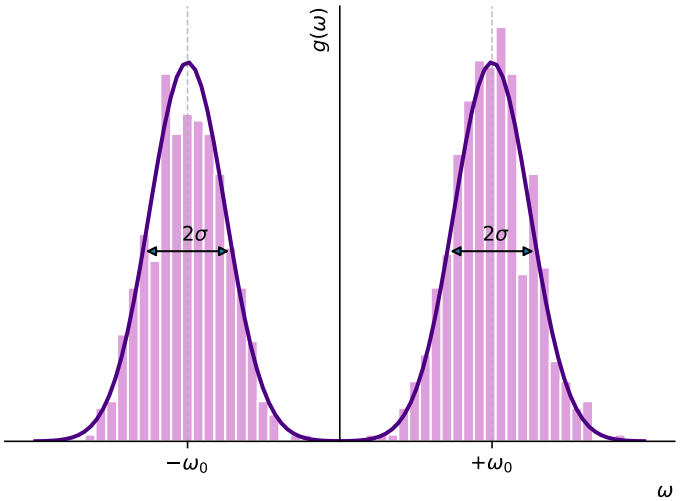


Figure 3.1 : Distribution gaussienne bimodale, formée à partir de deux gaussiennes de moyennes $\pm\omega_0$ et d'écart-type σ. En histogramme les valeurs obtenues avec la simulation B, comparée à la distribution théorique en trait plein.

3.2. Transition incohérence vers synchronisation

Pour ce premier exemple on utilise une distribution g avec ω_0 et $\sigma = 1$. Les deux gaussiennes sont proches, vu leur écart-type, et ce cas est semblable à une gaussienne unimodale. En représentant R moyen pour différentes valeurs de couplage K on retrouve la transition entre le régime incohérent et la synchronisation, observée en figure 2.3.

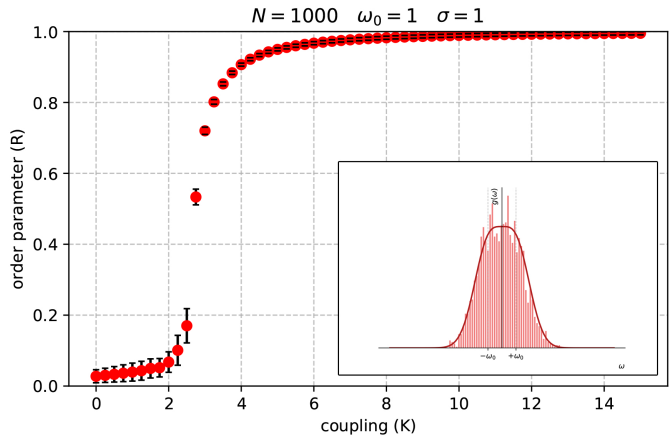


Figure 3.2 : Paramètre d'ordre R en fonction de K pour un distribution bimodale de paramètre $\omega = 1$ et $\sigma = 1$, représentée dans l'encadré. $\delta K = 0.25$, Temps de simulation $T = 100$, $deltat = 10^{-3}$ et $t_{transient} = 50$.

Dans le cas où les deux gaussiennes sont proches le système commence à se synchroniser aux alentours de $K = 2.25$ (voir figure 3.2).

3.3. Ondes stationnaires

Si on augmente ω_0 pour éloigner les deux pics et avoir une distribution véritablement bimodale on remarque l'apparition d'un nouveau régime. Entre l'état incohérent et l'état complètement synchronisé apparait un régime "d'onde stationnaires". En traçant R en fonction de K (voir figure 3.3 à gauche) on voit un plateau autour de $R = 0.6$ où l'écart-type de R fluctue beaucoup (voir les bars d'erreur en figure 3.3 à gauche). Pour caractériser ces fluctuations dans le temps de R on peut tracer l'écart-type en fonction du couplage K (à droite dans la figure 3.3). On remarque que lors de la première transition l'écart-type augmente de façon continue, puis diminue avant de s'effondrer lors de la seconde transition.

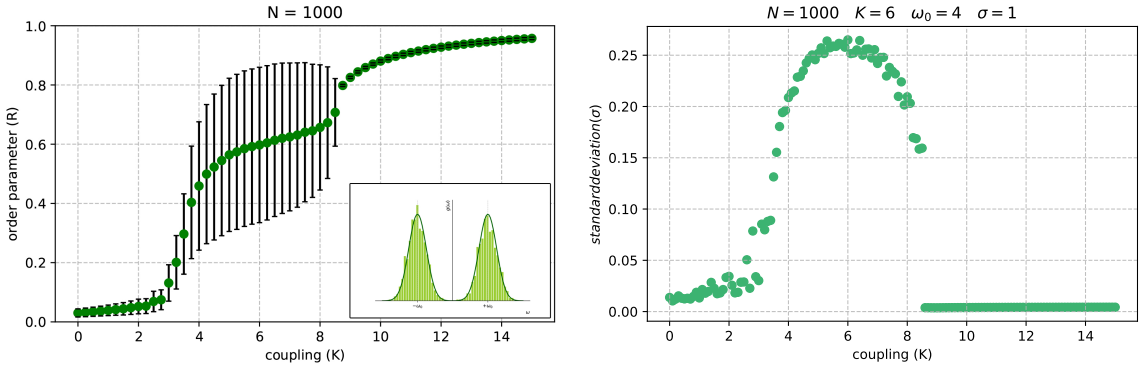


Figure 3.3 : À gauche : le paramètre d'ordre R en fonction du couplage K, la distribution de paramètres $\omega_0 = 4$ et $\sigma = 1$ est représentée dans l'encadré. À droite l'écart type en fonction du couplage

On observe donc ici les transitions entre trois régimes :

$$incohérence \rightarrow ondes\ stationnaires \rightarrow synchronisation$$

La première transition est continue, c'est une transition du second ordre, et la deuxième est discontinue,

donc du premier ordre. Le simple fait d'éloigner suffisamment les deux gaussiennes nous a permis de voir ce nouveau régime.

Pour visualiser les odes stationnaires du régime intermédiaire, on peut tracer le paramètre d'ordre R en fonction du temps pour une valeur de couplage K dans ledit régime. On obtiens la figure 3.4 où l'on voit bien R osciller comme des ondes stationnaires. Ces oscillations varient de la dynamique stationnaire des autres régimes (voir figure 2.2) elles induisent une valeur finie de l'écart-type (à droite sur 3.3) dans le régime des ondes stationnaires.

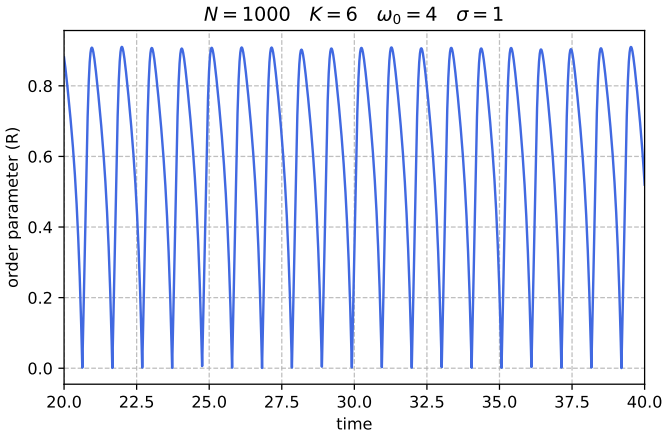


Figure 3.4 : Paramètre d'ordre en fonction du temps, dans le régime des ondes stationnaires avec $K = 6$. Agrandissement entre $T = 20$ et $T = 40$, on a toujours $\delta t = 10^{-3}$.

Ce régime correspond à la situation où les deux populations d'oscillateurs dont les fréquences sont centrées autour $+\omega_0$ et $-\omega_0$ tournent autour du cecle 1.3 à la même vitesse mais dans des sens opposés. On a donc R qui varie entre 0 quand les deux groupes sont l'un en face de l'autre, et qui tend vers 1 quand ils se croisent.

3.4. Diagramme de phase

On peut faire les simulations précédentes en faisant varier les paramètres de la distribution et en compilant les résultats on trouve le diagramme de phase du système (figure 3.5) où l'on peut visualiser R et l'écart-type associé pour différents paramètres de la distribution et du couplage.

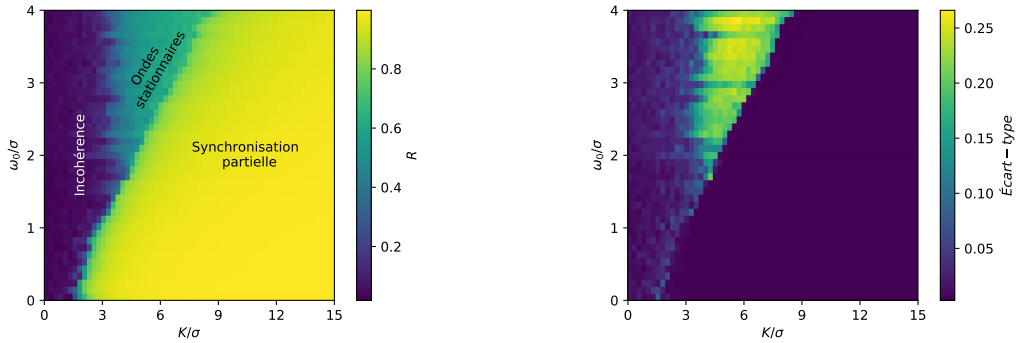


Figure 3.5 : Diagramme de phase. Resultats des simulations montrant le comportement à long terme du système. Pour chaque valeur de ω_0 on fait varier le couplage K. $\delta\omega_0 = 0.1$ et $\delta K = 0.25$. À gauche on calcule R moyenné sur un intervale de temps où le système est stable, à droite on représente l'écart-type associé.

Comme attendu, quand le couplage est inférieur à la valeur critique on reste dans le régime incohérent peu importe la valeur de ω_0, R est proche de 0 et l'écart-type aussi. On peut voir que si on éloigne les deux gaussiennes (i.e ω_0 augmente) la valeur de couplage K requise pour l'apparition de la synchronisation partielle augmente aussi. Dans ce régime R tend vers 1 mais l'écart-type tend vers

0. Entre ces deux régimes et pour une distribution "suffisamment bimodale" (à partir de $\omega_0 \approx 1.5$) apparaît le régime des ondes stationnaires qui semble s'élargir quand ω_0 augmente. C'est dans cette zone que l'écart-type sur R est le plus élevé (± 0.25), le paramètre d'ordre R a lui une valeur intermédiaire, autour de 0.5.

3.5. Simulation adiabatique et courbe d'hystérésis

Pour la distribution lorentzienne l'article [3] montre qu'autour du point triple (c'est à dire au point de transition entre les trois régimes, dans le cas de la figure 3.5 $K/\sigma = 1$ et $\omega_0/\sigma = 1.7$) existe deux régime où coexistent incohérence et synchronisation partielle pour l'un et ondes stationnaires et synchronisation partielle pour l'autre. Pour vérifier l'existence de cette bistabilité dans notre modèle nous avons fait des simulations adiabatiques autour du point triple de la figure 3.5 en réduisant le pas δK.

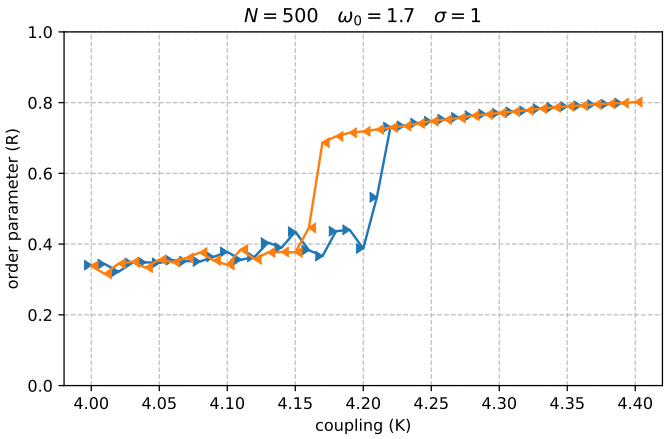


Figure 3.6 : Simulation adiabatique du paramètre d'ordre R en fonction du couplage K. On calcule la valeur moyenne de R en laissant φ (1.6) évoluer pour des valeurs $K = 4.0 \to 4.40 \to 4.0, \delta K = 0.01$. $T = 20$, $\delta = 10^{-4}$, $t_{transient} = 5$.

Pour réaliser une simulation adiabatique on laisse la phase de chaque oscillateur évoluer, on change la valeur de K tel que $K = K_{n+1} = K_n + \delta K$ puis quand $K = K_n$ on refait ces calculs mais en "sens inverse" c'est-à-dire que $K = K_{n-1} = K_n - \delta K$ Nous obtenons la courbe d'hystérésis 3.6, en effet on remarque R ne suit pas le même chemin quand on augmente ou quand on diminue K. À l'aller (quand K augmente, en bleu sur 3.6) la synchronisation partielle apparaît à partir de $K = 4.20$ mais au retour (K diminue, en orange sur 3.6) le système reste partiellement synchronisé jusqu'à $K = 4.17$. On est face à une région de bistabilité où les deux régimes peuvent apparaître, ondes stationnaires ou synchronisation partielle. Si on fait plusieurs simulations sans changer aucun paramètres on tombe aléatoirement sur l'un ou l'autre des état (voir Figure 3.7).

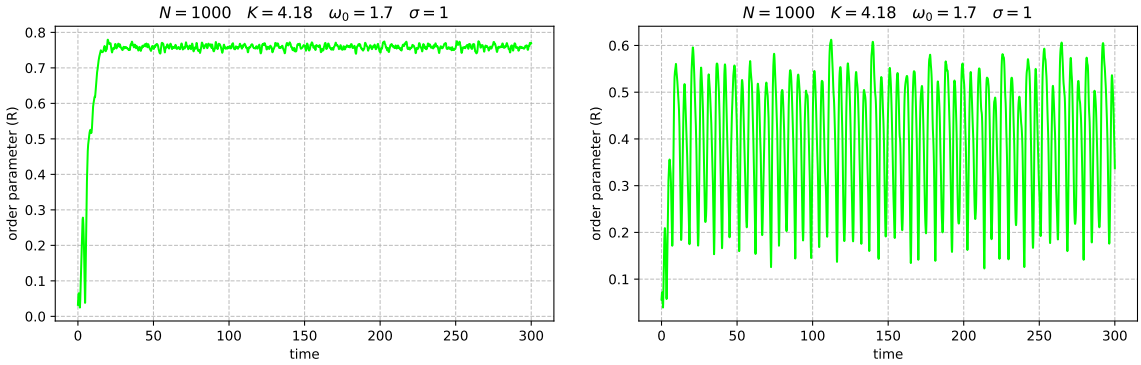


Figure 3.7 : Paramètre d'ordre R en fonction du temps. On a utilisé la valeur de couplage au centre de la courbe d'hystérésis sur la figure 3.6, soit $K = 4.18$. À gauche le système est partiellement synchronisé, à droite on est dans le régime des ondes stationnaires. On a utilisé $T = 300, \delta t = 10^{-4}, t_{transient} = 50$ dans le code fournis en annexe B.

Les ondes stationnaires ne sont pas parfaitement régulières mais varient globalement entre 0.1 et 0.6. On constate en tous cas, que simplement en redistribuant les phases initiales des oscillateurs on obtient une nouvelle dynamique du système. Dans cette région de bistabillité c'est donc les conditions initiales qui vont faire basculer le système dans un état ou l'autre.

4

Conclusion

4.1. Bilan des résultats

Après l'étude du cas simple d'une distribution gaussienne où nous avons simulé l'évolution de R dans le temps, retrouvé la valeur de couplage critique et caractérisé l'influence du nombre d'oscillateurs sur l'évolution du système. Nous avons vu que dans le cas d'une distribution bimodale on observait trois régimes distincts et nous avons trouver une région de bistabilité autour du point triple du système. Une façon simple d'améliorer ces résultats serait d'affiner les résultats notamment d'améliorer la résolution du diagramme de phase, en utilisant des pas d'évolution plus petits et un plus grand nombre d'oscillateurs.

4.2. Futurs travaux

4.2.1. Couplage

Dans la nature les systèmes où tous les oscillateurs sont reliés ne sont pas très fréquents, on pourrait donc couper des liens entre oscillateurs pour simuler un système plus réaliste. Ceci devient particulièrement vrai quand $N \to \infty$.

4.2.2. Distribution bimodale asymétrique

Durant ce stage et dans toutes les simulations nous avons toujours gardé une distribution de fréquence symétrique. Pour de futurs travaux on pourrait donc étudier des distributions non symétrique. On pourrait s'attendre à observer des ondes progressives.

Références

[1] Yoshiki Kuramoto. *Chemical Oscillations, Waves, and Turbulence*. Springer-Verlag, 1984.

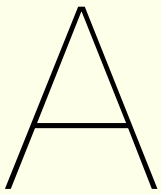
[2] Yoshiki Kuramoto. « Self-entrainment of a population of coupled non-linear oscillators ». In : *International Symposium on Mathematical Problems in Theoretical Physics*. Sous la dir. d'Huzihiro Araki. Berlin, Heidelberg : Springer Berlin Heidelberg, 1975, p. 420-422. isbn : 978-3-540-37509-8.

[3] E. A. Martens et al. « Exact results for the Kuramoto model with a bimodal frequency distribution ». In : *Phys. Rev. E* 79 (2 fév. 2009), p. 026204. doi : 10.1103/PhysRevE.79.026204. url : https://link.aps.org/doi/10.1103/PhysRevE.79.026204.

[4] Steven Strogatz. *Sync : The Emerging Science of Spontaneous Order*. Hyperion, 2003.

[5] Steven H. Strogatz. « From Kuramoto to Crawford : exploring the onset of synchro- nization in populations of coupled oscillators. » In : *Physica D : Nonlinear Phenomena* (2000). url : https://static1.squarespace.com/static/5436e695e4b07f1e91b30155/t/544525a8e4b0b8e2e8230fa3/1413817768995/from-kuramoto-to-crawford.pdf.

[6] Arthur T. Winfree. « Biological rhythms and the behavior of populations of coupled oscillators ». In : *Journal of Theoretical Biology* 16.1 (1967), p. 15-42. issn : 0022-5193. doi : https://doi.org/10.1016/0022-5193(67)90051-3. url : https://www.sciencedirect.com/science/article/pii/0022519367900513.



C++ Code

```cpp
/*
Auteur : Ferdinand Tixidre
M1 Physique et applications
Université de Cergy

Etude du modèle de Kuramoto : ce programme permet de calculer
le paramètre d'ordre R en fonction du temps, en fonction de K
et pour différents nombres d'oscillateurs N.
*/
#include <iostream>
#include <sstream>
#include <fstream>
#include <vector>
#include <cstdio>
#include <random>
#include <chrono>
#include <string>
#include <cmath>
#include <ctime>
#define PI 3.14159265359
using namespace std;

// Retourne la somme des cos
double RealPart(vector<double> array, int N){
 double sum = 0.0;
 double  A  = 0.0;
 for (int i = 0 ; i < N ; i++){
  sum += cos( array[i] );
 }
 A = 1.0 / N * sum;
 return A;
}

// Retourne la somme des sin
double ImaginaryPart(vector<double> array, int N){
 double sum = 0.0;
 double  B  = 0.0;
 for (int i = 0 ; i < N ; i++){
  sum += sin( array[i] );
 }
```

```
 B = 1.0 / N * sum;
 return B;
}

double StandardDeviation(vector <double> data, double mean){
 double standardDeviation = 0.0;
 for(int i = 0; i < data.size(); ++i){
        standardDeviation += pow(data[i] - mean, 2);
 }
return sqrt(standardDeviation / data.size());
}

// Pour numéroter les fichiers
string intToString(int t){
  std::string ch;
  ostringstream outs;
  outs << t;
  ch = outs.str();
  return ch;
}

// CONSTANTES
const int N  = 800;      // Nombre d'oscillateurs max
double   W_0 = 0.0;      // Fréquence moyenne
double sigma = 1;        // Ecart type
double   K   = 2;        // Coupling max
double   dk  = 0.01;     // Coupling step
double   L   = K/dk;     // Nombres de différents k
double   T   = 300.;     // temps d'évolution
double   dt  = 0.001;    // pas temporelle
double   M   = T/dt;     // nombre d'itérations
double t_transient = 50;// temps de stabilisation
double k = 0.0;          // Coupling min

int main(){
 clock_t begin = clock();
 std::string Rmean_vs_K = "N" + intToString(N) + "_Rmean_vs_K.txt";
 ofstream fichier2 ( Rmean_vs_K.c_str() );
 cout << " n = " << N << endl;

   // Seed du RNG
   unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
   std::default_random_engine generator (seed);

   std::normal_distribution<double> distribution (W_0, sigma);
    std::uniform_real_distribution<double> unif(0, 1);
    std::vector<double> W;
    std::vector<double> Phi;
    W.reserve(N);
    Phi.reserve(N);

    for (int i = 0 ; i < N ; i++){
        double currentRandomNumber = unif(generator);
        Phi.emplace_back( 2*PI*currentRandomNumber );
        W.emplace( distribution(generator) );
    }
```

```
97
98      // BOUCLE DES DIFFERENTES VALEURS DE K
99      for (int j = 0 ; j <= L ; j ++){
100      // un nouveau fichier pour chaque valeur de N et de K
101      std::string R_vs_t = "N" + intToString(N) + "_R_vs_t_k" + intToString(j) +
          ↪  ".dat";
102      ofstream fichier ( R_vs_t.c_str() );
103      double R_mean = 0.0;       // Initialisation de la somme des R
104      std::vector<double> R_vec;
105      std::vector<double> Phi_clone(N);
106      for (int i = 0 ; i < N ; i++){Phi_clone[i] = Phi[i];}
107
108      // BOUCLE TEMPORELLE
109      for(int l = 0 ; l < M ; l++){
110       int pourcent = (100 * (l + 1)) / M;   // Progress bar
111
112       double A = RealPart(Phi_clone, N);
113       double B = ImaginaryPart(Phi_clone, N);
114       double R = sqrt( A*A + B*B );    // Paramètre d'ordre
115       //double theta = atan( B / A );
116
117         if(l*dt>t_transient){   // Si on est plus dans le régime transitoire
118          R_mean += R;
119          R_vec.push_back(R);
120         }
121       fichier << l*dt << ' ' << R << endl;   // Ecriture de R_vs_t
122
123       // CALCUL DES PHASES DES N OSCILLATEURS
124       for( int i = 0 ; i < N ; i++ ) {
125        //Phi_clone[i] = Phi_clone[i] + (W[i] + k * R * sin(abs(theta) -
           ↪  Phi_clone[i])) * dt ;  // Méthode d'Euler
126        Phi_clone[i] = Phi_clone[i] + ( W[i] + k * ( B*cos(Phi_clone[i])  -
           ↪  A*sin(Phi_clone[i]) ) ) * dt ; //Méthode améliorée
127       }
128
129      std::cout << "\r" << "[" << std::string(pourcent / 5, (char)35) <<
          ↪  std::string(100 / 5 - pourcent / 5, ' ') << "]";
130      std::cout.flush();
131      }
132      fichier.close(); // Ferme R_vs_t
133      R_mean = R_mean / ( M - t_transient / dt );
134      double sd = StandardDeviation(R_vec, R_mean);
135      fichier2 << k << ' ' << R_mean << ' ' << sd << endl;
136      k += dk;            // Prochaine valeur de couplage
137      std::cout << j/L * 100 << "%" << endl;     // % of computation done
138   }
139   fichier2.close();           // Ferme R_vs_K
140   clock_t end = clock();
141   double elapsed_secs = double(end - begin) / CLOCKS_PER_SEC;
142   std:: cout << '\a';
143   std::cout << "Terminé en " << " " << elapsed_secs << " secondes" << endl;
144   return 0;
145  }
```

B

Distribution bimodale

```cpp
/*
Auteur : Ferdinand Tixidre
M1 Physique et applications
Université de Cergy

Deuxième partie du stage, étude du modèle de Kuramoto avec une distribution de fréquence bimodale.
Simulation adiabatique.
*/

#include <iostream>
#include <sstream>
#include <fstream>
#include <vector>
#include <cstdio>
#include <random>
#include <chrono>
#include <string>
#include <cmath>
#include <ctime>
#include <array>
using namespace std;
#define PI 3.14159265359

double RealPart(vector<double> array, int N)   // Retourne la somme des cos
{
 double sum = 0.0;
 double  A  = 0.0;
 for (int i = 0 ; i < N ; i++){
  sum += cos( array[i] );
 }
 A = 1.0 / N * sum;
 return A;
}

double ImaginaryPart(vector<double> array, int N) // Retourne la somme des sin
{
 double sum = 0.0;
 double  B  = 0.0;
 for (int i = 0 ; i < N ; i++){
```

```
40    sum += sin( array[i] );
41   }
42   B = 1.0 / N * sum;
43   return B;
44  }
45
46  double StandardDeviation(vector <double> data, double mean)
47  {
48   double standardDeviation = 0.0;
49   for(unsigned long i = 0; i <= data.size(); i++){
50      standardDeviation += pow(data[i] - mean, 2);
51    }
52    return sqrt(standardDeviation / data.size());
53  }
54
55  string intToString(int t)    // Pour numéroter les fichiers
56  {
57    std::string ch;
58    ostringstream outs;
59    outs << t;
60    ch = outs.str();
61    return ch;
62  }
63
64  // CONSTANTES
65  const int N  = 1000;        // Nombre d'oscillateurs
66  //double   W_0 = 2.0;       // Fréquence moyenne
67  double sigma = 1.0;       // Ecart type
68  double   K   = 4.40;       // Coupling max
69  double   Kmin = 4.0;          //coupling min
70  double   dk  = 0.01;          // Coupling step
71  double   L   = (K-Kmin)/dk;  // Nombres de différents k
72  double   T   = 300.;      // Temps d'évolution
73  double   dt  = 0.0001;     // Pas temporelle
74  double   M   = T/dt;      // Nombre d'itérations
75  double t_transient = 50.;     // Temps de transition
76  int FileNb = 0;
77
78  int main(){
79  clock_t begin = clock();
80
81  std::vector<double> K_values; // SIMULATION ADIABATIQUE
82  //K_values.reserve(2*L + 1);
83  K_values.emplace_back(4.18);
84
85
86
87  //for(int i = 0 ; i <= L ; i++){ K_values.emplace_back( Kmin + dk * i ); };
88  //for(int i = 1 ; i <  L ; i++){ K_values.emplace_back( K - dk * i ); };
89  double length = K_values.size();
90
91  cout << " n = " << N << endl;
92
93  for( double W_0 = 1.7 ; W_0 <= 1.7 ; W_0 += 0.05){
94    cout << "W_0 = " << W_0 <<endl;
95
```

```
96      FileNb = W_0 * 10;
97      std::string Rmean_vs_K = "W_0" + intToString(FileNb) + "_N" + intToString(N) +
        ↪  "_Rmean_vs_K.txt";
98      ofstream fichier1 ( Rmean_vs_K.c_str() );    //fichier1 = R_vs_K
99
100     // Seed du RNG
101     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
        ↪  //Time based seed
102     std::default_random_engine generator (seed);
103
104     using normal_dist   = std::normal_distribution<>;
105     using discrete_dist = std::discrete_distribution<std::size_t>;
106     std::uniform_real_distribution<double> unif(0, 1);  // Distribution uniforme
        ↪  entre 0 et 1
107     std::vector<double> W;
108     std::vector<double> Phi;
109     W.reserve(N);
110     Phi.reserve(N);
111
112     // Génère les fréquences selon deux Gaussiennes
113     auto G = std::array<normal_dist, 2>{
114       normal_dist{ W_0, sigma}, // mean, stddev of G[0]
115       normal_dist{-W_0, sigma}, // mean, stddev of G[1]
116     };
117     auto weight = discrete_dist{
118       0.5, // weight of G[0]
119       0.5, // weight of G[1]
120     };
121     // Cree une distribution bimodale à partir des deux Gaussiennes et crée la
        ↪  distribution de phase
122     for(int n = 0; n <= N; n++){
123       auto index  = weight(generator);
124       double currentRandomNumber = unif(generator);
125       W.emplace_back( G[index](generator) );
126       Phi.emplace_back( 2*PI*currentRandomNumber );
127     };
128
129     // BOUCLE DES DIFFERENTES VALEURS DE K
        ↪  ////////////////////////////////////////////
130     for (unsigned j = 0 ; j < K_values.size() ; j ++){
131
132      std::string R_vs_t = "W_0" + intToString(W_0) + "_N" + intToString(N) +
        ↪  "_R_vs_t_k" + intToString(j) + ".dat";// un nouveau fichier pour chaque
        ↪  valeur de N et de K
133      ofstream fichier2 ( R_vs_t.c_str() );   //fichier2 = R_vs_t
134
135      //On copie la distribution Phi pour toujours utiliser la même
136      //std::vector<double> Phi_clone(N);
137      //for (int i = 0 ; i < N ; i++){Phi_clone[i] = Phi[i];}
138
139      std::vector<double> R_vec;
140      R_vec.reserve( (T - t_transient)*dt );
141     double R_mean = 0.0;      // Initialisation de la somme des R
142
143      // BOUCLE TEMPORELLE
         ↪  ////////////////////////////////////////////////////////
```

```
144       for(int l = 0 ; l < M ; l++){
145        int pourcent = (100 * (l + 1)) / M;  // Progress bar
146        double A = RealPart(Phi, N);
147        double B = ImaginaryPart(Phi, N);
148        double R = sqrt( A*A + B*B );        // Paramètre d'ordre
149        //double theta = atan( B / A );
150
151         if(l*dt>t_transient){     // Si on est plus dans le régime transitoire
152          R_mean += R;
153          R_vec.emplace_back(R);
154         }
155
156        fichier2 << l*dt << ' ' << R << endl;  // Ecriture de R_vs_t
157
158        // CALCUL DES PHASES DES N OSCILLATEURS ////////////////////////////////
159        for( int i = 0 ; i < N ; i++ ){
160         //Phi_clone[i] = Phi_clone[i] + (W[i] + k * R * sin(abs(theta) -
          ↪  Phi_clone[i])) * dt ;  // Méthode d'Euler
161           Phi[i] = Phi[i] + ( W[i] + K_values[j] * ( B*cos(Phi[i]) - A*sin(Phi[i])
            ↪  ) ) * dt ;
162        }
163
164      std::cout << "\r" << "[" << std::string(pourcent / 5, (char)35) <<
        ↪  std::string(100 / 5 - pourcent / 5, ' ') << "]";
165      std::cout.flush(); //Progress bar
166       }
167       fichier2.close(); // Ferme R_vs_t
168
169       R_mean = R_mean / ( M - t_transient / dt );
170       double sd = StandardDeviation(R_vec, R_mean);
171       fichier1 << K_values[j] << ' ' << R_mean << ' ' << sd << endl;
172       std::cout << j/length * 100 << "%" << endl;  // % of computation done
173    }
174    fichier1.close();        // Ferme R_vs_K
175   }
176   clock_t end = clock();
177   double elapsed_secs = double(end - begin) / CLOCKS_PER_SEC;
178   std::cout << "Terminé en " << " " << elapsed_secs << " secondes" << endl;
179   std:: cout << '\a';
180   return 0;
181   }
```