4.1. Introduction Générale

Ce **Chapitre 4** s'inscrit dans la continuité des précédents, après avoir exploré, dans le Chapitre 3, les différentes manières de **représenter** les entités (sub-symbolique, symbolique, hybride) et de **calculer** la synergie $S(i,j)$, nous allons maintenant voir **comment** ces entités et synergies s'**animent** au sein d'un **réseau** (le Synergistic Connection Network, SCN). Au cœur de ce processus se trouve la **dynamique d'auto-organisation** : un mécanisme par lequel les pondérations $\omega_{i,j}$ entre entités évoluent (augmentent ou diminuent), sous l'effet de la synergie $S(i,j)$ et d'un terme de régulation $(\tau, \eta, ...)$. C'est cette **dynamique** qui fait émerger, à partir de données initiales et d'un calcul de $S(i,j)$, des **clusters** cohérents ou des **patterns** inattendus — bref, la structure finale du réseau.

4.1.1. Rappel du Contexte et des Objectifs

Le **Chapitre 2** a posé les bases théoriques du **Deep Synergy Learning** en définissant les mécanismes fondamentaux par lesquels des entités interagissent via une **fonction de synergie**, notée $S(i,j)$. Ces premiers éléments ont mis en évidence la possibilité d'associer à chaque paire (i,j) un score susceptible d'indiquer le degré de similarité, de compatibilité ou de complémentarité entre les entités $\mathcal{E}_i$ et $\mathcal{E}_j$. Le **Chapitre 3** s'est ensuite concentré sur la **représentation** de ces entités, il a explicité les cadres sub-symbolique et symbolique, proposé divers modes de calcul d'**embeddings** et évoqué la façon d'agréger des règles ou des concepts dans la synergie S.

À l'issue de ces deux chapitres, le **DSL** dispose ainsi d'une **mesure** $S(i,j)$ attribuant à chaque couple $(\mathcal{E}_i, \mathcal{E}_j)$ une **valeur** susceptible de capter leur affinité ou leur pertinence mutuelle. Cependant, l'essence même du Deep Synergy Learning ne réside pas dans la seule évaluation statique de ces synergies, la véritable originalité vient du **processus d'auto-organisation**, c'est-à-dire la façon dont un réseau dynamique, appelé **Synergistic Connection Network (SCN)**, fait évoluer les **pondérations** $\omega_{i,j}$ en réponse à la synergie $S(i,j)$.

Cette évolution se formalise par une **règle de mise à jour**, par exemple

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right],$$

qui instaure un **équilibre** entre la **croissance** des liaisons en cas de synergie élevée et la **décroissance** imposée par un terme $\tau\,\omega_{i,j}(t)$. Le but d'un tel mécanisme n'est pas simplement de calculer un score, mais de laisser le réseau s'**organiser** pour faire émerger des **clusters** ou des **motifs** particuliers, reflétant la structure latente des données ou des règles. Certains liens se renforcent alors jusqu'à atteindre une valeur stable, tandis que d'autres s'étiolent et peuvent même s'annuler si la synergie ne justifie pas leur maintien.

Cette **auto-organisation** ne mène pas toujours à un unique état final ; elle peut aboutir à une **configuration stable**, à des **oscillations** persistantes ou à plusieurs **attracteurs** coexistant selon l'initialisation et les paramètres. L'objectif de ce **Chapitre 4** est de comprendre en détail comment l'on passe de la définition de la règle de mise à jour à l'analyse de comportements variés, émergence de clusters, stabilité ou instabilité, compétition latérale, et contrôle par **recuit simulé** ou par **inhibition** plus prononcée.

Ce chapitre approfondit donc plusieurs aspects. Il décrit dans un premier temps les **formes** possibles de mise à jour (additive, multiplicative ou hybrides), montrant comment chacune traduit un certain parti pris sur la plasticité des liens. Il discute ensuite des **régimes** de convergence et des méthodes pour les caractériser. Il met en perspective le rôle central de la **synergie** S , lorsque $S(i,j)$ demeure constant, il est aisé d'analyser la convergence locale vers $\omega_{i,j}^{*} = S(i,j)/\tau$. Quand au contraire $S(i,j)$ varie parce que la représentation des entités se réajuste (contexte d'**apprentissage continu**), la dynamique peut échapper à la stationnarité et exiger d'autres outils d'étude.

Cette continuité théorique et algorithmique entre la **définition** de S (Chap. 3) et la **mise à jour** de ω (Chap. 4) incarne le cœur conceptuel du **DSL**. Les prochaines sections illustrent pas à pas comment, à partir d'un simple "score de similarité" entre entités, un **réseau** adaptatif finit par révéler la structure profonde d'un ensemble de données, ou se réorganiser dans le temps pour s'accorder aux changements de flux et de représentations.

4.1.2. Place et Cotenu du Chapitre

Le présent chapitre s'attache à détailler la **dynamique d'auto-organisation** mise en œuvre dans un **DSL** (Deep Synergy Learning) une fois que la **synergie** S(i,j) est définie pour chaque paire d'entités. Après avoir posé les bases théoriques (Chapitre 2) et présenté (Chapitre 3) la manière dont on construit les **représentations** et dont on calcule la fonction S, il s'agit maintenant d'étudier la **loi d'évolution** des pondérations $\omega_{i,j}(t)$. Cette loi est cruciale dans la mesure où elle dicte comment, à partir de synergies statiques ou variables, le **Synergistic Connection Network (SCN)** va former, détruire ou maintenir certains liens, jusqu'à aboutir à un réseau non supervisé structuré par la **coopération** ou la **compétition** des entités.

Dans la première partie (section 4.2), l'accent est mis sur les **règles de mise à jour** fondamentales. On examine notamment la forme **additive**, considérée comme la plus classique, où l'on ajoute ou retranche un incrément proportionnel à S(i,j) et à la différence $\omega_{i,j}(t)$. On étudie également les variantes **multiplicatives**, dans lesquelles la croissance ou la décroissance dépend de $\omega_{i,j}(t)$ de manière proportionnelle, et les mécanismes d'**inhibition** qui introduisent une interaction compétitive entre liaisons. À chaque étape, il sera montré comment la croissance $\Delta\omega_{i,j}$ reflète la corrélation mesurée par S(i,j) et la **décroissance** imposée par un terme de pénalisation $\tau\,\omega_{i,j}(t)$. On verra que la structure émergente résulte d'un **équilibre** entre ces forces de renforcement et de régulation.

Dans un second temps (sections 4.3 et 4.4), l'intérêt se porte sur la façon dont cette dynamique favorise la **formation spontanée** de **clusters**. Des entités disposant d'une synergie mutuelle élevée se trouvent amenées à consolider leurs liaisons $\omega_{i,j}$, tandis que les connexions de moindre synergie tendent à s'affaiblir ou à disparaître. On abordera la notion d'**attracteurs multiples**, qui s'expriment par le fait que différents points fixes (ou cycles) peuvent coexister, menant à des organisations distinctes selon l'initialisation. L'analyse des **oscillations** et des régimes de **pseudo-chaos** mettra en lumière la possibilité de comportements temporels plus complexes, dont il convient de maîtriser les effets si l'on souhaite obtenir un réseau stable. Les **méthodes de contrôle** (par **recuit simulé**, **inhibition latérale** ou **saturation**) seront ainsi présentées comme des outils pour résorber ou canaliser ces fluctuations, de sorte à garantir ou à forcer une organisation plus robuste.

Dans une dernière partie (sections 4.5 à 4.7), des **exemples concrets** seront décrits pour illustrer la **mise en œuvre** de ces règles de mise à jour dans des scénarios variés. Des simulations à taille réduite montreront la constitution progressive d'un ou plusieurs **clusters**, validant expérimentalement la théorie énoncée. Par la suite, des cas plus étendus, tels qu'une population d'agents en robotique ou un jeu de données multimodales, mettront en évidence le comportement du **DSL** face à un univers plus riche, où l'on observe l'**auto-organisation** continuer d'agir même lorsque la fonction S évolue au fil du temps.

En somme, ce chapitre entend dévoiler la "**vie interne**" du **SCN** en examinant pas à pas ce qui se produit lorsqu'on laisse $\omega_{i,j}(t)$ s'ajuster en fonction de $\mathrm{S}(i,j)$. Le propos est de comprendre comment naissent des **clusters**, des **attracteurs stables** et comment gérer les **oscillations** ou **conflits** susceptibles d'apparaître. Cet approfondissement des phénomènes d'**auto-organisation** servira de fondement pour aborder, dans les chapitres ultérieurs, l'intégration plus large du **DSL** dans des architectures complexes et dans des applications en **temps continu**.

4.2.1.1. Équation Principale

La **mise à jour** de la **pondération** $\omega_{i,j}(t)$ peut se formaliser par l'équation suivante, considérée comme une forme canonique dans la dynamique du **Deep Synergy Learning** :

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\big[\mathrm{S}(i,j) \;-\; \tau\,\omega_{i,j}(t)\big].$$

Dans cette expression, la variable $\omega_{i,j}(t)$ représente la **force de lien** à l'itération t, tandis que la fonction $\mathrm{S}(i,j)$ incarne la **synergie** instantanée entre les entités $\mathcal{E}_i$ et $\mathcal{E}_j$. Le terme η fait office de **taux d'apprentissage**, déterminant la vitesse à laquelle la pondération se modifie, et $\tau\,\omega_{i,j}(t)$ assure une **décroissance** proportionnelle à la valeur courante de $\omega_{i,j}(t)$. L'itération s'effectue de manière **additive**, c'est-à-dire qu'à chaque pas, on ajoute un incrément linéaire dépendant de la différence $\big[\mathrm{S}(i,j) - \tau\,\omega_{i,j}(t)\big]$.

La présence du facteur τ contribue à une forme de **relaxation** : si la **synergie** $\mathrm{S}(i,j)$ ne parvient pas à compenser la décroissance due au terme $\tau\,\omega_{i,j}(t)$, la pondération $\omega_{i,j}(t)$ tend à diminuer au fil des itérations. À l'inverse, lorsque $\mathrm{S}(i,j)$ est suffisamment élevée, le terme $\eta\,\mathrm{S}(i,j)$ amène $\omega_{i,j}(t)$ à croître progressivement, favorisant un **lien** de plus en plus fort entre i et j.

Du point de vue analytique, il est souvent utile d'examiner le **point d'équilibre** où la variation $\omega_{i,j}(t+1) - \omega_{i,j}(t)$ s'annule. Dans ce cas, on obtient l'égalité

$$\omega^*_{i,j} \;=\; \omega^*_{i,j} \;+\; \eta\big[\mathrm{S}(i,j) \;-\; \tau\,\omega^*_{i,j}\big],$$

ce qui implique

$$\mathrm{S}(i,j) \;-\; \tau\,\omega^*_{i,j} \;=\; 0 \quad\Rightarrow\quad \omega^*_{i,j} \;=\; \frac{\mathrm{S}(i,j)}{\tau}.$$

Cette solution $\omega^*_{i,j}$ illustre la **zone de stabilisation** autour de laquelle la pondération tend à se fixer quand le schéma de mise à jour est laissé à lui-même. Ainsi, en présence d'une **forte synergie**, la valeur $\omega_{i,j}$ s'élève et se maintient à un niveau $\mathrm{S}(i,j)/\tau$, tandis que dans le cas d'une synergie plus faible, la décroissance prend le dessus. Du point de vue **physique**, l'équation ci-dessus évoque un

gradient ou une forme de mouvement orienté vers un attracteur local, traduisant la logique de renforcement ou d'atténuation des liens selon l'intensité de la coopération sous-jacente.

4.2.1.2. Sens Interprétatif : Lien avec la "Descente d'Énergie" et Convergence vers $\boldsymbol{\omega^* = \frac{S}{\tau}}$

Le **Deep Synergy Learning** peut être éclairé par une analogie physique et un principe de **descente d'énergie**. Dans le chapitre 2.4.3, il a été proposé de conceptualiser le réseau comme un système qui tend à minimiser une **pseudo-énergie** ou "fonction potentielle", notée $\mathcal{J}(\mathbf{\Omega})$. Une formulation simplifiée de cette énergie prend généralement la forme

$$\mathcal{J}(\mathbf{\Omega}) \;=\; -\sum_{(i,j)} \omega_{i,j}\,\mathbf{S}(i,j) \;+\; \frac{\tau}{2}\sum_{(i,j)} (\omega_{i,j})^2 \;+\; \ldots$$

Dans cette expression, le terme $-\,\omega_{i,j}\,\mathbf{S}(i,j)$ favorise l'augmentation de la **pondération** $\omega_{i,j}$ lorsque la **synergie** $\mathbf{S}(i,j)$ est élevée, tandis que la partie $\tau/2\;(\omega_{i,j})^2$ impose une **pénalisation** qui s'accroît avec la magnitude du lien. La **règle additive** décrite précédemment,

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\left[\mathbf{S}(i,j) - \tau\,\omega_{i,j}(t)\right],$$

peut alors s'interpréter comme un schéma de **descente** (ou de "quasi-descente") dans le gradient négatif de $\mathcal{J}$. La composante $\mathbf{S}(i,j)$ joue un rôle de "force" qui pousse $\omega_{i,j}$ à croître, tandis que le terme $\tau\,\omega_{i,j}$ sert de "frein" ou de "relaxation" vers des valeurs plus modestes.

Lorsque la **synergie** $\mathbf{S}(i,j)$ reste **constante** au cours du temps, il est possible d'étudier la **convergence** de la dynamique en cherchant le point fixe $\omega^*_{i,j}$ pour lequel $\omega_{i,j}(t+1) = \omega_{i,j}(t) = \omega^*_{i,j}$. En imposant cette condition stationnaire, on obtient

$$\omega^*_{i,j} \;=\; \omega^*_{i,j} \;+\; \eta\left[\mathbf{S}(i,j) - \tau\,\omega^*_{i,j}\right] \quad\Rightarrow\quad \mathbf{S}(i,j) \;=\; \tau\,\omega^*_{i,j} \quad\Rightarrow\quad \omega^*_{i,j} \;=\; \frac{\mathbf{S}(i,j)}{\tau}.$$

Ce résultat indique que la **valeur d'équilibre** $\omega^*_{i,j}$ est directement proportionnelle à la **synergie** $\mathbf{S}(i,j)$ et inversement proportionnelle au paramètre τ, lequel représente la "raideur" ou l'intensité du freinage imposé à la croissance du lien. Aussi longtemps que η (taux d'apprentissage) est choisi de façon à éviter les oscillations excessives, on peut montrer que $\omega_{i,j}(t)$ se rapproche de $\omega^*_{i,j}$, ce qui confère à cet équilibre le statut de **point attracteur** local.

Sur le plan **énergétique**, cette situation correspond à un **minimum** de la pseudo-énergie $\mathcal{J}$ dans un scénario simple où la **synergie** ne dépend pas elle-même de $\omega_{i,j}$. Au sein du **SCN**, chaque lien se "met en conformité" avec la valeur $\mathbf{S}(i,j)/\tau$, de sorte que les liaisons **fortes** correspondent précisément aux paires (i,j) ayant une **synergie** élevée, et les liaisons **faibles** sont observées là où $\mathbf{S}(i,j)$ demeure basse. Cette mise à niveau progressive peut être considérée comme une "relaxation" du réseau, dans laquelle les forces de renforcement (la **synergie**) et de décroissance (le terme $\tau\,\omega_{i,j}$) parviennent à un **compromis**.

Dans le cas plus général où $\mathbf{S}(i,j)$ varie au fil des itérations (parce que la représentation des entités se modifie ou que des règles symboliques sont adaptées en continu), l'équilibre local $\omega^*_{i,j}$ se déplace

simultanément. La **règle additive** agit alors comme un processus d'**auto-organisation** qui s'efforce de suivre ces déplacements, sans nécessairement parvenir à une véritable convergence statique. Cette faculté d'adaptation souligne la souplesse du **DSL** dans des environnements évolutifs, tout en conservant l'idée d'une "descente d'énergie" vers des configurations où la **synergie** et la **décroissance** se neutralisent mutuellement.

4.2.1.3. Cas Stationnaire vs. Cas Dynamique : $S(i,j)$ constant ou recalculé à chaque itération

La **règle additive** introduite précédemment repose sur une mise à jour des pondérations $\omega_{i,j}$ proportionnelle à la **synergie** $S(i,j)$ et à un terme de **relaxation** $\tau\,\omega_{i,j}(t)$. Cette formulation se décline en deux scénarios : un **cas stationnaire**, où $S(i,j)$ demeure fixe dans le temps, et un **cas dynamique**, où $S(i,j)$ est recalculé d'une itération à l'autre, par exemple lorsque les représentations ou les règles sous-jacentes se modifient.

Dans le **cas stationnaire**, on suppose que la **synergie** $S(i,j)$ ne subit aucune variation au fil du temps. Les entités $\mathcal{E}_i$ et $\mathcal{E}_j$ conservent alors leurs caractéristiques ou leurs représentations sans être affectées par un nouvel apprentissage ou par l'introduction de changements contextuels. Le système $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right]$ se comporte dès lors comme une **descente d'énergie** dirigée vers un point fixe, vérifiant $\omega_{i,j}^{*} = S(i,j)/\tau$. Lorsque la valeur $\omega_{i,j}(t)$ atteint ce rapport, il n'existe plus d'incitation à croître ni à décroître, dans la mesure où la composante $S(i,j)$ équilibre parfaitement la pénalisation $\tau\,\omega_{i,j}(t)$. D'un point de vue analytique, ce régime simplifie grandement l'étude de la **convergence** : le système tend à se stabiliser, et l'on peut ainsi prouver, sous certaines conditions (en particulier sur le pas d'apprentissage η), une convergence locale vers un attracteur. Un **DSL** exploité dans ce cadre stationnaire permet de clarifier l'effet de la synergie sur la formation de liens forts, tout en évitant les complications liées à la variation de $S(i,j)$.

Dans un **cas dynamique**, la **synergie** $S(i,j)$ subit un recalcul, potentiellement à chaque itération, de sorte qu'elle devient dépendante des modifications intervenues dans le réseau. Il se peut que les représentations vectorielles des entités $\mathbf{r}(i)$ et $\mathbf{r}(j)$ soient elles-mêmes ajustées par un processus d'apprentissage sous-jacent, ou que des blocs de règles symboliques évoluent (ajout/suppression d'axiomes, reparamétrage logique). Dans ces circonstances, $S(i,j)$ prend la forme d'une fonction $S_t(i,j)$ explicitement liée au temps t. La mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) \;+\; \eta\big[S_t(i,j) \;-\; \tau\,\omega_{i,j}(t)\big]$$

opère alors sur un **paysage** en mouvement. La valeur de $\omega_{i,j}(t)$ doit sans cesse s'adapter, car l'équilibre instantané $S_t(i,j)/\tau$ se déplace à mesure que $S_t(i,j)$ évolue. Un **DSL** de ce type décrit plus fidèlement un système soumis à des variations externes ou internes (arrivée de nouvelles entités, apprentissage continu, ajustement de règles symboliques). Il n'est pas toujours garanti que l'on aboutisse à un point fixe au sens strict ; la dynamique peut osciller, changer de régime ou poursuivre un déplacement lent suivant la trajectoire imposée par $S_t(i,j)$.

En conclusion, la **formulation additive** se prête tout autant au **cas stationnaire** qu'à un **cas dynamique** plus réaliste. Lorsque $S(i,j)$ reste constant, l'analyse du point fixe $\omega_{i,j}^{*} = S(i,j)/\tau$ permet de mettre en évidence un comportement stable et interprétable. Si, au contraire, la **synergie** varie, la boucle de mise à jour se conçoit comme un **système adaptatif** en mouvement constant,

dont la convergence peut se révéler délicate à établir. Les sections suivantes approfondissent ces notions en examinant la stabilisation, la formation de **clusters**, et les phénomènes oscillatoires dans des configurations plus complètes du **Deep Synergy Learning**.

4.2. Règles de Mise à Jour Fondamentales

Après l'introduction générale (4.1) qui présente la dynamique d'auto-organisation comme le moteur principal du **DSL** (Deep Synergy Learning), nous abordons maintenant les **règles** mathématiques qui permettent de **faire évoluer** les pondérations $\omega_{i,j}$ dans le **SCN** (Synergistic Connection Network). Ces règles traduisent la manière dont un lien $\omega_{i,j}$ se **renforce** ou se **détériore** en réponse à la **synergie** $S(i,j)$, et elles constituent le **noyau** de l'auto-organisation.

4.2.1. Formulation Additive Classique

La première formulation, très fréquente dans la littérature et déjà mentionnée en chapitres précédents (2.3, 2.4.3), est dite **additive** : on met à jour $\omega_{i,j}(t)$ en lui **ajoutant** un terme proportionnel à la différence $\left[S(i,j) - \tau\,\omega_{i,j}(t)\right]$.

4.2.1.1. Équation Principale

La **mise à jour** de la **pondération** $\omega_{i,j}(t)$ peut se formaliser par l'équation suivante, considérée comme une forme canonique dans la dynamique du **Deep Synergy Learning** :

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\left[\mathrm{S}(i,j) \;-\; \tau\,\omega_{i,j}(t)\right].$$

Dans cette expression, la variable $\omega_{i,j}(t)$ représente la **force de lien** à l'itération t, tandis que la fonction $\mathrm{S}(i,j)$ incarne la **synergie** instantanée entre les entités $\mathcal{E}_i$ et $\mathcal{E}_j$. Le terme η fait office de **taux d'apprentissage**, déterminant la vitesse à laquelle la pondération se modifie, et $\tau\,\omega_{i,j}(t)$ assure une **décroissance** proportionnelle à la valeur courante de $\omega_{i,j}(t)$. L'itération s'effectue de manière **additive**, c'est-à-dire qu'à chaque pas, on ajoute un incrément linéaire dépendant de la différence $\left[\mathrm{S}(i,j) - \tau\,\omega_{i,j}(t)\right]$.

La présence du facteur τ contribue à une forme de **relaxation** : si la **synergie** $\mathrm{S}(i,j)$ ne parvient pas à compenser la décroissance due au terme $\tau\,\omega_{i,j}(t)$, la pondération $\omega_{i,j}(t)$ tend à diminuer au fil des itérations. À l'inverse, lorsque $\mathrm{S}(i,j)$ est suffisamment élevée, le terme $\eta\,\mathrm{S}(i,j)$ amène $\omega_{i,j}(t)$ à croître progressivement, favorisant un **lien** de plus en plus fort entre i et j.

Du point de vue analytique, il est souvent utile d'examiner le **point d'équilibre** où la variation $\omega_{i,j}(t+1) - \omega_{i,j}(t)$ s'annule. Dans ce cas, on obtient l'égalité

$$\omega^*_{i,j} \;=\; \omega^*_{i,j} \;+\; \eta\left[\mathrm{S}(i,j) \;-\; \tau\,\omega^*_{i,j}\right],$$

ce qui implique

$$\mathrm{S}(i,j) \;-\; \tau\,\omega^*_{i,j} \;=\; 0 \quad\Rightarrow\quad \omega^*_{i,j} \;=\; \frac{\mathrm{S}(i,j)}{\tau}.$$

Cette solution $\omega^*_{i,j}$ illustre la **zone de stabilisation** autour de laquelle la pondération tend à se fixer quand le schéma de mise à jour est laissé à lui-même. Ainsi, en présence d'une **forte synergie**, la

valeur $\omega_{i,j}$ s'élève et se maintient à un niveau $\mathrm{S}(i,j)/\tau$, tandis que dans le cas d'une synergie plus faible, la décroissance prend le dessus. Du point de vue **physique**, l'équation ci-dessus évoque un **gradient** ou une forme de mouvement orienté vers un attracteur local, traduisant la logique de renforcement ou d'atténuation des liens selon l'intensité de la coopération sous-jacente.

4.2.1.2. Sens Interprétatif : Lien avec la "Descente d'Énergie" et Convergence vers $\boldsymbol{\omega^* = \mathrm{S}/\tau}$

Il est souvent utile d'analyser la **règle additive** comme une forme de **descente de gradient** dans un **paysage d'énergie**. Au chapitre 2.4.3, il a été proposé de considérer une **fonction d'énergie** $\mathcal{J}(\boldsymbol{\Omega})$ dont l'un des termes dominants peut s'écrire sous la forme $-\sum_{i,j}\ \omega_{i,j}\,\mathrm{S}(i,j)$. Dans un cadre simplifié, on peut poser

$$\mathcal{J}(\boldsymbol{\Omega})\ =\ -\sum_{i,j}\ \omega_{i,j}\,\mathrm{S}(i,j)\ +\ \frac{\tau}{2}\sum_{i,j}\ (\omega_{i,j})^2\ +\ \ldots$$

et vérifier que la mise à jour

$$\omega_{i,j}(t+1)\ =\ \omega_{i,j}(t)\ +\ \eta\left[\mathrm{S}(i,j)-\tau\,\omega_{i,j}(t)\right]$$

correspond, dans ses grandes lignes, à la **descente** du gradient de $\mathcal{J}$ pour peu que $\mathrm{S}(i,j)$ ne dépende pas de $\omega_{i,j}$.

La présence du **terme** $\tau\,\omega_{i,j}(t)$ peut se lire comme un "coût" qui **régularise** la **pondération** $\omega_{i,j}$. Plus la valeur de $\omega_{i,j}(t)$ est élevée, plus le produit $\tau\,\omega_{i,j}(t)$ vient la ramener à la baisse, sauf si la **synergie** $\mathrm{S}(i,j)$ est suffisamment grande pour compenser ce mouvement. Dans un scenario stationnaire, où $\mathrm{S}(i,j)$ est constante au cours du temps, la convergence locale de $\omega_{i,j}(t)$ vers un point fixe peut être étudiée en imposant $\omega_{i,j}(t+1)=\omega_{i,j}(t)=\omega^*$. Il en résulte l'équation

$$\omega^*\ =\ \omega^*\ +\ \eta\,[\mathrm{S}(i,j)-\tau\,\omega^*],$$

impliquant $\mathrm{S}(i,j)=\tau\,\omega^*$. Autrement dit, on obtient

$$\omega^*\ =\ \frac{\mathrm{S}(i,j)}{\tau}.$$

Cette **valeur** ω^* peut être considérée comme la **solution d'équilibre** : si la **règle additive** est appliquée de façon itérative et que $\mathrm{S}(i,j)$ reste fixe, la pondération $\omega_{i,j}(t)$ évolue jusqu'à se stabiliser approximativement autour de $\mathrm{S}(i,j)/\tau$. La signification profonde de ce résultat est qu'il y a un **équilibre** entre le **gain** (lié à $\mathrm{S}(i,j)$) et la **pénalisation** quadratique (associée à $\tau\,\omega_{i,j}^2$), comme si le système « descendait » la fonction d'énergie $\mathcal{J}$ et s'arrêtait dans une vallée correspondant à un minimum local.

Dans un contexte plus général, où la **synergie** $\mathrm{S}(i,j)$ varie au fil du temps, il n'existe pas forcément de convergence définitive vers une valeur fixe ω^*. Il est toutefois possible de constater que la **pondération** $\omega_{i,j}$ "suit" la synergie à travers une suite de régimes quasi-stationnaires, ou qu'elle oscille si les changements de $\mathrm{S}(i,j)$ sont trop rapides ou si l'on choisit un **taux d'apprentissage** η trop grand. Le facteur τ, agissant comme un **terme de régularisation**, garantit néanmoins que

$\omega_{i,j}(t)$ ne croîtra pas indéfiniment en l'absence de feedback négatif ; il limite donc la possibilité d'un emballement des poids et maintient le réseau dans un régime stable ou faiblement oscillant.

4.2.1.3. Cas Stationnaire vs. Cas Dynamique : $S(i,j)$ constant ou recalculé à chaque itération

La **règle additive** introduite précédemment repose sur une mise à jour des pondérations $\omega_{i,j}$ proportionnelle à la **synergie** $S(i,j)$ et à un terme de **relaxation** $\tau\,\omega_{i,j}(t)$. Cette formulation se décline en deux scénarios : un **cas stationnaire**, où $S(i,j)$ demeure fixe dans le temps, et un **cas dynamique**, où $S(i,j)$ est recalculé d'une itération à l'autre, par exemple lorsque les représentations ou les règles sous-jacentes se modifient.

Dans le **cas stationnaire**, on suppose que la **synergie** $S(i,j)$ ne subit aucune variation au fil du temps. Les entités $\mathcal{E}_i$ et $\mathcal{E}_j$ conservent alors leurs caractéristiques ou leurs représentations sans être affectées par un nouvel apprentissage ou par l'introduction de changements contextuels. Le système $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right]$ se comporte dès lors comme une **descente d'énergie** dirigée vers un point fixe, vérifiant $\omega^*_{i,j} = S(i,j)/\tau$. Lorsque la valeur $\omega_{i,j}(t)$ atteint ce rapport, il n'existe plus d'incitation à croître ni à décroître, dans la mesure où la composante $S(i,j)$ équilibre parfaitement la pénalisation $\tau\,\omega_{i,j}(t)$. D'un point de vue analytique, ce régime simplifie grandement l'étude de la **convergence** : le système tend à se stabiliser, et l'on peut ainsi prouver, sous certaines conditions (en particulier sur le pas d'apprentissage η), une convergence locale vers un attracteur. Un **DSL** exploité dans ce cadre stationnaire permet de clarifier l'effet de la synergie sur la formation de liens forts, tout en évitant les complications liées à la variation de $S(i,j)$.

Dans un **cas dynamique**, la **synergie** $S(i,j)$ subit un recalcul, potentiellement à chaque itération, de sorte qu'elle devient dépendante des modifications intervenues dans le réseau. Il se peut que les représentations vectorielles des entités $\mathbf{r}(i)$ et $\mathbf{r}(j)$ soient elles-mêmes ajustées par un processus d'apprentissage sous-jacent, ou que des blocs de règles symboliques évoluent (ajout/suppression d'axiomes, reparamétrage logique). Dans ces circonstances, $S(i,j)$ prend la forme d'une fonction $S_t(i,j)$ explicitement liée au temps t. La mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) \;+\; \eta\big[S_t(i,j) \;-\; \tau\,\omega_{i,j}(t)\big]$$

opère alors sur un **paysage** en mouvement. La valeur de $\omega_{i,j}(t)$ doit sans cesse s'adapter, car l'équilibre instantané $S_t(i,j)/\tau$ se déplace à mesure que $S_t(i,j)$ évolue. Un **DSL** de ce type décrit plus fidèlement un système soumis à des variations externes ou internes (arrivée de nouvelles entités, apprentissage continu, ajustement de règles symboliques). Il n'est pas toujours garanti que l'on aboutisse à un point fixe au sens strict ; la dynamique peut osciller, changer de régime ou poursuivre un déplacement lent suivant la trajectoire imposée par $S_t(i,j)$.

En conclusion, la **formulation additive** se prête tout autant au **cas stationnaire** qu'à un **cas dynamique** plus réaliste. Lorsque $S(i,j)$ reste constant, l'analyse du point fixe $\omega^*_{i,j} = S(i,j)/\tau$ permet de mettre en évidence un comportement stable et interprétable. Si, au contraire, la **synergie** varie, la boucle de mise à jour se conçoit comme un **système adaptatif** en mouvement constant, dont la convergence peut se révéler délicate à établir. Les sections suivantes approfondissent ces notions en examinant la stabilisation, la formation de **clusters**, et les phénomènes oscillatoires dans des configurations plus complètes du **Deep Synergy Learning**.

4.2.1. Formulation Additive Classique

La première formulation, très fréquente dans la littérature et déjà mentionnée en chapitres précédents (2.3, 2.4.3), est dite **additive** : on met à jour $\omega_{i,j}(t)$ en lui **ajoutant** un terme proportionnel à la différence $\left[S(i,j) - \tau\,\omega_{i,j}(t)\right]$.

4.2.1.1. **Équation Principale**

Dans le cadre d'un **Deep Synergy Learning (DSL)**, l'évolution de la **pondération** $\omega_{i,j}$ entre deux entités $\mathcal{E}_i$ et $\mathcal{E}_j$ se fait de manière itérative et s'exprime par une équation de mise à jour additive qui repose sur la synergie mesurée entre ces entités et un terme de décroissance régulateur. La forme canonique de cette équation est donnée par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right],$$

où chaque symbole revêt une signification particulière. Dans cette équation, $\omega_{i,j}(t)$ représente la **pondération** (ou force de lien) entre les entités $\mathcal{E}_i$ et $\mathcal{E}_j$ à l'itération t. Le terme $S(i,j)$ désigne la **synergie instantanée** ou **statique** entre ces deux entités, telle que définie dans le chapitre 3, et qui est souvent calculée à partir de mesures de similarité (par exemple, la similarité cosinus ou une distance transformée par un noyau gaussien). Le paramètre η correspond au **taux d'apprentissage**, un facteur qui contrôle la vitesse avec laquelle la pondération est mise à jour, tandis que le produit $\tau\,\omega_{i,j}(t)$ représente un **terme de décroissance** ou de relaxation, modulé par le paramètre τ qui détermine l'intensité de cette décroissance. Ainsi, un τ élevé aura pour effet de réduire plus fortement $\omega_{i,j}(t)$ lorsque $S(i,j)$ n'est pas suffisamment grand pour compenser cette décroissance.

Mathématiquement, l'évolution de $\omega_{i,j}$ est dite **additive** dans la mesure où, à chaque itération, on ajoute (ou l'on soustrait) un delta proportionnel à la différence $S(i,j) - \tau\,\omega_{i,j}(t)$. Cette formulation rappelle un mécanisme de **gradient descent**, où la pondération se déplace progressivement vers un point d'équilibre. En effet, en fixant la mise à jour à zéro dans l'état stationnaire, on obtient l'équation

$$\omega_{i,j}(t+1) \approx \omega_{i,j}(t) \quad \Rightarrow \quad S(i,j) - \tau\,\omega_{i,j}(t) \approx 0,$$

ce qui implique que, à l'équilibre, la pondération satisfait approximativement

$$\omega_{i,j} \approx \frac{S(i,j)}{\tau}.$$

On peut ainsi interpréter cette équation comme une **force attractive** qui tend à renforcer le lien entre deux entités lorsque la synergie est forte, tout en assurant une décroissance progressive du lien lorsque la synergie est insuffisante. Si $\omega_{i,j}(t)$ est initialement faible et que $S(i,j)$ est grand – c'est-à-dire si la synergie mesurée est élevée – la mise à jour incrémentale va entraîner une augmentation itérative de $\omega_{i,j}$, conduisant le système vers une zone de stabilisation où la valeur de la pondération converge vers l'équilibre $\omega_{i,j} \approx \frac{S(i,j)}{\tau}$. À l'inverse, si $\omega_{i,j}(t)$ est élevé mais que

$S(i,j)$ demeure faible, le terme négatif $-\eta\,\tau\,\omega_{i,j}(t)$ prédomine, provoquant une décroissance progressive de la pondération.

Cette règle de mise à jour est souvent considérée comme la plus simple et la plus intuitive pour modéliser la dynamique des liens dans un DSL. Elle offre une représentation claire de la manière dont un lien évolue en fonction de la synergie instantanée et de la décroissance proportionnelle à son état courant, permettant ainsi de capturer à la fois l'aspect d'**apprentissage** local et l'effet de **relaxation** qui prévient une croissance incontrôlée des pondérations. En résumé, l'équation principale

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big]$$

se présente comme un modèle simple et efficace pour l'auto-organisation d'un réseau de connexions, où la pondération entre deux entités évolue de manière additive en réponse à une synergie mesurée, convergeant vers une valeur d'équilibre définie par la relation $\omega_{i,j} \approx \frac{S(i,j)}{\tau}$.

4.2.1.2. **Sens Interprétatif : lien avec la "descente d'énergie" (2.4.3), convergence vers** $\omega^* = \frac{S}{\tau}$

Dans le cadre du **Deep Synergy Learning (DSL)**, l'équation de mise à jour des pondérations est souvent interprétée sous l'angle d'une **descente d'énergie**. Cette interprétation repose sur l'idée qu'une fonction d'énergie, notée $\mathcal{J}(\mathbf{\Omega})$, définie sur l'ensemble des pondérations $\mathbf{\Omega}$ du réseau, diminue progressivement au cours des itérations d'apprentissage. Le chapitre 2.4.3 introduisait déjà le concept d'une fonction potentielle, par exemple sous la forme

$$\mathcal{J}(\mathbf{\Omega}) = -\sum_{i,j} \omega_{i,j}\; S(i,j) + \cdots,$$

où $S(i,j)$ représente la **synergie** entre les entités $\mathcal{E}_i$ et $\mathcal{E}_j$. La règle de mise à jour canonique du DSL s'exprime par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big],$$

où $\eta > 0$ est le **taux d'apprentissage** et $\tau > 0$ représente le **coefficient de décroissance** qui agit comme un terme de régularisation. On peut considérer cette équation comme une descente de gradient appliquée à la fonction d'énergie $\mathcal{J}$ dans un cas simplifié où $S(i,j)$ est indépendant de $\omega_{i,j}$. L'analogie avec la descente d'énergie se fait apparaître en remarquant que le terme $\big[S(i,j) - \tau\,\omega_{i,j}(t)\big]$ joue le rôle du gradient partiel de la fonction d'énergie par rapport à $\omega_{i,j}$, ce qui conduit, en cas de convergence, à un équilibre stationnaire.

Pour étudier cet équilibre, nous considérons la situation stationnaire où la mise à jour cesse, c'est-à-dire lorsque

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) = \omega^*.$$

En substituant cette condition dans l'équation de mise à jour, nous obtenons

$$\omega^* = \omega^* + \eta[S(i,j) - \tau\,\omega^*],$$

ce qui implique nécessairement que

$$S(i,j) - \tau\,\omega^* = 0.$$

Il en découle alors que la pondération stationnaire satisfait

$$\omega^* = \frac{S(i,j)}{\tau}.$$

Cette relation d'équilibre est particulièrement interprétable dans le sens où elle illustre comment la **force** du lien entre deux entités se stabilise à une valeur qui est directement proportionnelle à leur synergie $S(i,j)$ et inversement proportionnelle à la décroissance imposée par τ. On peut comprendre ce mécanisme en analogie avec un système physique où une particule se déplace dans un champ de potentiel et atteint un minimum d'énergie lorsque la force nette agissant sur elle devient nulle. Ici, le terme $\tau\,\omega_{i,j}(t)$ agit comme une force de **régularisation** ou un « coût » associé au maintien d'un lien fort, ce qui empêche les pondérations de croître indéfiniment en l'absence d'une synergie suffisante.

Dans un scénario idéal où $S(i,j)$ est constant dans le temps, la mise à jour additive converge de manière monotone vers $\omega^* = \frac{S(i,j)}{\tau}$. Toutefois, en pratique, $S(i,j)$ peut varier d'une itération à l'autre, ce qui rend la convergence plus complexe et peut empêcher l'atteinte d'un équilibre strict. Néanmoins, l'idée centrale demeure que la pondération $\omega_{i,j}$ tend à se "caler" sur une valeur qui reflète la force moyenne de la synergie entre les entités, assurant ainsi une **auto-organisation** harmonieuse du réseau.

Le terme de décroissance $\tau\,\omega_{i,j}(t)$ joue également un rôle essentiel en agissant comme un mécanisme d'**anti-explosion**. Sans ce terme, si $S(i,j)$ était constamment élevé, la pondération pourrait croître de façon indéfinie, ce qui serait incompatible avec la stabilité du système. La présence de ce terme garantit que, même en cas de synergie élevée, la pondération est ramenée à des valeurs réalistes, modélisant ainsi un compromis entre la **coopération** entre entités et le **coût** de maintenir des liens trop forts.

En résumé, la règle additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big],$$

peut être interprétée comme une descente de gradient sur une fonction d'énergie $\mathcal{J}(\mathbf{\Omega})$, où la convergence vers l'équilibre stationnaire $\omega^* = \frac{S(i,j)}{\tau}$ illustre la manière dont la synergie entre deux entités est équilibrée par la décroissance. Cette interprétation confère à la mise à jour un sens interprétatif profond, reliant le comportement numérique du DSL à des principes physiques tels que la minimisation de l'énergie, et offre ainsi une explication intuitive de la **stabilité** des liens dans le SCN.

4.2.1.3. **Cas Stationnaire vs. Cas Dynamique : $S(i,j)$ constant ou recalculé à chaque itération**

Dans l'analyse d'un **Deep Synergy Learning (DSL)**, l'évolution des **pondérations** $\omega_{i,j}$ repose sur la synergie $S(i,j)$ entre les entités, laquelle peut être considérée soit comme une valeur **constante**

au cours du temps, soit comme une grandeur **dynamique** recalculée à chaque itération. Ces deux cas présentent des comportements distincts qui influencent la convergence du système et la stabilité de l'**auto-organisation** dans le **Synergistic Connection Network (SCN)**.

Lorsqu'on adopte le **cas stationnaire**, on suppose que $S(i,j)$ demeure fixe. Dans ce contexte, les représentations des entités, qu'elles soient sub-symboliques ou symboliques, ne subissent pas de modifications significatives d'une itération à l'autre et aucune mise à jour contextuelle n'est introduite. La règle de mise à jour, qui s'exprime sous la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big],$$

se traduit alors par une dynamique qui converge vers un **point fixe** ou un attracteur stable. Pour démontrer ce comportement, il suffit de poser que, dans l'état stationnaire, $\omega_{i,j}(t+1) = \omega_{i,j}(t) = \omega^*$. En substituant dans l'équation, on obtient

$$\omega^* = \omega^* + \eta[S(i,j) - \tau\,\omega^*].$$

Cette égalité est satisfaite si et seulement si

$$S(i,j) - \tau\,\omega^* = 0,$$

ce qui conduit immédiatement à la relation

$$\omega^* = \frac{S(i,j)}{\tau}.$$

Cette convergence vers $\omega^* = \frac{S(i,j)}{\tau}$ permet d'interpréter le système en termes d'**équilibre énergétique** ou de descente d'énergie, où la pondération atteint une valeur d'équilibre déterminée par la synergie mesurée et par le coefficient de décroissance. Dans ce cas, l'ensemble du réseau se stabilise et les transitions entre états restent fixes, ce qui facilite l'analyse théorique de la convergence et permet de servir de référence pour comparer des systèmes dans des environnements plus dynamiques.

Dans le **cas dynamique**, en revanche, la synergie $S(i,j)$ n'est pas considérée comme constante, mais est recalculée à chaque itération. Cette situation se produit lorsque les représentations des entités évoluent au fil du temps en raison d'un apprentissage continu ou de modifications contextuelles, telles que des mises à jour des embeddings ou des ajustements dans les blocs symboliques. En effet, lorsque le DSL intègre un **apprentissage adaptatif**, le score $S(i,j)$ devient une fonction temporelle, que l'on peut noter $S(i,j,t)$. La règle de mise à jour s'écrit alors

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j,t) - \tau\,\omega_{i,j}(t)\big],$$

et il apparaît un couplage itératif dans lequel la pondération $\omega_{i,j}$ s'ajuste en même temps que $S(i,j)$ évolue. En pratique, cette interaction rend le système plus complexe, car les mises à jour des représentations sub-symboliques (par exemple, via le fine-tuning ou la réorganisation des données) entraînent des variations dans $S(i,j,t)$ qui peuvent perturber la convergence du réseau. Le système se comporte alors comme un **système adaptatif** où aucune convergence stationnaire absolue n'est garantie, et l'auto-organisation devient un processus continu dans lequel la structure du réseau est sans cesse réévaluée et reconfigurée. Un tel processus peut engendrer des oscillations ou des

réajustements brusques dans les pondérations, surtout si les variations de $S(i,j,t)$ sont significatives d'une itération à l'autre. Pour pallier ces effets, il est souvent nécessaire d'introduire des mécanismes de **lissage** ou de **verrouillage partiel**, afin que la transition entre différentes valeurs de $S(i,j,t)$ soit plus progressive et que l'ensemble du système reste stable malgré des mises à jour fréquentes.

En définitive, la **formulation additive** qui sous-tend la mise à jour des pondérations dans le DSL repose sur l'équilibre entre le **renforcement** induit par $S(i,j)$ et la **relaxation** imposée par $\tau\,\omega_{i,j}(t)$. Dans le cas stationnaire, où $S(i,j)$ est constant, la dynamique converge de manière prévisible vers $\omega^* = \frac{S(i,j)}{\tau}$, offrant une interprétation claire en termes de descente d'énergie. Dans le cas dynamique, le recalcul constant de $S(i,j,t)$ fait du processus une boucle itérative plus complexe, qui nécessite une analyse supplémentaire de la stabilité et de la formation des clusters dans le SCN. Cette distinction fondamentale entre un $S(i,j)$ stationnaire et un $S(i,j,t)$ dynamique est au cœur de la compréhension du comportement du DSL, et constitue une base de référence pour étudier l'impact de l'évolution des représentations sur la qualité de l'auto-organisation.

Ainsi, le cas stationnaire offre une situation de référence idéale pour l'analyse théorique de la convergence, tandis que le cas dynamique reflète la réalité d'un système en apprentissage continu, dans lequel la structure du réseau se reconfigure en permanence. Ces deux cas permettent de quantifier l'influence de la **variabilité** des représentations sur la stabilité des pondérations, et servent de point de départ pour développer des stratégies visant à optimiser la **robustesse** du DSL en environnement non stationnaire.

4.2.2. Variantes Multiplicatives, Inhibition, etc.

Bien que la **formulation additive** (4.2.1) soit souvent considérée la plus "classique" pour un **DSL** (Deep Synergy Learning), il existe d'autres **règles** de mise à jour des pondérations $\omega_{i,j}$. Certaines de ces variantes cherchent à introduire plus de **dynamisme** (croissance multiplicative), d'autres à intégrer de l'**inhibition** compétitive, ou encore à imposer des **mécanismes** de saturation pour prévenir une montée incontrôlée de certains liens.

4.2.2.1. Multiplicative : $\omega_{i,j}(t+1) = \omega_{i,j}(t)\,[\,1 + \eta\,(\ldots)]$

Dans cette variante, plutôt que d'ajouter un delta à $\omega_{i,j}(t)$ (comme dans la règle additive), on **multiplie** $\omega_{i,j}(t)$ par un **facteur** dépendant de la synergie $S(i,j)$ et de la décroissance τ. Autrement dit :

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t)\;\Big[1 \;+\; \eta\;\Big(\alpha\,S(i,j) \;-\; \beta\,\tau\,\omega_{i,j}(t)\Big)\Big],$$

ou une forme voisine (il existe plusieurs formulations possibles). L'idée générale est que la **mise à jour** se fait en **proportion** de $\omega_{i,j}(t)$, plutôt que par un **delta** absolu.

Mise à jour multiplicative : motivations, avantages et risques

Dans certains scénarios de Deep Synergy Learning (DSL), on abandonne la règle additive traditionnelle (sous la forme $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta$) au profit d'une **mise à jour multiplicative** permettant de modéliser une **croissance proportionnelle** ou quasi exponentielle. Concrètement, au lieu d'ajouter un terme Δ, on applique un **facteur** $[1 + \Delta]$, qui multiplie la valeur actuelle de $\omega_{i,j}(t)$. Ainsi, si la pondération $\omega_{i,j}(t)$ est déjà élevée, une légère variation de Δ peut se traduire par une forte augmentation, ce qui peut accélérer la structuration du réseau en mettant rapidement en évidence certains **liens dominants**.

Sur le plan analytique, on peut illustrer cette approche par la formule :

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t)\,\big[1 \;+\; \eta\, S(i,j) \;-\; \eta\,\tau\,\omega_{i,j}(t)\big].$$

Dans cette équation, η représente un taux d'apprentissage, τ un terme de régulation ou d'oubli, et $S(i,j)$ la synergie calculée entre les entités $\mathcal{E}_i$ et $\mathcal{E}_j$. Si $\omega_{i,j}(t)$ est déjà non négligeable, et que la partie $\eta\, S(i,j)$ dépasse $\eta\,\tau\,\omega_{i,j}(t)$, le produit dans la parenthèse demeure supérieur à 1, ce qui renforce massivement la pondération. C'est en cela que l'on parle de **dynamique non linéaire** : plus un lien est fort, plus il est susceptible d'augmenter vite, de sorte que l'on observe fréquemment un phénomène d'**auto-amplification**.

Cette caractéristique s'avère intéressante dès lors que l'on souhaite modéliser ou encourager une **coopération** exponentielle : par exemple, deux entités qui collaborent déjà fortement verront leur liaison s'intensifier encore, entraînant la création de **clusters** en "tout ou rien" (certains liens "explosent", tandis que d'autres chutent). Ce schéma **multiplicatif** reflète également divers comportements biologiques ou écosystémiques, où la vitesse de croissance dépend de la "valeur" présente (lois de populations, mécanismes de reproduction, etc.). En revanche, il comporte un **risque de divergence** : si $\omega_{i,j}(t)$ devient trop important et n'est pas freiné par le terme $\eta\,\tau\,\omega_{i,j}(t)$, la pondération peut croître sans limite, perturbant profondément la structure du réseau. Pour **éviter** ce problème, on introduit souvent des dispositifs de **saturation** (voir section 4.2.2.3) limitant la valeur maximale de $\omega_{i,j}$, ou des mécanismes d'**inhibition compétitive** (section 4.2.2.2) afin d'empêcher qu'un petit nombre de liens n'accapare l'essentiel du "poids" synergique.

En comparaison, la **règle additive** classique conduit plutôt à un point fixe $\omega^* = S(i,j)/\tau$ en l'absence d'autres rétroactions, et autorise un contrôle plus fin sur la progression des pondérations. La version multiplicative, elle, **accélère** la différenciation des liens, mais exige une **vigilance accrue** quant aux risques d'**explosion**, d'**instabilité** et de forte **sensibilité** aux variations de la synergie. En somme, choisir un schéma multiplicatif revient à opter pour un modèle dans lequel la mise en valeur rapide de certaines liaisons est un avantage, tout en s'assurant qu'un encadrement (clipping, inhibition) est bien en place pour maintenir la cohérence globale du réseau.

4.2.2.2. Inhibition Compétitive : ajout d'un terme $-\gamma \sum_{k \neq j} \omega_{i,k}\,(t)$ ou autre forme

Dans certains systèmes adaptatifs (biologiques, cognitifs, ou plus généralement en **self-organisation**), l'**inhibition compétitive** joue un rôle déterminant pour **réguler** la croissance simultanée de liens. L'idée est de **limiter** la progression de plusieurs connexions parallèles, incitant ainsi certaines à **dominer** au détriment d'autres. Cela évite un "tout ou rien" généralisé, ou la situation où trop de liens forts coexistent et saturent la dynamique. Concrètement, dans un **DSL**

(Deep Synergy Learning), on peut ajouter au terme de mise à jour un **facteur d'inhibition** qui pénalise $\omega_{i,j}$ lorsque, pour la même entité $\mathcal{E}_i$, d'autres pondérations $\omega_{i,k}$ (avec $k \neq j$) sont élevées.

A. Principe de l'inhibition compétitive

Une règle additive (ou multiplicative) peut être agrémentée d'un **terme** d'inhibition de la forme :

$$-\gamma \sum_{k \neq j} \omega_{i,k}\,(t),$$

où $\gamma > 0$ est un coefficient d'inhibition, et la somme $\sum_{k \neq j} \omega_{i,k}\,(t)$ représente la **"masse"** totale des liens que $\mathcal{E}_i$ entretient avec d'autres entités $k \neq j$.

En pratique, ce terme vient **s'ajouter** (avec un signe négatif) à la mise à jour de $\omega_{i,j}$. Plus $\mathcal{E}_i$ possède des liens forts avec d'autres $\mathcal{E}_k$, plus la **progression** de $\omega_{i,j}$ s'en trouve **limitée**.

L'**idée intuitive** derrière ce mécanisme repose sur une forme de **compétition** : si une entité $\mathcal{E}_i$ investit déjà fortement dans certains liens $\omega_{i,k}$, ses "ressources" disponibles pour renforcer de nouveaux liens $\omega_{i,j}$ sont de facto limitées. D'un point de vue **biologique**, ce principe évoque un phénomène de "**winner-take-all**" partiel ou d'"**inhibition latérale**", couramment décrit dans les modèles neuronaux, où la somme d'activité globale d'un neurone (c'est-à-dire la somme de ses liens) restreint sa capacité à s'activer simultanément vers d'autres connexions.

Exemple

Supposez une règle additive standard :

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right] \;-\; \gamma \sum_{k \neq j} \omega_{i,k}\,(t).$$

Quand $\sum_{k \neq j} \omega_{i,k}\,(t)$ est grand, cela **décourage** l'augmentation de $\omega_{i,j}$, forçant $\mathcal{E}_i$ à "faire un choix" parmi plusieurs liaisons concurrentes.

B. Avantages de l'inhibition compétitive

L'un des premiers bénéfices consiste à éviter toute "dispersion" : sans mécanisme d'inhibition, une entité $\mathcal{E}_i$ risquerait de créer de nombreux liens modérément forts avec un large éventail d'autres entités.

En introduisant une compétition, on **focalise** l'attention sur quelques liaisons dominantes, tout en réduisant l'intensité des autres ; il en résulte la formation de **clusters** plus nettement délimités.

Par ailleurs, l'inhibition compétitive aide à **contrôler la croissance simultanée** des connexions : même si la synergie $S(i,j)$ se révèle positive pour plusieurs entités, le réseau ne permet pas à toutes ces liaisons de croître en parallèle, introduisant une forme de **sélection** où seules les connexions les plus pertinentes se renforcent réellement.

Enfin, cette approche s'appuie sur des principes **biomimétiques**, rappelant l'**inhibition latérale** observée dans le cerveau, où un neurone en forte activité limite celle de ses voisins.

Dans un **DSL**, ce phénomène favorise la structuration de clusters plus précis et encourage la spécialisation, chaque entité s'orientant principalement vers les connexions qui maximisent la qualité de la coopération.

C. Limites et paramétrage

L'**inhibition compétitive** dans un réseau requiert une attention particulière quant à ses limites et à son paramétrage. D'abord, il existe un **risque de désactivation excessive** : si le coefficient γ qui contrôle la force de l'inhibition est trop élevé, il peut **empêcher** la croissance significative de tout nouveau lien, menant ainsi à un réseau **faiblement connecté**. Il est donc crucial de fixer γ à un niveau qui ne freine pas abusivement l'établissement de connexions utiles.

Par ailleurs, la **dépendance à la taille** du voisinage pose également problème. Dans la formule $\left[-\gamma \sum_{k \neq j} \omega_{i,k}\right]$, la **somme** peut devenir très importante lorsque l'entité $\mathcal{E}_i$ compte un grand nombre de voisins, annihilant de fait la mise à jour et rendant difficile l'émergence de nouvelles liaisons. Pour y remédier, on peut envisager de normaliser cette somme (par exemple, en la divisant par le nombre de voisins) ou d'adopter une approche "d'inhibition latérale stricte", qui retient uniquement la valeur maximale au lieu de la somme.

Enfin, il est fréquent de **combiner** l'inhibition compétitive avec un **mécanisme de saturation** : même si la synergie $S(i,j)$ incite un lien à s'intensifier, on l'empêche de dépasser un certain plafond. Cette mesure limite les phénomènes de "toute-puissance" où une entité tenterait d'activer tous ses liens simultanément, renforçant ainsi la sélectivité et la cohérence globale du réseau.

D. Conclusion

L'**inhibition compétitive** constitue un **levier** dans la mise à jour $\omega_{i,j}$ pour **forcer** un certain niveau de **compétition** entre les connexions sortant d'une même entité $\mathcal{E}_i$. Elle a pour effet de :

1. **Éviter** que de multiples liens forts cohabitent sans restriction,
2. **Promouvoir** des liens plus structurés ou plus "discriminants" (un "choix" se fait),
3. **Mieux** calquer certains phénomènes biologiques ou cognitifs (inhibition latérale), ou tout simplement améliorer la **lisibilité** des clusters en créant des sélections claires.

En pratique, elle se formule souvent comme un **terme négatif** proportionnel à la somme (ou au max) des liens $\omega_{i,k}(t)$ pour $k \neq j$. C'est un **complément** utile aux règles additive/multiplicative, afin de **rendre** la dynamique plus **sélective** et **moins** susceptible de disperser la force sur trop de connexions.

4.2.2.3. Saturation : clipping $\omega_{i,j}$ pour éviter la croissance infinie

Même dans un **DSL** (Deep Synergy Learning) recourant à une mise à jour plutôt "additive" (4.2.1) ou intégrant de l'inhibition (4.2.2.2), il peut arriver que les valeurs $\omega_{i,j}$ aient tendance à **s'emballer**. Par exemple, dans la variante **multiplicative** (4.2.2.1), un lien déjà élevé peut croître encore plus

vite, menant à un risque d'**explosion** numérique ou de "monopolisation" du réseau par quelques connexions hypertrophiées. Pour éviter cela, on introduit souvent un **mécanisme de saturation**, parfois appelé **clipping**, limitant la valeur maximale (ou minimale) que peut prendre $\omega_{i,j}$.

Le **principe du clipping** consiste à imposer, à l'issue de chaque itération de mise à jour, une **borne** maximale (et parfois minimale) à la pondération $\omega_{i,j}(t+1)$. Concrètement, cela se traduit par une opération de type :

$$\omega_{i,j}(t+1) \leftarrow \max\Big\{0,\ \min\big\{\omega_{i,j}(t+1),\ \omega_{\max}\big\}\Big\},$$

assurant ainsi que $\omega_{i,j}(t+1)$ ne dépasse pas la valeur $\omega_{\max}$. Selon les besoins, on peut également l'empêcher de devenir négative, notamment si l'on considère que les liens ne doivent prendre que des valeurs positives.

Dans la pratique, ce **clipping** s'opère **après** le calcul brut de la mise à jour (qu'elle soit additive, multiplicative ou incluant un mécanisme d'inhibition). On recadre la pondération finale uniquement si elle outrepasse l'intervalle autorisé : si, par exemple, $\omega_{i,j}(t+1)$ atteignait 50 alors que $\omega_{\max} = 10$, on la ramènerait à la valeur 10.

La **détermination** de $\omega_{\max}$ varie selon la nature du problème. Certaines configurations imposent, par exemple, qu'un lien ne dépasse jamais 1 pour des raisons de normalisation. Dans d'autres contextes, on fixe $\omega_{\max} = 10$ en considérant qu'au-delà, la distinction entre pondérations serait négligeable. Par ailleurs, on peut instaurer un seuil plancher (comme $\omega_{\min} = 0$) si l'on souhaite empêcher les liens de devenir négatifs ou si un sens physique (ou logique) exige que les pondérations restent non négatives.

Les **avantages de la saturation** dans un mécanisme de mise à jour des pondérations se manifestent à plusieurs niveaux.

En premier lieu, la présence d'un **plafond** évite l'**explosion** de certaines connexions : dans les règles multiplicatives (4.2.2.1), ou même en l'absence d'un terme de décroissance convenable, quelques liens pourraient croître indéfiniment si la partie positive (par exemple $\eta\, S(i,j)$) l'emportait toujours sur la partie négative. Grâce à un **clipping** imposant $\omega_{\max}$, on bride ces valeurs pour éviter qu'une minorité de liaisons n'atteigne des niveaux disproportionnés.

Ensuite, ce mécanisme contribue à **stabiliser la dynamique** globale. En empêchant $\omega_{i,j}$ de dépasser $\omega_{\max}$, on restreint la **disparité** entre les connexions, ce qui peut faciliter l'**analyse** du réseau ou accélérer la **convergence**, puisque l'espace de recherche (celui des pondérations) demeure borné.

Enfin, la saturation favorise un **contrôle de la lisibilité** des pondérations. Lorsqu'on désire interpréter $\omega_{i,j}$ comme un simple "score" dans un intervalle $[0,1]$, le clipping assure que ce score restera dans le **domaine** de lecture intuitif ; on n'observera ainsi pas de valeur hors échelle (telle qu'un score à 29,5), dépourvue de sens sémantique pour l'utilisateur ou l'analyste.

Les **inconvénients potentiels** de la saturation (ou points d'attention) doivent être soigneusement considérés au moment d'implémenter le clipping. En premier lieu, on peut évoquer l'**effet brutal** : dès que $\omega_{i,j}$ atteint $\omega_{\max}$, toute mise à jour positive devient sans effet. Cela risque de **biaiser** la dynamique si, par exemple, un lien devrait continuer à croître pour signaler sa domination. Le

réseau ne percevra plus la différence entre des liens saturés et ceux proches de la borne, ce qui peut diminuer le contraste souhaité.

Un autre point concerne la **discontinuité** dans la mise à jour. Le clipping crée une **coupure** non lisse : une fois la limite atteinte, la valeur reste figée. Sur le plan mathématique, ces discontinuités peuvent compliquer certaines analyses de convergence.

Le **choix** de $\omega_{\max}$ est également crucial. Une borne trop **basse** étouffe les liens qui pourraient être significativement plus élevés que d'autres, affectant la capacité du réseau à différencier des connexions "très fortes" de connexions "modérées". À l'inverse, si $\omega_{\max}$ est fixé à un niveau **très élevé**, on ne résout pas réellement le problème de l'explosion, et on n'apporte qu'un contrôle minimal.

Exemple d'Intégration dans la Mise à Jour

Le clipping peut s'insérer de différentes manières dans le processus de mise à jour :

1. Dans la **version additive** avec clipping, on calcule d'abord :

$$1.\,\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j)\tau\,\omega_{i,j}(t)\right],$$

puis on applique :

$$\omega_{i,j}(t+1) \;\leftarrow\; \min\{\max\{0,\,\omega_{i,j}(t+1)\},\omega_{\max}\}.$$

On garantit ainsi que $\omega_{i,j}(t+1)$ demeure dans l'intervalle $[0,\omega_{\max}]$.

1. Dans la **version multiplicative**, une fois le calcul effectué (voir 4.2.2.1), on impose simplement $\omega_{i,j}(t+1) \leq \omega_{\max}$.

2. Lorsqu'on **combine** cette approche avec une **inhibition compétitive**, l'inhibition agit en amont (pour réduire la mise à jour) et le **clamping** final vient s'assurer qu'un lien ne dépasse pas le plafond fixé. De cette manière, même si la synergie est très élevée, on limite la croissance excessive d'un lien unique, préservant l'équilibre général du réseau.

Conclusion

Le **clipping** ou la **saturation** des pondérations $\omega_{i,j}$ est une mesure **pratique** visant à **contrôler** :

- La croissance potentielle **infinie** (notamment dans des règles multiplicatives),
- Les **excès** de la dynamique (liaisons trop fortes qui éliminent toute nuance),
- L'**interprétation** (on se limite à un range $[0,\omega_{\max}]$ pour lire facilement le "poids" d'un lien).

Cette technique, assez courante dans les algorithmes neuronaux (on pense au "gradient clipping" dans le deep learning), s'avère tout aussi utile dans un **DSL** pour **stabiliser** ou **normaliser** la dynamique d'auto-organisation. Elle peut s'intégrer à n'importe quelle version de la mise à jour

(additive, multiplicative, inhibition) en garantissant qu'on n'outrepasse pas des bornes considérées comme logiquement ou numériquement souhaitables.

4.2.3. Cas Multi-Entités Hétérogènes

Les sections précédentes (4.2.1 et 4.2.2) ont présenté diverses **règles** de mise à jour (additives, multiplicatives, avec ou sans inhibition, saturation…) pour les pondérations $\omega_{i,j}$. Cependant, elles partaient souvent de l'hypothèse qu'il existait un seul **type** d'entité ou une forme unique de $S(i,j)$. Or, dans un **DSL** (Deep Synergy Learning) plus complet (chap. 3), on peut croiser :

- Des **entités sub-symboliques** (embeddings, vecteurs),
- Des **entités symboliques** (règles logiques, blocs ontologiques),
- Des **entités hybrides** (mix sub + sym),

et chacune d'elles peut interagir sous forme de différents **types de liens**. Cette diversité amène à considérer un **multi-réseau** ou des **sous-réseaux** partiellement interconnectés, avec des règles de mise à jour possiblement **différenciées** selon la nature des entités et des liaisons.

4.2.3.1. Sous-réseaux sub-symboliques / symboliques (cf. chapitre 3)

Dans le Deep Synergy Learning (DSL), il peut exister des **sous-réseaux sub-symboliques** (où les entités $\mathcal{E}_i$ sont décrites par des embeddings $\mathbf{x}_i$, et où la synergie $S_{\text{sub}}(i,j)$ se base sur des critères vectoriels) et des **sous-réseaux symboliques** (où d'autres entités $\mathcal{E}_p$ sont définies par des **règles** ou un **bloc** ontologique, et où $S_{\text{sym}}(p,q)$ repose sur la **compatibilité** logique). Ce modèle mixte, déjà évoqué au chapitre 3, illustre la capacité du DSL à intégrer, dans un **même cadre**, des entités numériques et des entités logiques.

Un **exemple** apparaît en **robotique** (chap. 3.6.3.3), avec des entités représentant des **perceptions** (embeddings sensoriels) et d'autres correspondant à des **actions** décrites symboliquement (précisant préconditions et effets). De même, pour un **agent conversationnel** (chap. 3.6.3.2), certaines entités sont des **segments textuels** (embedding BERT), tandis que d'autres englobent des **règles** de cohérence de dialogue.

Sur le plan **structurel**, on se retrouve souvent avec plusieurs "couches" d'entités :

- La couche "sub" manipule $\mathbf{x}_i$ et inclut des liens sub–sub,
- La couche "sym" gère les blocs logiques et inclut des liens sym–sym,
- Et il existe parfois des **passerelles** (liens sub–sym) reliant les deux couches.

L'**enjeu** majeur réside dans la capacité de chaque type d'entité à se **mettre à jour** (par exemple, une représentation sub-symbolique qui évolue, ou des règles symboliques qui se modifient), et dans la prise en compte du fait que la **synergie** varie selon la nature des entités connectées. Par conséquent, les mécanismes de mise à jour (additifs, multiplicatifs, saturation, etc.) décrits en 4.2.1

et 4.2.2 doivent être **adaptés** à la catégorie de la liaison (sub–sub, sym–sym ou sub–sym) pour demeurer cohérents avec les spécificités de chaque sous-réseau.

4.2.3.2. Mise à jour distincte selon le type de lien (ex. sub–sub, sym–sym, sym–sub)

Dans le cadre d'un **Deep Synergy Learning (DSL)** hétérogène, les entités peuvent appartenir à des natures différentes, qu'elles soient **sub-symboliques** (décrites par des **embeddings** tels que $\mathbf{x}_i \in \mathbb{R}^d$), **symboliques** (décrites par un ensemble de **règles**, axiomes ou ontologies, noté R_i) ou **hybrides** (combinant à la fois des représentations vectorielles et des blocs logiques). Cette diversité des représentations se traduit par des **types de liens** distincts dans le réseau : on distingue ainsi les liens **sub–sub** (entre deux entités sub-symboliques), les liens **sym–sym** (entre deux entités symboliques) et les liens **sym–sub** (passerelles entre le domaine symbolique et le domaine sub-symbolique). Chaque type de lien requiert une **mise à jour** adaptée, car la **fonction de synergie** $S(i,j)$ utilisée pour moduler l'évolution de la pondération $\omega_{i,j}$ diffère en fonction des caractéristiques de chaque modalité. Nous détaillons ci-après les principes sous-jacents à la mise à jour pour chacun de ces types de liens.

*A. **Liens sub–sub***

Lorsqu'un lien relie deux entités $\mathcal{E}_i$ et $\mathcal{E}_j$ qui sont toutes deux représentées par des **embeddings** $\mathbf{x}_i$ et $\mathbf{x}_j$ dans $\mathbb{R}^d$, la **synergie sub-symbolique** est généralement évaluée à l'aide de mesures vectorielles classiques. Par exemple, la **similarité cosinus** se définit par

$$S_{\text{sub}}(i,j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\| \mathbf{x}_i \| \| \mathbf{x}_j \|},$$

ce qui quantifie le degré de rapprochement entre les directions de ces vecteurs. Alternativement, on peut utiliser une mesure basée sur une fonction noyau, comme un noyau gaussien exprimé par

$$S_{\text{sub}}(i,j) = \exp\left(-\alpha \ \| \mathbf{x}_i - \mathbf{x}_j \|^2\right),$$

où $\alpha > 0$ règle la sensibilité du score. La **mise à jour** standard de la pondération dans ce cas se fait selon la règle additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S_{\text{sub}}(i,j) - \tau\, \omega_{i,j}(t)\right],$$

où η est le **taux d'apprentissage** et τ le **coefficient de décroissance** qui agit comme un terme de régularisation. Cette formulation, d'une complexité de l'ordre de $O(d)$ pour l'évaluation du score, permet de renforcer progressivement les liens entre des entités dont les représentations sont numériquement proches, par exemple deux images similaires ou deux segments textuels ayant des significations proches. Des variantes multiplicatives peuvent être envisagées si l'on souhaite accentuer l'effet exponentiel sur la croissance des liens, mais le principe fondamental demeure celui de la convergence vers un équilibre où, en situation stationnaire, $\omega^*_{i,j} \approx \frac{S_{\text{sub}}(i,j)}{\tau}$.

*B. **Liens sym–sym***

Dans le cas des liens **sym–sym**, les entités $\mathcal{E}_i$ et $\mathcal{E}_j$ sont caractérisées par des **blocs de règles** ou des structures logiques, notées R_i et R_j. La fonction de synergie symbolique $S_{\text{sym}}(i,j)$ évalue la

compatibilité entre ces ensembles de règles, ce qui peut être formulé de manière binaire ou graduée. Par exemple, on peut définir

$$S_{\text{sym}}(R_i, R_j) = \begin{cases} 1, & \text{si } R_i \text{ et } R_j \text{ sont entièrement compatibles,} \\ 0, & \text{en cas de contradiction totale,} \\ s_{ij}, & \text{si la compatibilité est partielle, avec } s_{ij} \in (0,1). \end{cases}$$

La mise à jour des pondérations pour les liens sym–sym suit un schéma similaire à celui des liens sub–sub, à savoir

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S_{\text{sym}}(i,j) - \tau\,\omega_{i,j}(t)\right],$$

ce qui permet de renforcer les connexions entre entités dont les blocs de règles se valident mutuellement. Toutefois, il faut noter que la comparaison symbolique peut s'avérer plus coûteuse en termes de calcul, surtout si l'inférence sur des systèmes logiques complexes est nécessaire. Des mécanismes d'**inhibition** ou des seuils peuvent être introduits pour limiter l'impact des incohérences et éviter une prolifération excessive des liens lorsque la logique est trop rigide.

C. Liens sym–sub

Les **liens sym–sub** constituent des passerelles entre les représentations sub-symboliques et symboliques. Dans ce cas, une entité $\mathcal{E}_i$ peut être représentée par un **embedding** $\mathbf{x}_i$ tandis qu'une autre entité $\mathcal{E}_j$ est décrite par un bloc de règles R_j, ou vice versa. Pour mesurer la synergie dans un tel cas, il est courant de définir une fonction hybride qui combine la **similarité vectorielle** et la **compatibilité logique**. Une telle fonction s'exprime par

$$S_{\text{hybrid}}(i,j) = \alpha\, S_{\text{sub}}(i,j) + (1-\alpha)\, S_{\text{sym}}(i,j),$$

où le paramètre $\alpha \in [0,1]$ ajuste l'importance relative des deux composantes. La mise à jour des pondérations pour ces liens hybrides est alors donnée par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S_{\text{hybrid}}(i,j) - \tau\,\omega_{i,j}(t)\right].$$

Cette approche permet de **fusionner** les informations provenant du monde numérique (embeddings) et du monde logique (règles), ce qui est particulièrement utile dans des applications où il est nécessaire de combiner la richesse des données sub-symboliques avec une **explicabilité** issue du raisonnement formel. Toutefois, l'évaluation simultanée des deux composantes peut augmenter la **complexité** de l'algorithme, et un réglage fin du paramètre α est indispensable pour obtenir le compromis optimal entre flexibilité et précision.

D. Conclusion sur la mise à jour multi-type

La mise à jour distincte selon le type de lien dans un DSL hétérogène illustre la capacité du système à adapter ses mécanismes d'auto-organisation aux caractéristiques spécifiques des représentations des entités. Pour les liens **sub–sub**, la mise à jour repose sur des calculs vectoriels efficaces et rapides, permettant de renforcer la similarité entre entités par des opérations de complexité $O(d)$. Pour les liens **sym–sym**, la dynamique repose sur une évaluation logique qui, bien que potentiellement plus coûteuse, offre une explicabilité et une cohérence formelle appréciables. Enfin, pour les liens **sym–sub**, une mise à jour hybride combine les avantages des deux approches

en ajustant les pondérations à l'aide d'un paramètre α qui permet de peser différemment la contribution de la similarité sub-symbolique et de la compatibilité symbolique.

Ainsi, le DSL hétérogène gagne en **souplesse** et en **capacité d'adaptation** en intégrant des règles de mise à jour différenciées pour chaque type de lien, ce qui permet de préserver une **dynamique** globale stable et cohérente malgré la diversité des représentations. Le contrôle de la **cohérence** du SCN repose sur une gestion fine de la **pondération** et de la **complexité** des interactions, garantissant que chaque type de liaison contribue de manière optimale à l'auto-organisation du réseau.

4.2.3.3. **Exemples de pseudo-code unifiant ces règles**

Dans un **Deep Synergy Learning (DSL)** hétérogène, la gestion des liens entre entités repose sur la mise à jour de pondérations $\omega_{i,j}$ qui varient selon la **nature** des entités en interaction. En effet, les entités peuvent être purement **sub-symboliques** (représentées par des **embeddings** $\mathbf{x}_i \in \mathbb{R}^d$), purement **symboliques** (définies par un ensemble de **règles** ou d'**axiomes** noté R_i), ou encore **hybrides** (combinant à la fois un embedding et un bloc logique). Pour chaque type de lien – qu'il s'agisse de liens sub–sub, sym–sym ou sym–sub – la **fonction de synergie** $S(i,j)$ et la stratégie de mise à jour doivent être adaptées. La présente section présente un **pseudo-code** unifié qui illustre comment, dans une même boucle d'itération, ces règles de mise à jour distinctes peuvent être intégrées de manière homogène dans l'algorithme de mise à jour des pondérations du **Synergistic Connection Network (SCN)**.

*A. **Présentation générale du cadre***

Soit un ensemble d'entités $\{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n\}$ dont chacune est associée à une représentation qui peut être sub-symbolique, symbolique ou hybride. Nous disposons d'une **matrice** $\{\omega_{i,j}\}$ représentant les pondérations actuelles entre chaque paire d'entités. Le système repose sur une fonction générique, que nous appellerons ComputeSynergy$(i,j,$link_type), qui, en fonction du **type de lien** entre $\mathcal{E}_i$ et $\mathcal{E}_j$ (déterminé par une fonction determine_link_type$(E[i], E[j])$), renvoie le score de synergie approprié. Ainsi, dans le cas d'un lien sub–sub, ComputeSynergy renverra un score $S_{\text{sub}}(i,j)$ calculé à partir des embeddings, typiquement par une mesure comme la similarité cosinus ou un noyau gaussien, tandis que pour un lien sym–sym, la fonction renverra un score $S_{\text{sym}}(i,j)$ évalué sur la compatibilité des règles, et dans le cas d'un lien sym–sub, une combinaison des deux scores sera utilisée, par exemple via une formule hybride.

La règle de mise à jour additive, commune à tous les types, se présente sous la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big],$$

où η est le **taux d'apprentissage** et τ le **coefficient de décroissance**. Des variantes multiplicatives peuvent également être envisagées, mais dans ce pseudo-code nous nous concentrerons sur la règle additive. Afin de contrôler la croissance excessive des pondérations, des mécanismes de **clipping** (pour borner les valeurs) et d'**inhibition** (pour tenir compte de la compétition entre liens) sont intégrés dans la mise à jour.

B. ***Pseudo-code unifié***

Le pseudo-code suivant, de style Python, illustre comment on peut unifier les mises à jour des différents types de liens dans une boucle d'itération :

```
# Paramètres globaux
eta_sub   = 0.01    # Taux d'apprentissage pour liens sub–sub
eta_sym   = 0.02    # Taux pour liens sym–sym
eta_hyb   = 0.015   # Taux pour liens sym–sub
tau       = 0.1     # Coefficient de décroissance
gamma     = 0.05    # Coefficient pour inhibition (optionnel)
omega_max = 5.0     # Valeur maximale pour le clipping
USE_INHIBITION = True  # Option pour activer le mécanisme d'inhibition

# Boucle principale sur toutes les entités
for i in range(n):              # Pour chaque entité i
   for j in range(n):           # Pour chaque entité j
      if i == j:
         continue           # Pas de mise à jour pour i=j

      # Déterminer le type de lien entre E[i] et E[j]
      link_type = determine_link_type(E[i], E[j])
      # Exemples de valeurs possibles: "sub-sub", "sym-sym", "sym-sub"

      # Calcul de la synergie S(i,j) en fonction du type de lien
      S_ij = ComputeSynergy(E[i], E[j], link_type)
      # Par exemple, pour un lien sub–sub, ComputeSynergy renvoie:
      #   S_sub = (E[i].x dot E[j].x) / (||E[i].x|| * ||E[j].x||)
      # Pour un lien sym–sym, il renvoie un score basé sur la compatibilité logique
      # Pour un lien sym–sub, il combine les deux scores via S_hybrid = alpha*S_sub + (1-alpha)
*S_sym

      # Calcul optionnel d'un terme d'inhibition, basé sur la somme des poids sortants de i
      inhib_term = 0.0
      if USE_INHIBITION:
         sum_weights = 0.0
         for k in range(n):
            if k != j:
               sum_weights += w[i][k]
         inhib_term = gamma * sum_weights

      # Sélection du taux d'apprentissage spécifique selon le type de lien
      if link_type == "sub-sub":
         learning_rate = eta_sub
      elif link_type == "sym-sym":
         learning_rate = eta_sym
      else:  # pour "sym-sub"
```

```
        learning_rate = eta_hyb

    # Mise à jour additive de la pondération
    # Formule : w_new = w(i,j) + learning_rate * (S_ij - tau*w(i,j)) - inhib_term
    w_new = w[i][j] + learning_rate * (S_ij - tau * w[i][j]) - inhib_term

    # Application d'un clipping pour éviter des valeurs extrêmes
    if w_new < 0:
        w_new = 0
    if w_new > omega_max:
        w_new = omega_max

    # Mise à jour de la matrice de pondérations
    w[i][j] = w_new
```

Explications détaillées :

Dans ce pseudo-code, la variable ***E*** représente la liste des entités du DSL. Chaque entité peut être accompagnée d'un **embedding** (pour les entités sub-symboliques), d'un **bloc de règles** (pour les entités symboliques) ou d'une combinaison des deux (pour les entités hybrides). La fonction ***determine_link_type(E[i], E[j])*** permet de distinguer le type de lien à établir entre deux entités en se basant sur leur nature respective. Par exemple, si deux entités possèdent toutes deux un embedding, le lien est qualifié de **sub–sub** ; si elles sont toutes deux décrites par des règles, le lien est **sym–sym** ; et si l'une est sub-symbolique et l'autre symbolique, le lien est dit **sym–sub**.

La fonction ***ComputeSynergy(E[i], E[j], link_type)*** renvoie le score de synergie approprié en fonction du type de lien. Pour les liens sub–sub, ce score est généralement calculé à partir de mesures classiques telles que la similarité cosinus ou une fonction de noyau gaussien, tandis que pour les liens sym–sym, il s'agit d'un score reflétant la compatibilité ou l'absence de contradictions entre les blocs logiques R_i et R_j. Pour les liens sym–sub, on combine les deux approches via une formule hybride, par exemple

$$S_{\text{hybrid}}(i,j) = \alpha\, S_{\text{sub}}\big(\mathbf{x}_i, \mathbf{x}_j\big) + (1-\alpha)\, S_{\text{sym}}\big(R_i, R_j\big).$$

Le **terme d'inhibition** est calculé (si activé) en sommant les pondérations sortantes d'une entité pour modérer la croissance excessive des liens et éviter une saturation du réseau. Chaque type de lien peut bénéficier d'un **taux d'apprentissage** distinct, ici symbolisé par η_{sub}, η_{sym} et η_{hyb} pour les liens sub–sub, sym–sym et sym–sub, respectivement. La mise à jour s'effectue de manière additive, comme le montre la formule utilisée, puis le **clipping** est appliqué pour garantir que les pondérations restent dans un intervalle prédéfini, par exemple $[0, \omega_{\max}]$.

Ce pseudo-code s'exécute de manière itérative sur l'ensemble des entités, et peut être répété pour de nombreuses itérations jusqu'à convergence ou pendant une durée spécifiée. Il offre une illustration concrète de la façon dont un **DSL** peut gérer simultanément des règles de mise à jour différenciées pour plusieurs types de liens, en assurant une intégration harmonieuse des informations sub-symboliques et symboliques.

Conclusion

Le présent pseudo-code constitue un exemple unifié qui intègre les différentes **règles de mise à jour** selon le type de lien dans un DSL hétérogène. En adaptant dynamiquement le taux d'apprentissage et en appliquant des mécanismes complémentaires tels que l'inhibition et le clipping, le système parvient à mettre à jour les pondérations $\omega_{i,j}$ de manière cohérente, tout en tenant compte des spécificités des interactions sub–sub, sym–sym et sym–sub. Ce schéma permet ainsi d'unifier la mise à jour des liens dans un SCN, assurant la **robustesse** et la **flexibilité** de l'auto-organisation même dans un environnement où coexistent plusieurs types de représentations.

4.3. Émergence de Clusters et Attracteurs

Dans les sections précédentes (4.2), nous avons décrit les différentes **règles** de mise à jour permettant aux pondérations $\omega_{i,j}$ d'évoluer sous l'effet de la synergie $S(i,j)$, de termes de décroissance, d'inhibition, ou de saturation. L'un des **résultats majeurs** de cette dynamique d'auto-organisation est l'**émergence** spontanée de **clusters** — c'est-à-dire de sous-ensembles d'entités qui se lient fortement entre elles, tout en se distançant des autres.

En creusant ce phénomène, on découvre également la notion d'**attracteurs** : selon la configuration initiale et la structure de la synergie, le **SCN** (Synergistic Connection Network) peut aboutir à différents **points fixes** (clusters stabilisés), ou manifester des comportements plus riches (multi-stabilité, oscillations). Cette section (4.3) analysera donc :

3. Comment les **clusters** se forment (4.3.1),
4. Les concepts d'**attracteurs multiples** et de **bascules** (4.3.2),
5. Les scénarios de **stabilisation** ou de **dynamisme continu** (4.3.3), montrant que le réseau peut se figer ou rester en mutation constante, selon le flux de synergie et l'ajout de nouvelles données.

4.3.1. Formation Spontanée de Groupes

Lorsque l'on met en place les **mécanismes** décrits en 4.2 (mise à jour additive ou multiplicative, inhibition, etc.), les **entités** $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ ne tardent pas à voir leurs liens $\omega_{i,j}$ s'**organiser** : certains liens deviennent **forts**, d'autres s'étiolent, et des **groupes** se forment. C'est précisément ce qui confère au **DSL** (Deep Synergy Learning) sa **capacité** d'auto-organisation : en "laissant faire" la dynamique, on aboutit à des **clusters** d'entités hautement interconnectées, tout en limitant les connexions hors-clusters.

4.3.1.1. Notion de Clusters : pondérations fortes à l'intérieur d'un sous-ensemble $\mathcal{C}$, faibles à l'extérieur

Lorsqu'on examine la structure d'un **Synergistic Connection Network (SCN)**, il est courant d'observer l'émergence de groupes d'entités, appelés **clusters** ou **communautés**, à l'intérieur desquels les **pondérations** $\omega_{i,j}$ sont nettement plus **élevées** que celles reliant ces entités au **reste** du réseau. Autrement dit, un **sous-ensemble** $\mathcal{C} \subseteq \{1, \dots, n\}$ va exhiber des liens $\omega_{i,j}$ particulièrement **forts** pour $(i,j \in \mathcal{C})$, tandis que les connexions $\omega_{i,k}$ vers un $k \notin \mathcal{C}$ resteront **relativement faibles**.

Pour illustrer cette idée, on peut considérer un exemple simple : si $\mathcal{C} = \{2,5,7\}$ forme un cluster, on **attend** à l'issue de la dynamique DSL que $\omega_{2,5}$, $\omega_{5,7}$ et $\omega_{2,7}$ deviennent **hautes**, tandis que les liaisons $\omega_{2,k}, \omega_{5,k}, \omega_{7,k}$ pour $k \notin \{2,5,7\}$ demeurent **faibles** ou se réduisent au fil des itérations.

La **raison mathématique** de cette formation de clusters tient à la **synergie** : lorsque, pour un ensemble donné d'entités, la fonction $S(i,j)$ est (en moyenne) plus **élevée** entre elles qu'avec l'extérieur, la mise à jour

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\big[S(i,j) \;-\; \tau\,\omega_{i,j}(t)\big]$$

tend à **renforcer** progressivement $\omega_{i,j}$ pour $(i,j \in \mathcal{C})$. Symétriquement, dès lors que la synergie entre $(i \in \mathcal{C})$ et $(k \notin \mathcal{C})$ est plus faible, la pondération $\omega_{i,k}(t)$ s'**amenuise** ou reste basse, et ne concurrence pas la liaison intra-groupe.

Un aspect central de cette dynamique est l'**absence de supervision** : le système n'exige aucun label ou contrainte externe pour "dire" quelles entités devraient s'assembler. Seuls les **patterns** de synergie, qu'ils soient issus de la similarité (embeddings, co-occurrences) ou d'une logique symbolique (chap. 3.5), suffisent à produire une **dépendance mutuelle** entre entités. Si la synergie révèle des affinités nettement plus fortes au sein du sous-ensemble $\mathcal{C}$, la **dynamique** DSL consolide précisément ces liens.

Sur le plan **théorique**, ce processus rejoint l'idée du **clustering** en apprentissage non supervisé, à ceci près que la dynamique DSL reste **entièrement itérative** et **coopérative**. Au lieu de définir un algorithme statique (k-means ou agglomératif) qui opère en une passe sur les données, on laisse la **matrice** $\{\omega_{i,j}\}$ évoluer au fil du temps et, de manière **auto-organisée**, "dessiner" des **îlots** fortement interconnectés.

Une fois l'étape de **convergence** atteinte, la matrice ω révèle explicitement l'existence de **clusters** : on peut repérer ces groupes en appliquant un **seuil** sur $\omega_{i,j}$ (les plus fortes valeurs identifient les sous-ensembles fortement liés), ou en recourant à des méthodes de **community detection** (louvain, modularité, etc.) sur le graphe formé par ω.

Sur le plan **macro**, cette **formation de clusters** marque l'un des aboutissements majeurs de la **synergie** : le SCN, parti d'entités possiblement dispersées et de pondérations initiales faibles ou aléatoires, se **structure** en regroupements cohérents, reflétant la **cohésion** intrinsèque que la fonction S a détectée dans les données. Cette propriété d'**auto-organisation** constitue l'un des piliers fondamentaux du DSL, montrant comment, à partir de **règles** locales et d'ajustements distribués, le réseau peut dégager une **partition** (partielle ou totale) du jeu d'entités en **communautés** pertinentes, sans qu'aucun paramétrage ou nombre de clusters ne soit imposé a priori.

4.3.1.2. Processus Auto-Organisé : sans supervision, certains liens se renforcent massivement (haute synergie), d'autres décroissent

Le **phénomène** de formation de **clusters** décrit en section 4.3.1.1 ne repose pas sur l'utilisation d'un algorithme de **clustering** classique ni sur la présence de **labels** ou de **catégories** prédéfinies. Au contraire, il résulte d'un **processus auto-organisé** intrinsèque aux pondérations $\omega_{i,j}$, lesquelles évoluent en fonction de la **synergie** $S(i,j)$ et des règles de mise à jour (décrites au paragraphe 4.2). C'est précisément ce caractère **non supervisé** et **dynamique** qui caractérise le **DSL** (Deep Synergy Learning) et qui en fait sa spécificité.

Caractère non supervisé.

Dans une configuration d'**apprentissage supervisé**, on dispose généralement de classes ou de cibles explicites pour guider la création de groupes. Ici, la démarche ne s'appuie sur **aucune** annotation : seule la **synergie** $S(i,j)$ — qu'elle provienne de similarités sub-symboliques, de compatibilités logiques ou d'autres sources (cf. chapitre 3) — oriente la **croissance** ou la **décroissance** des pondérations $\omega_{i,j}$. L'évolution du réseau reste ainsi *auto-organisée*, car elle ne fait appel à aucun signal externe imposant la façon de répartir ou de regrouper les entités.

L'**analogie** avec certains réseaux neuronaux biologiques est souvent mise en avant : dans le cerveau, les synapses voient leur force modifiée selon l'activité neuronale partagée (une certaine forme de "co-activation" ou de "synergie"), en l'absence d'un "superviseur" qui affecterait manuellement des étiquettes à chaque neurone. Le **DSL** transpose ces principes à des entités abstraites (tels que des textes, des images, des règles logiques, etc.), dont les liens se renforcent ou s'affaiblissent selon leur synergie mutuelle.

Renforcement massif de certains liens, décroissance des autres.

Imaginons qu'à l'initialisation, certaines paires (i,j) présentent une **synergie** $S(i,j)$ relativement élevée, tandis que d'autres pairs (i,k) sont moins corrélées. La dynamique DSL (qu'elle soit **additive** ou **multiplicative**, avec ou sans mécanismes d'**inhibition**, de **saturation**, etc.) va **favoriser** la montée en puissance de $\omega_{i,j}$ dès lors que $S(i,j)$ se révèle important. Au fil des itérations, la pondération $\omega_{i,j}$ tend à **se stabiliser** (voire à continuer d'augmenter si les régulations le permettent), constituant ainsi un **cœur** de liaisons robustes.

De manière symétrique, les couples (i,k) pour lesquels la synergie $S(i,k)$ se montre plus faible subissent l'effet de **décroissance** lié à $-\tau\,\omega_{i,k}(t)$ (et éventuellement à l'inhibition compétitive). Cela se traduit par une **baisse** progressive, voire une extinction, de $\omega_{i,k}$. Au terme de ce processus, certains liens s'en trouvent **renforcés** de façon marquée, et la majorité finit par s'**affaiblir** notablement. Cette situation aboutit à des **"îlots de forte connectivité"**, constituant les futurs **clusters**.

Dans les versions dites **multiplicatives** de la mise à jour, un léger avantage initial d'une pondération peut conduire à une **croissance exponentielle**, accentuant encore la séparation entre les connexions "vainqueurs" (dotées d'une synergie forte) et les connexions "perdantes" (qui demeurent modestes ou décroissent). Même en mode **additif**, un contraste initial de synergie suffit à créer, à l'issue de la dynamique, un écart stable : l'une des liaisons converge vers $S(i,j)/\tau$, l'autre vers $S(i,k)/\tau$. Si $S(i,j) > S(i,k)$, la différence se creuse inévitablement.

Émergence de clusters.

À mesure que ces liaisons puissantes se démarquent et que la majorité des autres s'affaiblit, on observe la formation naturelle de **sous-ensembles** d'entités — c'est-à-dire de **clusters** — unis par des pondérations $\omega_{i,j}$ élevées. Au plan géométrique ou sémantique, cette différence renvoie à la **répartition** des synergies S dans l'espace des données : là où $S(i,j)$ s'avère plus grand, la pondération $\omega_{i,j}$ finit par créer des **composantes** ou **communautés** (au sein du SCN). La **matrice** $\{\omega_{i,j}\}$ se "sculpte" littéralement sous l'action de la dynamique DSL, laissant apparaître des **îlots de forte connectivité**.

En pratique, on peut suivre, au fil des itérations, la progression vers des "clusters" plus ou moins stables. Durant les premiers pas, plusieurs connexions peuvent simultanément croître, puis l'**inhibition** ou la **saturation** vont faire émerger des groupes plus définis, car un lien fort d'une entité vers un ensemble donné empêche la prolifération de liens simultanés vers d'autres ensembles. Selon les conditions initiales (valeurs de η, τ, coefficient d'inhibition γ, etc.) et la distribution des synergies, le réseau finit fréquemment par converger vers un **état stable** (ou un quasi-état stable), découpant implicitement l'ensemble des entités en **sous-groupes**. Cette séparation se produit sans affectation de labels ni contrainte externe, illustrant la pleine **auto-organisation** : la seule fonction S, intégrée à la mise à jour, produit l'**émergence** de **clusters** pertinents.

Conclusion : la dynamique auto-organisée comme moteur de clusterisation.

Les pondérations $\omega_{i,j}$, guidées par la **synergie** $S(i,j)$ et la **règle** de mise à jour locale, aboutissent à la consolidation d'un ensemble fini de **communautés** d'entités, sans imposer de supervision. Cette propriété joue un rôle clé dans la "découverte" de groupes cohérents : le réseau se **structure** de l'intérieur, révélant en filigrane la **topologie** ou la **sémantique** dissimulée dans la mesure de synergie. Cet **auto-appariement** de connexions "fortes" fait émerger naturellement une **partition** globale du SCN, inaugurant l'étude de la **multi-stabilité** et des **attracteurs** (section 4.3.2) qui analysent les états finaux de cette dynamique.

4.3.2. Attracteurs Multiples et Bascules

Lorsque la dynamique d'auto-organisation (chap. 4.2) agit sur les pondérations $\omega_{i,j}$, on peut observer divers **phénomènes** de stabilité ou d'instabilité. En particulier, il arrive que le **SCN** (Synergistic Connection Network) dispose de **plusieurs** configurations stables (ou quasi-stables), vers lesquelles il peut converger selon l'initialisation des $\omega_{i,j}$ ou l'évolution du **flux de synergie** $S(i,j)$. C'est ce qu'on appelle la **multi-stabilité**, ou la co-existence de **plusieurs attracteurs**.

Par "**attracteurs** multiples", on entend que le système (la matrice $\{\omega_{i,j}\}$) peut aboutir à différentes **configurations finales** (points fixes, cycles, régimes complexes) selon les conditions initiales ou la chronologie des mises à jour. On peut alors assister à des **basculements** entre attracteurs, si le **flux** $S(i,j)$ ou d'autres paramètres changent en cours de route.

4.3.2.1. Multi-Stabilité : un SCN peut atteindre différents points fixes selon l'initialisation ou le flux de synergie (2.3.2)

A. Rappel mathématique : points fixes et stabilité

Les réseaux **DSL** (Deep Synergy Learning), ou plus largement les systèmes de type **SCN** (Synergistic Connection Network), se décrivent par un ensemble de liens $\omega_{i,j}(t)$ qui évoluent au fil du temps. De manière formelle, on peut écrire cette évolution sous la forme d'une **application F** opérant sur la configuration globale $\boldsymbol{\omega}(t)$. À chaque pas d'itération, la dynamique s'écrit :

$$\boldsymbol{\omega}(t+1) \;=\; \mathbf{F}\big(\boldsymbol{\omega}(t)\big),$$

où $\boldsymbol{\omega}(t)$ représente le vecteur (ou la matrice) regroupant l'ensemble $\{\omega_{i,j}(t)\}$. Un **point fixe** $\boldsymbol{\omega}^*$ de ce système satisfait

$$\boldsymbol{\omega}^* = \mathbf{F}(\boldsymbol{\omega}^*).$$

Lorsqu'on a affaire à une fonction **F non linéaire**, il peut exister plusieurs **solutions** $\boldsymbol{\omega}_1^*, \boldsymbol{\omega}_2^*, \ldots$ satisfaisant cette même équation, chaque solution correspondant à un **arrangement stable** des pondérations $\omega_{i,j}$. Dans la théorie des systèmes dynamiques, chacun de ces états constitue un **attracteur local**. Ainsi, selon l'**initialisation** $\boldsymbol{\omega}(0)$ ou les perturbations ultérieures, la trajectoire $\boldsymbol{\omega}(t)$ peut aboutir à l'un ou l'autre de ces points fixes, chacun possédant son **bassin d'attraction** propre.

Dans le cadre spécifique du **DSL**, la fonction **F** dépend de la **synergie** $S(i,j)$. Si la synergie est **constante** (aucune variation au fil du temps, aucun nouveau flux de données), la multi-stabilité découle essentiellement de la **non-linéarité** (par exemple l'inhibition compétitive ou le mécanisme multiplicatif). Au contraire, si la synergie $S(i,j)$ évolue (arrivée de nouvelles entités, changement de paramètres), le système peut, pendant sa trajectoire temporelle, **changer** d'attracteur si le "paysage" dynamique se modifie suffisamment.

B. Exemples conceptuels de multi-stabilité

Pour se faire une intuition, il est possible d'imaginer un **mini réseau** de seulement trois entités $\{1,2,3\}$ et de se donner une mise à jour additive ou multiplicative. Le flux de synergies $S(1,2)$, $S(2,3)$, $S(1,3)$ peut être choisi de façon à ce que deux configurations concurrentes soient toutes deux "plausibles". Dans la première, les entités $\{1,2\}$ forment un cluster, excluant la troisième, tandis que dans la seconde, ce sont $\{2,3\}$ qui se solidarisent. Si l'initialisation favorise l'émergence de $\{1,2\}$ (par un bruit initial en ce sens), on converge naturellement vers cette partition, tandis qu'une initialisation différente fait converge vers $\{2,3\}$.

À plus grande échelle, si l'on dispose de nombreuses entités, il peut y avoir **plusieurs partitions** tout aussi cohérentes. La nature **non linéaire** de la règle (par exemple l'inhibition compétitive) accentue cette tendance : un groupe se forme et se stabilise d'un côté, un autre se stabilise différemment d'un autre côté, et il se peut que deux exécutions nominalement équivalentes aboutissent à des **arrangements** distincts. Cela renvoie à la notion de **paysage d'énergie** comportant plusieurs "puits" locaux, chacun correspondant à un **attracteur** stable. Ces principes sont largement étudiés dans la littérature des systèmes dynamiques et rappellent, dans une certaine mesure, les réseaux de Hopfield (où plusieurs états mémorisés coexistent comme attracteurs), ou encore les chaînes de Markov possédant différents états absorbants.

C. Rôle de l'initialisation et du flux de synergie

La **dépendance** vis-à-vis de l'initialisation $\boldsymbol{\omega}(0)$ se comprend en analysant la trajectoire donnée par $\boldsymbol{\omega}(t+1) = \mathbf{F}\big(\boldsymbol{\omega}(t)\big)$. Très souvent, on part d'une matrice $\omega_{i,j}(0)$ proche de zéro (ou perturbée par un faible bruit gaussien). Mais de petites variations dans cette phase initiale peuvent se répercuter fortement, en raison des **bifurcations** qu'introduit la non-linéarité.

Si la **synergie** $S(i,j)$ reste **constante** dans le temps, le système évolue vers l'un de ses points fixes disponibles, typiquement en une seule "phase" de convergence. En revanche, si S **varie** (arrivée de nouvelles données, recalcul d'embeddings, etc.), la fonction **F** se modifie, et un attracteur qui était

stable peut cesser de l'être. Le **réseau** peut alors **basculer** vers un autre attracteur plus conforme au nouveau contexte, parfois par un phénomène de "cascade" de réajustements.

Les **perturbations** extérieures (inhibition accrue, bruit stochastique, recuit simulé) peuvent également déstabiliser l'état courant et "pousser" le réseau vers une autre configuration stable.

D. Illustration mathématique : Jacobienne et stabilité locale

Pour analyser la stabilité locale d'un point fixe, on se réfère à la **matrice jacobienne** $\mathrm{D}\mathbf{F}(\boldsymbol{\omega}^*)$. Étant donné la dynamique

$$\boldsymbol{\omega}(t+1) \;=\; \mathbf{F}\big(\boldsymbol{\omega}(t)\big),$$

la **condition** de stabilité veut que l'opérateur linéarisé $\mathrm{D}\mathbf{F}(\boldsymbol{\omega}^*)$ (qui décrit la sensibilité à de petites variations autour de $\boldsymbol{\omega}^*$) ait un rayon spectral (ou une norme appropriée) < 1. Cela garantit que la trajectoire reste dans l'orbite de $\boldsymbol{\omega}^*$ et ne diverge pas.

Lorsque $\mathrm{D}\mathbf{F}(\boldsymbol{\omega}^*)$ permet à plusieurs $\boldsymbol{\omega}^*$ d'être stables, on parle de **co-existence** d'attracteurs. Chaque attracteur dispose de son **bassin**. Dans un cadre DSL de grande dimension, ce phénomène de co-existence peut amener à une grande diversité de partitions possibles, ou à des états stables différents selon l'histoire de la trajectoire.

E. Basculement entre attracteurs

Le fait de **changer** un paramètre η, γ ou τ, de faire **arriver** un nouveau flux de données, ou de recalculer **S** peut soudainement **rendre** un attracteur obsolète et permettre à la dynamique de *s'échapper* vers une configuration plus basse en énergie. Dans la théorie des systèmes dynamiques, on parle alors de **bifurcation**, ou de "transition de phase" lorsqu'on l'analyse avec une perspective inspirée de la physique statistique.

À grande échelle, on peut **voir** ce basculement se traduire par l'évolution de la matrice ω : un bloc d'entités qui formait un cluster disparaît subitement ou se scinde en deux, tandis qu'un autre cluster émerge ou fusionne avec un tiers. Tout cela résulte d'une multitude de **petits ajustements** $\Delta\omega_{i,j}$ qui, accumulés, aboutissent à un **changement** macroscopique.

F. Conclusion : la multi-stabilité comme source de diversité

La présence de **plusieurs** solutions stables dans un SCN souligne la **richesse** des dynamiques non linéaires : non seulement il existe la possibilité de se fixer sur un arrangement unique, mais il peut y avoir **divers** arrangements également stables (ou métastables). Cette **co-existence** explique comment deux exécutions, pourtant équivalentes en apparence, produisent des **partitions** légèrement ou radicalement différentes.

Un simple décalage dans l'initialisation, ou un changement progressif du flux **S**, peut faire **sauter** le réseau d'un attracteur à l'autre. Dans certains cas, ce phénomène se révèle bénéfique, car il autorise le système à **s'adapter** aux conditions (nouveaux types d'entités, re-paramétrage).

Sur le plan de l'**auto-organisation**, la multi-stabilité illustre la **diversité** des *solutions* ou *consensus* qu'un DSL peut obtenir, reflétant divers compromis entre les synergies présentes. Elle participe aussi de la **flexibilité** d'un SCN, qui ne s'enferme pas nécessairement dans une unique forme de partition, mais peut explorer plusieurs structures cohérentes.

Dans la prochaine étape (section 4.3.2.2), on considérera que la dynamique ne se limite pas à converger vers un point fixe : il se peut qu'elle donne naissance à des **cycles** ou **oscillations**, c'est-à-dire à des attracteurs d'une nature plus mobile que le simple état stationnaire.

4.3.2.2. Cycles ou "Oscillations" : si la fonction de mise à jour favorise des rétroactions positives, on peut avoir des boucles (2.3.2.2)

Dans la continuité du phénomène de **multi-stabilité** (voir 4.3.2.1), il arrive qu'un **SCN** (Synergistic Connection Network) n'aboutisse pas à un point fixe statique mais entre dans un **cycle** récurrent ou une forme d'**oscillation**. Au lieu de converger vers une configuration unique, la dynamique parcourt successivement plusieurs états qui se répètent de manière périodique (cycle de période T) ou quasi-périodique (oscillation complexe). Cette situation survient lorsque les **rétroactions positives** dans la mise à jour de $\omega_{i,j}$ sont suffisamment fortes pour qu'aucun état stationnaire ne s'impose durablement.

A. Rappel conceptuel : rétroactions positives et non-linéarités

Lorsque, dans la règle de mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \cdots$, la contribution qui accroît $\omega_{i,j}$ se trouve amplifiée dès que $\omega_{i,j}$ est déjà élevée, on parle de **rétroaction positive**. Si cette force d'amplification n'est pas contrebalancée par un mécanisme stabilisateur (par exemple une décroissance assez puissante, ou une inhibition globale), la trajectoire de $\omega_{i,j}$ peut se mettre à fluctuer au lieu de s'ancrer à une valeur fixe.

Ces comportements de fluctuation ou de "va-et-vient" sont généralement liés à des **non-linéarités**. Dans un **DSL** (Deep Synergy Learning), plusieurs ingrédients non linéaires peuvent intervenir : la mise à jour multiplicative ($\omega_{i,j}(t+1) = \omega_{i,j}(t) \times (1 + \cdots)$), l'inhibition compétitive qui ajoute un terme $-\gamma \sum_{k \neq j} \omega_{i,k}$, ou encore la saturation ($\omega_{\max}$). Les théories des **systèmes dynamiques** enseignent que de tels couplages non linéaires suffisent à produire des solutions périodiques (cycles limites) ou, plus rarement, des régimes chaotiques.

B. Caractérisation mathématique : cycle de période T

Sur le plan purement mathématique, un **cycle** de période T pour la dynamique

$$\boldsymbol{\omega}(t+1) \;=\; \mathbf{F}\big(\boldsymbol{\omega}(t)\big)$$

se définit comme une suite de configurations $\{\boldsymbol{\omega}^*(1), \dots, \boldsymbol{\omega}^*(T)\}$ telle que

$$\mathbf{F}\big(\boldsymbol{\omega}^*(t)\big) \;=\; \boldsymbol{\omega}^*\big((t \bmod T) + 1\big), \quad t = 1, \dots, T,$$

et $\boldsymbol{\omega}^*(T+1) = \boldsymbol{\omega}^*(1)$. Autrement dit, après avoir appliqué $\mathbf{F}$ T fois, on revient à l'état initial $\boldsymbol{\omega}^*(1)$.

Dans la pratique, un **cycle de période 2** est particulièrement fréquent : la configuration oscille en "ping-pong" entre deux états. Des cycles de période 3 ou plus longs existent également. Comme pour les points fixes, on peut analyser la **stabilité** de ces cycles en examinant l'évolution locale sur une orbite complète. Si l'analyse (via la matrice dérivée sur ce tour d'orbite) montre que les perturbations y restent contenues ou se contractent, le cycle est stable ; sinon, il peut s'avérer instable et ne se maintenir qu'en conditions idéales.

C. Illustrations dans un DSL

Dans un **DSL** de dimension significative, ces cycles peuvent se traduire par des phénomènes de "fluctuation de clusters". Par exemple, un ensemble d'entités se regroupe momentanément en un bloc cohérent, puis la dynamique se déplace pour défaire ce cluster et en bâtir un autre, et la configuration revient ensuite au premier arrangement. Cette succession décrit un **cycle** : l'état global du SCN n'est jamais figé, mais repasse par un chemin périodique dans l'espace des $\{\omega_{i,j}\}$.

Les **transitions** causées par l'inhibition compétitive sont un autre vecteur classique d'oscillations. Imaginons un triangle $\{1,2,3\}$: si l'on renforce d'abord $\omega_{1,2}$ au détriment de $\omega_{2,3}$, puis que $\omega_{2,3}$ repart à la hausse et inhibe $\omega_{1,2}$, etc. Le système tourne en boucle et ne se stabilise jamais sur un unique cluster. Des simulations minimalistes montrent aisément cette alternance.

D. Signification et implications

Un tel **cycle** s'oppose à l'idée d'une convergence vers un unique arrangement stable. La présence de rétroactions positives et de couplages non linéaires signifie que le réseau peut "tourner" dans l'espace des configurations. Sur un plan pratique, cela peut compliquer l'exploitation des résultats : s'il n'y a pas d'état final unique, on ne peut pas conclure à une partition définitive. Cependant, ces oscillations peuvent être recherchées lorsqu'on désire un comportement de type "exploration continue" ou qu'on souhaite que le SCN "réfléchisse" en alternant plusieurs configurations plausibles.

Dans les grands systèmes, on peut même entrevoir des régimes plus complexes, y compris **chaotiques**, quoique dans un DSL standard, la plupart des mécanismes (décroissance τ, saturation, etc.) limitent souvent l'apparition d'un véritable chaos. On assiste plus fréquemment à des cycles stables (limite cycles) ou des oscillations amorties. Selon les objectifs, on peut vouloir favoriser ces oscillations (pour explorer plus de partitions) ou les réduire (pour atteindre un état stable), en ajustant la force de l'inhibition γ ou de la décroissance τ.

E. Conclusion

La **dynamique** d'un **SCN** n'est donc pas invariablement condamnée à converger vers une partition fixe : l'**exemple** des **cycles** ou oscillations récurrentes démontre que, sous l'effet de rétroactions positives et de couplages non linéaires, l'évolution de $\{\omega_{i,j}(t)\}$ peut se maintenir dans une orbite périodique plutôt que se figer. Dans certains contextes, cette oscillation correspond à une **exploration** alternée de plusieurs regroupements concurrents ; dans d'autres, elle peut constituer un **phénomène perturbateur** à modérer. Les sections ultérieures (4.3.2.3) discuteront des stratégies pour sortir d'une oscillation indésirable (par exemple via recuit simulé) ou, au contraire, pour les exploiter lorsqu'elles sont bénéfiques à la dynamique globale du réseau.

4.3.2.3. Stratégies pour détecter / caractériser un attracteur : qu'est-ce qu'un "bassin d'attraction" ?

Dans les sections précédentes (4.3.2.1 et 4.3.2.2), nous avons vu qu'un **SCN** (Synergistic Connection Network) pouvait admettre divers types d'attracteurs, tels que des **points fixes** (multi-stabilité) ou des **cycles** (oscillations). Une question fondamentale se pose alors : comment **déterminer** qu'un réseau a effectivement atteint un attracteur ? Et comment **décrire** la "zone d'influence" de cet attracteur, connue sous le nom de **bassin d'attraction** ?

Il s'agit d'un sujet central en **théorie des systèmes dynamiques** et en **analyse numérique**, qui, dans le cadre d'un DSL, peut être abordé à la fois par des méthodes analytiques (examen de dérivées, jacobiennes) et par des méthodes empiriques (simulation, exploration des conditions initiales).

A. Qu'est-ce qu'un attracteur ?

Un **attracteur** se définit comme un **ensemble** (point fixe, cycle limite, orbite quasi-périodique, attracteur chaotique, etc.) auquel un grand nombre de **trajectoires** convergent — ou auquel elles se maintiennent de façon récurrente. Une fois la dynamique $\{\omega_{i,j}(t)\}$ entrée dans un voisinage de cet attracteur, elle n'en sort plus (ou s'en éloigne très difficilement).

Lorsque l'attracteur est un **point fixe** $\boldsymbol{\omega}^*$, on a

$$\boldsymbol{\omega}^* = \mathbf{F}(\boldsymbol{\omega}^*),$$

et, à partir d'initialisations proches, les trajectoires $\boldsymbol{\omega}(t)$ convergent vers $\boldsymbol{\omega}^*$. Si l'attracteur est un **cycle** de période T, un ensemble de configurations $\{\boldsymbol{\omega}^*(1), \dots, \boldsymbol{\omega}^*(T)\}$ se répète : après T applications de **F**, l'état du SCN retrouve exactement sa valeur initiale, et les trajectoires voisines viennent s'« attacher » à ce cycle.

Un **exemple** en DSL pourrait être la formation définitive d'un cluster stable, ou bien une oscillation entre deux ou trois « configurations » récurrentes dans lesquelles les liaisons fortes changent tour à tour.

B. Détection et caractérisation numérique

1. Recherche d'un point fixe

En pratique, on itère la dynamique

$$\boldsymbol{\omega}(t+1) = \mathbf{F}\big(\boldsymbol{\omega}(t)\big)$$

et on surveille l'évolution du **pas** :

$$\| \boldsymbol{\omega}(t+1) - \boldsymbol{\omega}(t) \|.$$

Si cette norme devient (et reste) inférieure à un seuil ε suffisamment petit, on conclut à l'approche d'un **point fixe**. Concrètement, on peut poser une condition comme

$$\| \boldsymbol{\omega}(t+1) - \boldsymbol{\omega}(t) \| < \varepsilon \quad \text{et} \quad \| \boldsymbol{\omega}(t+\Delta t) - \boldsymbol{\omega}(t) \| < \varepsilon \quad \text{sur quelques pas successifs,}$$

pour s'assurer qu'aucun ré-emballement ne se produit. On déclare alors la convergence vers un **état stationnaire** (attracteur fixe).

2. Recherche d'un cycle

Pour découvrir un cycle de période T :

6. On enregistre les configurations $\boldsymbol{\omega}(t)$ sur une certaine « fenêtre » de temps (par exemple, sur un buffer circulaire).

7. On compare $\boldsymbol{\omega}(t)$ et $\boldsymbol{\omega}(t+T)$ via une certaine norme (comme $\|\cdot\|$). Si, pour un T donné, on trouve que $\boldsymbol{\omega}(t+T) \approx \boldsymbol{\omega}(t)$ pour plusieurs valeurs de t, on en conclut l'existence d'une **périodicité** T.

Un cas particulier est la **période 2** (un « ping-pong » entre deux états) : on vérifie si

$$\boldsymbol{\omega}(t+2) \approx \boldsymbol{\omega}(t).$$

Dans des systèmes plus complexes, on peut chercher des traces d'**orbites** de période 3 ou plus, ce qui nécessite des tests plus élaborés.

3. Problème de la dimension dans un DSL

Un **SCN** implique $O(n^2)$ liaisons $\omega_{i,j}$ pour n entités, et la dimension de $\boldsymbol{\omega}$ est potentiellement énorme. Pour détecter un attracteur numériquement, on restreint souvent :

- Soit on **échantillonne** un sous-ensemble représentatif de liaisons,
- Soit on considère un **indicateur global** (énergie $\mathcal{J}(\boldsymbol{\omega})$, modularité, etc.) permettant de repérer la stagnation ou la périodicité.
  Ces approches restent des heuristiques, mais elles fonctionnent bien en pratique pour savoir si le système s'est stabilisé (point fixe) ou est entré dans un cycle repérable.

C. Bassins d'attraction : définition et intuition

Le **bassin d'attraction** associé à un attracteur A est l'ensemble de toutes les initialisations $\boldsymbol{\omega}(0)$ dont la trajectoire finit par atteindre (ou approcher) A. Formulé plus formellement,

$$\mathcal{B}(A) \;=\; \{\boldsymbol{\omega}_0 \;\big|\; \lim_{t\to\infty} \mathrm{dist}(\boldsymbol{\omega}(t), A) = 0\},$$

c'est-à-dire que toute trajectoire commençant dans $\mathcal{B}(A)$ aboutira dans A. Quand le DSL présente plusieurs attracteurs (points fixes ou cycles), chacun possède son **bassin** propre, et ces bassins se partagent l'espace initial via des **frontières** potentiellement complexes.

Savoir qu'un attracteur A existe est une chose, mais **savoir** dans quel bassin se trouve l'état initial $\boldsymbol{\omega}(0)$ indique vers **quel** attracteur la dynamique ira. Deux initialisations très voisines peuvent se trouver dans deux bassins différents, entraînant l'aboutissement à deux **configurations** finales radicalement différentes. Cela met en évidence la **sensibilité** aux conditions initiales possible dans un DSL non linéaire.

D. Méthodes pratiques pour étudier bassins et attracteurs

Une façon simple et courante d'**explorer** les bassins d'attraction consiste à lancer de **nombreuses** simulations, chacune avec une initialisation $\boldsymbol{\omega}(0)$ distincte (p. ex. tirée aléatoirement). On fait tourner la dynamique jusqu'à la détection d'un attracteur (point fixe ou cycle). On étiquette alors chaque initialisation par l'attracteur atteint et on essaie de visualiser (en réduction de dimension ou autrement) la zone initiale associée. On obtient ainsi un **maillage** approximatif du paysage des attracteurs.

Dans certains DSL, on dispose d'une **fonction d'énergie** $\mathcal{J}(\boldsymbol{\omega})$ (chap. 2.4) qui, bien qu'elle ne soit pas toujours strictement un potentiel, peut fournir un repère. Si $\mathcal{J}$ descend globalement au fil des

itérations, on remarque où elle se **stabilise** ou **oscille**. Les vallées de $\mathcal{J}$ correspondent en général à des points fixes ou des cycles. On peut alors repérer la "profondeur" de ces vallées et apprécier la **coexistence** d'attracteurs si plusieurs minima existent.

E. Conclusion

Les stratégies pour **détecter** un attracteur (ou un cycle) dans un **SCN** se ramènent à :

- **Observer** la trajectoire $\boldsymbol{\omega}(t)$ et vérifier sa stagnation ou sa périodicité,
- **Explorer** plusieurs initialisations pour connaître les **bassins d'attraction**,
- **Mesurer** la stabilité (jacobienne locale, écart entre $\boldsymbol{\omega}(t)$ et $\boldsymbol{\omega}(t+1)$ ou $\boldsymbol{\omega}(t+T)$, etc.).

L'idée-clé est que la **non-linéarité** d'un DSL ouvre la voie à de multiples attracteurs (points fixes, cycles, régimes complexes) et rend le comportement **dépendant** des conditions initiales et du **flux** (modifications de synergie, insertion d'entités). Cette pluralité d'attracteurs apporte un grand **potentiel** d'adaptation et de diversité, tout en nécessitant une **analyse** méthodique pour comprendre vers où la dynamique d'un SCN se dirige et dans quels **bassins** elle évolue.

4.3.3. Stabilisation vs. Dynamisme Continu

4.3.3.1. Scénario Stationnaire : Synergie Fixe, on Converge vers un Arrangement Stable

Le **Deep Synergy Learning** (DSL) suppose en général que la **synergie** $S(i,j)$ entre entités i et j peut évoluer si la représentation des entités change (embeddings, règles symboliques, etc.) ou si de nouvelles entités apparaissent. Cependant, il existe un *cas plus simple* où la **synergie** $S(i,j)$ reste **constante** au cours du temps : aucun recalcul d'embeddings, pas de nouveau nœud, pas de modification des règles symboliques. On parle alors d'un **scénario stationnaire**, dans lequel la **fonction F** (décrivant l'évolution des pondérations ω) ne dépend que de ω elle-même et non d'autres facteurs changeants. Mathématiquement, on a un **système** :

$$\boldsymbol{\omega}(t+1) = \mathbf{F}\big(\boldsymbol{\omega}(t)\big),$$

où **F** demeure **invariante** dans le temps. Il en résulte qu'on peut étudier la **convergence** vers un état final $\boldsymbol{\omega}^*$ stable, sans se soucier de modifications extrinsèques de la synergie.

A. Équation de Mise à Jour sous Synergie Constante

Lorsqu'on parle de **scénario** stationnaire, la **synergie** $S(i,j)$ ne varie pas avec t. Cela signifie que, pour tout couple d'entités (i,j), on a une valeur **fixe** $S(i,j)$. La **règle** DSL devient alors purement **dynamique** sur ω.

Un schéma particulièrement simple (chap. 4.2.1) est la mise à jour **additive** :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big],$$

dans lequel η est un taux d'apprentissage et τ un coefficient de décroissance (pour éviter la croissance illimitée). Puisque $S(i,j)$ est **constant**, chaque $\omega_{i,j}$ s'ajuste en se rapprochant d'une "valeur-cible".

Si $S(i,j)$ est plus **grand** que $\tau\,\omega_{i,j}(t)$, on ajoute un incrément positif. Inversement, si $\omega_{i,j}(t)$ dépasse $S(i,j)/\tau$, le terme devient négatif.

On peut montrer qu'un **point fixe** se situe en $\omega_{i,j}^{*} = \frac{S(i,j)}{\tau}$. Une fois $\omega_{i,j}$ parvenu à cette valeur, l'update $\omega_{i,j}(t+1) - \omega_{i,j}(t)$ s'annule. Sur le plan **opérationnel**, cela signifie que la **force** de la liaison (i,j) atteint un équilibre où la synergie (qui tend à augmenter $\omega_{i,j}$) compense exactement la décroissance imposée par $\tau\,\omega_{i,j}$.

B. Convergence vers un Arrangement Stable

Dans l'ensemble, la **dynamique** stationnaire produit une **stabilisation** de la matrice $\{\omega_{i,j}\}$. Les liens se rapprochent de valeurs d'équilibre ($S(i,j)/\tau$) ou de 0 (si $S(i,j) = 0$). On parle alors d'un **arrangement stable** : plus rien ne bouge, car la synergie ne change pas et la mise à jour DSL a abouti à un **point fixe**.

Sur chaque liaison $\omega_{i,j}$, si la règle est **additive** :

$$\omega_{i,j}(t+1) - \omega_{i,j}(t) = \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right].$$

Cela se réécrit comme un **processus** de relaxation vers $\frac{S(i,j)}{\tau}$:

8. Tant que $\omega_{i,j}(t) < \frac{S(i,j)}{\tau}$, la différence est positive, $\omega_{i,j}$ **monte**.
9. Si $\omega_{i,j}(t) > \frac{S(i,j)}{\tau}$, la différence est négative, $\omega_{i,j}$ **baisse**.

Pourvu que $\eta\,\tau < 2$ et sans autres termes (ex. inhibition non linéaire), on obtient un **comportement** contractant qui converge.

Une fois **tous** les liens convergés, on dispose d'une **matrice** ω^{*} avec $\omega_{i,j}^{*} \approx \frac{S(i,j)}{\tau}$. Les **entités** i et j pour lesquelles $S(i,j)$ est élevée deviennent fortement liées, celles pour lesquelles $S(i,j) \approx 0$ ont des $\omega_{i,j} \approx 0$. Si l'on applique un **seuil** ou une forme d'inhibition, on peut voir émerger des **clusters** plus marqués.

C. Interprétation en Termes de Clusters

Dans un SCN stationnaire, un jeu de **données** (ou de "synergies" prédéterminées) aboutit à une **partition** implicite. Les liens **forts** à l'équilibre forment des **groupes** d'entités fortement interconnectées.

Si la **matrice** $\{S(i,j)\}$ reflète une structure de type "clusters" (ex. entités proches en embedding), alors la **dynamique** additive fait en sorte que les entités internes à un cluster se retrouvent avec des **liens** $\omega_{i,j}$ élevés. Entre clusters, si $S(i,j)$ est faible, $\omega_{i,j}$ reste bas.

Ce phénomène s'apparente à une forme de **clustering** non supervisé (cf. chap. 2), où le résultat final $\boldsymbol{\omega}^*$ exprime la **partition** latente.

Si la règle de mise à jour est **linéaire**, comme ci-dessus, le **point fixe** est souvent unique et découle aisément de $\frac{S(i,j)}{\tau}$. Mais dès qu'on ajoute un **terme** d'inhibition plus complexe (compétition entre liens) ou des non-linéarités, il peut exister **plusieurs** attracteurs stables (chacun correspondant à un arrangement cluster différent). Dans ce cas, le **SCN** stationnaire peut se figer dans l'un ou l'autre arrangement selon l'initialisation ou des bruits aléatoires (cf. chap. 4.3.2 sur la multiplicité d'attracteurs).

D. Liens Formels avec la Descente d'Énergie

Dans la plupart des **implémentations** DSL, on peut définir une **énergie** $\mathcal{J}(\boldsymbol{\omega})$ (ex. $-\sum_{i,j} \omega_{i,j}\, S(i,j) + \tau/2 \sum_{i,j} \left(\omega_{i,j}\right)^2$) qui, en régime stationnaire, atteint un **minimum** local. Dans ce scénario, la synergie $S(i,j)$ ne changeant pas, la **descente** de $\mathcal{J}$ n'est pas contrariée par des révisions de synergie, et le SCN **aboutit** à un minimum local stable (donc un arrangement final ω^*).

On dit que $\mathcal{J}(\omega)$ est **fixe** car la fonction $\boldsymbol{\omega} \mapsto \mathcal{J}$ ne dépend pas d'autres variables évolutives. La **descente** locale de $\mathcal{J}$ se fait par une dynamique de relaxation, menant au point d'équilibre.

Si on n'ajoute pas de **bruit** ou de **compétition** trop forte, la convergence est généralement assurée (chaque liaison converge vers $S(i,j)/\tau$). En revanche, des schémas multiplicatifs ou inhibiteurs peuvent générer des **oscillations** transitoires avant la stabilisation. Mais in fine, dans un cadre stationnaire, on **s'installe** dans un attracteur.

Conclusion (4.3.3.1)

Dans un **scénario** où la **synergie** $S(i,j)$ demeure **fixe** (aucune mise à jour d'embeddings, aucune apparition de nouveaux nœuds), la **dynamique** DSL (Deep Synergy Learning) se ramène à un **système** autonome sur ω. Cela aboutit typiquement à un **arrangement stable** : chaque liaison $\omega_{i,j}$ converge vers une valeur d'équilibre (ex. $S(i,j)/\tau$ pour le modèle additif), et la **matrice ω** cesse de bouger. On obtient alors une **partition** ou un **réseau** stationnaire mettant en évidence les affinités structurelles dictées par S.

Cette situation *contraste* avec les **cas dynamiques** (voir § 4.3.3.2) où la **synergie** peut se redéfinir au fil du temps : dans le scénario stationnaire, on a une **descente** d'énergie *pure* (ou une relaxation) sans que les entrées S ne perturbent l'équation au fur et à mesure. Le **SCN** se **fige** sur un **"arrangement stable"**, révélant la configuration la plus compatible avec la synergie statique.

4.3.3.2. Scénario Évolutif : synergie modifiée par de nouvelles données / événements, le réseau ne se fige jamais (apprentissage continu)

Dans la section précédente (4.3.3.1), l'analyse d'un **SCN** (Synergistic Connection Network) en **scénario stationnaire** (avec une fonction de synergie $S(i,j)$ fixée) montrait comment le réseau peut converger vers un **arrangement stable**. Cependant, dans bien des cas pratiques, cette hypothèse de synergie *constante* s'avère trop restrictive. En effet, dans un système réel, de nouvelles entités peuvent faire leur apparition, des attributs ou des représentations peuvent évoluer

(embeddings ajustés, règles modifiées), et l'environnement lui-même peut changer. Ainsi, la **fonction** $S(i,j)$ entre deux entités $\mathcal{E}_i$ et $\mathcal{E}_j$ devient **dépendante du temps**, ou se voit "redéfinie" à chaque étape. On se retrouve alors dans un régime d'**apprentissage continu**, où le **réseau** ne parvient pas forcément à une configuration finale unique et peut rester en **réorganisation** perpétuelle.

A. Origines de l'Évolution de la Synergie

Dans un *DSL* fonctionnant en mode "flux" (stream), de nouveaux nœuds $\mathcal{E}_{n+1}$ apparaissent régulièrement : par exemple, de nouveaux documents, de nouveaux utilisateurs, ou encore de nouveaux capteurs. Chaque fois qu'un tel nœud est inséré, on doit :

1. Calculer ou initialiser les liens $\omega_{(n+1),i}$ pour $i = 1, \dots, n$.
2. Mettre à jour ou réévaluer la **synergie** $S(n+1, i)$.

Ces opérations peuvent rompre l'équilibre local déjà en place : en liant $\mathcal{E}_{n+1}$ à divers nœuds, on réoriente éventuellement la structure de **clusters** ou de **communautés** formées jusque-là.

Même si le nombre de nœuds reste le même, la *représentation* d'un nœud peut changer. Par exemple, un *embedding* sub-symbolique $\mathbf{x}_i$ se re-entraîne et prend de nouvelles valeurs, modifiant les **distances** ou **similarités** $\| \mathbf{x}_i - \mathbf{x}_j \|$ ou encore $\mathbf{x}_i \cdot \mathbf{x}_j$. De même, un nœud symbolique peut ajouter ou retrancher une règle (logique), entraînant un **changement** dans la fonction $S_{\text{sym}}(i,j)$. On assiste alors à une **synergie** redéfinie en cours de route, ce qui oblige la mise à jour $\omega_{i,j}$ à se poursuivre en s'adaptant à cette nouvelle donne.

Dans des scénarios temps réel (ex. robotique, environnement capteur), les conditions (luminosité, bruit, configuration spatiale, etc.) varient de façon incessante. La synergie $S(i,j,t)$ — qui peut refléter la cohérence instantanée de deux signaux sensoriels — doit être recalculée en continu, empêchant le SCN de "figer" un arrangement final.

B. Impact sur la Dynamique du Réseau

Contrairement au **cas stationnaire** où l'on peut exhiber un point fixe ω^*, ici la loi

$$\boldsymbol{\omega}(t+1) \;=\; \mathbf{F}\big(\boldsymbol{\omega}(t),\, S(\cdot,\cdot,t)\big)$$

est *non autonome*. On ne peut plus parler de $\boldsymbol{\omega}^* = \mathbf{F}\big(\boldsymbol{\omega}^*, S(\cdot,\cdot)\big)$ de manière permanente, car S varie à chaque étape. En conséquence, le réseau peut évoluer *indéfiniment* :

- Il peut se reconfigurer au gré des insertions de nœuds,
- Il peut réajuster ses liens quand un embedding $\mathbf{x}_i$ change,
- Il peut basculer d'un type de partition à un autre selon l'arrivée de données ou l'évolution d'un paramètre (inhibition, température de recuit, etc.).

Cette logique rejoint l'idée d'un **apprentissage en ligne** : à chaque itération (ou petit batch), on:

1. Observe les changements (nouveaux nœuds, embeddings modifiés…),

2. Recalcule (au moins partiellement) la fonction $S(i,j,t)$,

3. Réalise un *delta update* $\Delta\omega_{i,j}(t)$ pour amener la pondération vers une nouvelle cohérence.

Le **SCN** reste ainsi *flexible* et *vivant*, réorientant ses clusters ou sous-structures au fil du temps.

Du fait que S n'est plus constant, le SCN peut parfois osciller : par exemple, un nœud dont la synergie vis-à-vis d'un cluster se renforce temporairement, puis s'affaiblit, puis se renforce, etc. On peut assister à des "phases" transitoires de stabilisation, suivies d'une "transition" quand une nouvelle *vague* de données survient.

C. Considérations Mathématiques : Systèmes Dynamiques Non Autonomes

Le système s'écrit

$$\boldsymbol{\omega}(t+1) = \mathbf{F}\big(\boldsymbol{\omega}(t),\, \mathbf{p}(t)\big),$$

où $\mathbf{p}(t)$ dénote la configuration (paramètres, représentations) à l'instant t. Les propriétés de stabilité ou de convergence dépendent du *trajectoire* $\{\mathbf{p}(t)\}$. Si $\mathbf{p}(t)$ varie lentement, on peut espérer un "quasi-équilibre" mobile ; si elle varie rapidement, la $\boldsymbol{\omega}(t)$ peut rester en turbulence, sans se poser.

On peut mobiliser des cadres "skew-product" en théorie des systèmes dynamiques non autonomes, ou des approches d'**adaptation** en contrôle. En pratique, la plupart des implémentations reposent sur des *simulations*, où l'on met à jour $\omega_{i,j}(t)$ de façon incrémentale, tout en recomputant $S(\cdot,\cdot,t)$. On observe l'existence ou non d'un "pseudo-ordre", de *phases* où le réseau se stabilise partiellement, puis se réadapte, etc.

Dans un système de recommandation, chaque itération introduit un *nouvel utilisateur* ou un *nouvel item*. La synergie $S(u,i)$ entre l'utilisateur u et l'item i doit être définie (ou recalculée). Cela conduit à des réajustements $\omega_{u,i}$, puis redistribue la force sur d'autres liens déjà existants. Cette dynamique se poursuit indéfiniment, car la base de données utilisateurs/items ne cesse d'évoluer.

D. Applications Pratiques

Recommandation en Ligne.
On ne peut jamais finaliser un "training" une fois pour toutes, car l'arrivée constante d'utilisateurs ou d'items (ou la modification des préférences) rend la matrice ω obsolète si elle n'est pas ajustée. Le **DSL** apporte une plasticité native : chaque nouveau flux de données modifie la synergie S, le SCN réapprend et réorganise les clusters, tout en restant opérationnel.

Robotique.
Les flux sensoriels évoluent (conditions d'éclairage, types d'objets manipulés), les capteurs ou actions peuvent se paramétrer. Le SCN recalcule la *pertinence* des liens sensorimoteurs. Il n'existe pas d'état final, seulement une suite d'états adaptatifs.

Dialogue.
Un chatbot perfectionne ses embeddings (BERT, GPT) à mesure qu'il ingère de nouvelles conversations. La *synergie textuelle* S n'est donc pas fixe. Le SCN en charge d'associer des thèmes ou des intentions réorganise ses liaisons ω en continu, parfois sans stabilisation.

E. Conclusion : Un Réseau en Mouvement

Dans ce **scénario évolutif**, la fonction de **synergie** $S(i,j)$ se modifie au fil du temps — parce que de *nouvelles entités* arrivent, parce que la *représentation* interne de chaque entité change, ou parce que l'environnement est *non stationnaire*. Il s'ensuit que le **SCN** (Synergistic Connection Network) est perpétuellement en **adaptation** :

- Il ne converge **pas** nécessairement vers un unique arrangement stable,
- Il peut manifester des **transitions** majeures au gré des données,
- Il s'inscrit dans un **apprentissage en ligne**, où l'on actualise ω en continu, révélant à chaque instant la configuration de clusters la plus cohérente avec l'état courant de la synergie.

Cela rend le **DSL** particulièrement adéquat pour des **contextes temps-réel** ou des **systèmes** "vivants" et "dynamiques", où le concept de "fin d'entraînement" n'existe pas. Le *réseau* se perçoit comme un organisme "en mouvement" : toujours prêt à **réorganiser** ses groupements (clusters) selon les changements constatés dans S.

4.3.3.3. Exemples de courbes d'évolution $\omega_{i,j}(t)$ montrant la formation progressive d'un cluster

Pour mieux **visualiser** ce qui se passe durant la **dynamique** d'auto-organisation, il est souvent instructif de tracer au fil du temps l'**évolution** de quelques pondérations $\omega_{i,j}(t)$. Dans un **SCN** (Synergistic Connection Network) à grande échelle (n élevé), on se limite généralement à un sous-ensemble représentatif de liens ou à des **indicateurs** agrégés, mais le principe reste le même : observer comment certains liens $\omega_{i,j}(t)$ passent de valeurs proches de 0 (ou très faibles) à des niveaux élevés (voire saturés), tandis que d'autres se réduisent ou stagnent à des valeurs négligeables.

L'idée est de **montrer** la **formation progressive** d'un **cluster** : au début, les liens sont dispersés ou quasi nuls, puis la dynamique sélectionne quelques couples (i,j) pour se renforcer, révélant ainsi un **groupe** cohérent.

A. Scénario simple à cinq entités

Considérons un exemple **pédagogique** où l'on dispose de cinq entités $\{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4, \mathcal{E}_5\}$. Supposons qu'en **réalité** la **synergie** $S(i,j)$ soit nettement **élevée** entre les paires du **sous-groupe** $\{1,2,3\}$ et plus **faible** pour leurs liens avec $\{4,5\}$. Autrement dit, il existe un "vrai" petit cluster $\{1,2,3\}$, alors que $\{4,5\}$ en est assez distant.

Pour amorcer la simulation, on pose :

$$\omega_{i,j}(0) \approx 0 \quad \text{(avec éventuellement un faible bruit aléatoire).}$$

Cette **condition initiale** illustre l'idée que, avant toute itération, les pondérations $\omega_{i,j}$ sont de niveau très bas (ou nuls) ; la dynamique du **DSL** va alors opérer progressivement la mise en place de liens plus ou moins forts.

Lorsque l'on **applique** la règle de mise à jour (par exemple la variante additive) :

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\left[S(i,j) \;-\; \tau\,\omega_{i,j}(t)\right],$$

les paires $(1,2)$, $(1,3)$ et $(2,3)$ commencent à s'**amplifier** si $S(1,2)$, $S(1,3)$, et $S(2,3)$ sont élevés. En pratique, dès les premières itérations, on observe une **croissance** notable de $\omega_{1,2}(t)$, $\omega_{1,3}(t)$, $\omega_{2,3}(t)$. La dynamique va rapidement les rapprocher de leur "point d'équilibre" $\frac{S(i,j)}{\tau}$ (hypothèse sans inhibition).

En revanche, pour les paires mixtes (c'est-à-dire entre $\{1,2,3\}$ et $\{4,5\}$), la valeur de la synergie $S(i,j)$ est plus faible. De ce fait, les mises à jour successives:

$$\Delta\omega_{i,j}(t) \;=\; \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right]$$

restent proches de zéro, voire négatives si $\omega_{i,j}$ a eu un léger sursaut de bruit initial. Les liens $(1,4)$, $(2,5)$, etc., finissent donc par **stagner** à un niveau très faible ou décroître pour s'approcher de zéro.

Pour **visualiser** cette progression, on peut tracer, à chaque itération t, la valeur de $\omega_{i,j}(t)$ pour diverses paires (i,j). Un exemple :

- $\omega_{1,2}(t)$: part d'environ 0 au temps $t = 0$, **monte** assez vite dans les dix ou vingt premières itérations, puis **se stabilise** dans une région proche de $S(1,2)/\tau$.
- $\omega_{1,4}(t)$: demeure très faible ; si le bruit initial la poussait légèrement, la correction $\eta\left[S(1,4) - \tau\,\omega_{1,4}(t)\right]$ devient vite négative ou quasi nulle, la ramenant vers zéro.

Cette différence de **trajectoire** entre paires intra-$\{1,2,3\}$ et paires hors-cluster (par ex. $\{1,4\}, \{2,5\}$) révèle le mécanisme **sélectif** : la synergie forte enclenche un renforcement local des liens, la synergie faible induit leur quasi-extinction.

En examinant les **graphes** $\omega_{i,j}(t)$ en fonction du temps, on distingue alors deux "familles" de courbes :

- **Famille haute** : $\omega_{1,2}(t), \omega_{1,3}(t), \omega_{2,3}(t)$. Ces liens croissent puis se maintiennent à un plateau **élevé**, typique de la **cohésion** interne au cluster $\{1,2,3\}$.
- **Famille basse** : $\omega_{1,4}(t), \omega_{2,5}(t), \omega_{3,4}(t)$, etc. Celles-ci restent très proches de **zéro** ou se tassent rapidement, montrant la **séparation** d'avec $\{4,5\}$.

Après un certain nombre d'itérations, on obtient une **matrice ω** où $\{1,2,3\}$ s'affiche comme un "**micro-bloc**" densément interconnecté, tandis que $\{4,5\}$ n'y est lié que par des poids très bas (ou à un autre cluster si $\{4,5\}$ se rapprochent entre eux).

Lorsque la **dynamique** se stabilise (ou atteint un état stationnaire), on a, pour les paires intra-{1,2,3}, des valeurs $\omega_{i,j}^{*} \approx \frac{S(i,j)}{\tau}$ (si le modèle est purement additif, sans inhibition). Entre {1,2,3} et {4,5}, on constate $\omega \approx 0$.

Dans un **SCN** plus vaste, on peut observer la **même** logique se reproduire sur des groupes plus grands : si la synergie est suffisante, ils émergent comme clusters plus ou moins compacts, et la visualisation des pondérations confirme l'allure "en blocs".

Cet **exemple** simplifié illustre la **progression** d'un DSL qui part d'une initialization "faible" $\omega(0) \approx 0$:

- Les **liens** correspondant à de **fortes** synergies se **renforcent**,
- Les autres se **réduisent** ou restent proches de zéro,
- Au bout d'un nombre d'**itérations**, on **observe** clairement la formation d'un ou plusieurs **clusters** (ici, {1,2,3}).

C'est en examinant les **courbes** $\omega_{i,j}(t)$ qu'on **saisit** la nature dynamique de la convergence : chaque liaison évolue selon la différence $\left[S(i,j) - \tau\,\omega_{i,j}(t)\right]$. Ce **principe** se généralise à de plus grands ensembles, avec des patterns plus complexes, mais la mécanique reste la même : fort $S(i,j)$ ⇒ renforcement, faible S ⇒ liaison quasi éteinte, aboutissant à une **auto-organisation** en groupes cohérents.

B. Exemples de variations en cas d'inhibition ou de multiplicatif

Dans le cadre du **DSL** (Deep Synergy Learning), l'évolution des pondérations $\omega_{i,j}(t)$ peut prendre des profils temporels sensiblement différents, notamment lorsqu'on introduit des **mécanismes** tels que l'**inhibition compétitive** ou une **mise à jour multiplicative**. Il s'ensuit des **dynamiques** qui peuvent produire des comportements parfois plus contrastés, allant de "va-et-vient" marqués à de véritables "sauts" exponentiels.

Lorsque l'on ajoute de l'**inhibition latérale** (voir la section 4.2.2.2) dans la règle de mise à jour, on introduit un **terme** $-\gamma \sum_{k\neq j} \omega_{i,k}\,(t)$ qui pénalise la croissance conjointe de plusieurs liens partant de la même entité $\mathcal{E}_i$. Mathématiquement, la mise à jour $\omega_{i,j}(t+1)$ peut alors s'écrire :

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\left[S(i,j) \;-\; \tau\,\omega_{i,j}(t) \;-\; \gamma \sum_{k\neq j} \omega_{i,k}\,(t)\right],$$

où $\gamma > 0$ représente l'intensité de l'inhibition.

Sur le plan dynamique, si un lien $\omega_{i,2}$ se met à croître rapidement (parce que la synergie $S(i,2)$ est forte), il "inhibe" ou freine la croissance d'autres liens partant de la même entité $\mathcal{E}_i$. Les courbes $\omega_{i,j}(t)$ peuvent alors révéler un **comportement compétitif** : tandis que $\omega_{i,2}$ se renforce, $\omega_{i,3}$ ou $\omega_{i,5}$ stagnent ou diminuent, même si leur synergie $S(i,3)$ ou $S(i,5)$ n'est pas complètement négligeable.

Il est possible d'observer des **va-et-vient** : par moment, $\omega_{i,2}(t)$ progresse, puis si la synergie $S(i,3)$ n'est pas trop faible, $\omega_{i,3}(t)$ profite d'un relâchement d'inhibition pour se renforcer à son tour, ce qui à nouveau rabaisse $\omega_{i,2}$, etc. Ces oscillations (ou successions de prises d'avantage) sont plus ou moins marquées selon les valeurs de γ, η et τ.

Sur un graphe "$\omega_{i,j}(t)$ vs. t", on discerne clairement des courbes qui croissent, puis s'infléchissent sous l'action de l'inhibition, favorisant l'émergence d'un "gagnant" local. Cette situation peut conduire à des clusters plus nets (car un nœud $\mathcal{E}_i$ n'entretient qu'un faible nombre de liens forts), mais aussi à des **phases de compétition** prolongées.

La **règle multiplicative** (voir la section 4.2.2.1) modifie la façon dont un lien $\omega_{i,j}$ se met à jour, en introduisant une **proportionnalité** à sa valeur actuelle. Au lieu d'ajouter un terme $\eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big]$ (cas additif), on applique souvent une formule de la forme :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) \;\times\; \big(1 + \eta\, S(i,j)\big) \;-\; \beta\,\omega_{i,j}(t),$$

ou même une exponentiation directe (par exemple $\omega_{i,j}(t+1) = \omega_{i,j}(t)\exp\{\alpha\big[S(i,j) - \tau\,\omega_{i,j}(t)\big]\}$).

Une caractéristique majeure de la mise à jour multiplicative tient à sa **dépendance** au niveau actuel de $\omega_{i,j}$. Si un lien est déjà modérément fort, la rétroaction positive agit de manière accélérée ("the rich get richer"). Inversement, si $\omega_{i,j}$ est initialement très faible, il peut rester collé à presque zéro, car l'incrément dépend de sa valeur courante.

Les simulations montrent souvent des **sauts exponentiels** : une liaison $\omega_{i,j}$ peut demeurer quasi nulle pendant plusieurs itérations, puis, dès qu'un léger bruit ou une fluctuation l'élève au-dessus d'un petit **seuil**, elle monte subitement (de façon exponentielle) jusqu'à un plateau ou une saturation $\omega_{\max}$. Sur un graphe "$\omega_{i,j}(t)$ vs. t", la courbe paraît stable proche de 0, puis effectue un **bond** rapide.

Ce type de mise à jour produit des **clusters** très tranchés : une fois un lien enclenché (parce qu'un nœud a franchi le seuil de synergie), il grandit fortement, reléguant d'autres liaisons concurrentes à un niveau bas. Dans un DSL, on peut donc aboutir à des partitions plus brutales où quelques liens dominent massivement, tandis que d'autres restent quasi inexistants.

Imaginons que $\omega_{1,2}(t)$ dépasse 0.02 à l'itération $t = 10$ (à cause d'une petite synergie favorable). La multiplicativité enclenche alors une **croissance** en quelques pas, menant $\omega_{1,2}$ à un niveau élevé (0.7, 0.8…). Dans le même temps, $\omega_{1,3}$ ou $\omega_{2,4}$ ne franchissent pas leur seuil, de sorte qu'ils demeurent sous 0.01. Le cluster "(1,2)" se forme ainsi quasi instantanément, tandis que (1,3) reste "ignoré".

Conclusion générale.

Les **variations** introduites par l'**inhibition** ou la **mise à jour multiplicative** façonnent de manière singulière les **courbes** $\omega_{i,j}(t)$. L'inhibition compétitive génère souvent des **dynamiques** de compétition plus ou moins oscillantes, favorisant la formation de liens "gagnants" et la suppression d'autres liaisons sortantes. La multiplicativité, quant à elle, exacerbe le phénomène de **seuil** et d'auto-amplification, conduisant parfois à des "sauts" soudains dans la courbe d'évolution d'un

lien. Dans un **DSL**, ces mécanismes rendent la clusterisation plus marquée (voire brutale) et peuvent mener à des partitions plus segmentées, mais aussi à des **conflits** ou "résonances" selon les paramètres η, β, γ. En pratique, la visualisation des trajectoires $\omega_{i,j}(t)$ aide à **comprendre** le cheminement dynamique vers la formation (ou la disparition) de chaque **lien fort**.

C. Mise en forme de ces courbes

Lorsqu'on souhaite **visualiser** l'évolution des pondérations $\omega_{i,j}(t)$ au cours des itérations, on peut procéder en plusieurs étapes afin de tirer le meilleur parti des **courbes** tracées. Les sous-sections suivantes décrivent ce processus et mettent en évidence son intérêt pour **comprendre** la dynamique d'**auto-organisation** du réseau.

Un **premier** aspect consiste à **stocker** les valeurs de chaque liaison $\omega_{i,j}$ à chaque itération $t = 0,1,\dots,T$. Dans la pratique, on pourra :

- Définir un tableau ou un dictionnaire pour les $\omega_{i,j}(t)$.
- Après chaque mise à jour, on mémorise $\omega_{i,j}(t)$ pour un petit nombre de paires (i,j) d'intérêt (pour ne pas trop alourdir la mémoire si le réseau est grand).
- À l'issue des itérations (par exemple au bout de 50, 100 ou 500 pas), on dispose alors de la **trajectoire** complète $\omega_{i,j}(0), \omega_{i,j}(1), \dots, \omega_{i,j}(T)$.

Pour **sélectionner** les paires (i,j) à tracer, on identifie généralement :

- **Une paire** à forte synergie a priori (ou bien dont on anticipe qu'elle va se renforcer si les entités sont très similaires).
- **Une paire** à synergie moyenne (où l'on veut vérifier si le lien monte ou descend).
- **Une paire** à synergie faible (pour constater si $\omega_{i,j}$ demeure au voisinage de 0 ou s'éteint).

On peut ainsi **comparer** les comportements de quelques liaisons représentatives, sans tracer la totalité des $O(n^2)$ courbes du SCN.

Après avoir récupéré ces **trajectoires** $\omega_{i,j}(t)$, un usage courant est de produire un **graphique** (par exemple via matplotlib ou un outil similaire en Python) où l'axe horizontal représente l'itération t et l'axe vertical la valeur de $\omega_{i,j}(t)$.

On peut tracer **plusieurs** liaisons sur un **même** diagramme :

$$\omega_{1,2}(t), \quad \omega_{1,3}(t), \quad \omega_{4,5}(t), \quad \dots$$

Ces différentes **courbes** de couleur distincte illustrent alors, sur une même figure, comment certains liens **montent** rapidement quand la synergie est forte (et donc la mise à jour DSL les renforce), tandis que d'autres **restent** très bas ou **progressent** lentement.

Cette **représentation** multi-lignes met en évidence :

- Les **phases** d'augmentation : par exemple, une courbe qui s'élève dès les 5 premières itérations, suggérant une forte cohésion entre les entités correspondantes.

- Les **paliers** ou saturations : il arrive qu'après un certain temps, une courbe se fixe autour d'une valeur (ex. $\omega_{1,2}(t) \to 0.8$), marquant la stabilisation d'un lien fort.

- Les **variations plus erratiques** ou "yoyo" : elles indiquent soit des oscillations (chap. 4.3.2.2), soit un changement de la fonction synergie (scénario évolutif, chap. 4.3.3.2).

En examinant ces **trajectoires** de $\omega_{i,j}(t)$, on obtient une **indication** très claire de la situation dynamique :

- Si, au bout d'un nombre raisonnable d'itérations (par exemple 50 ou 100), les **valeurs** $\omega_{i,j}(t)$ montrent toutes un **plateau** ou une zone presque stationnaire, c'est un signe fort de **convergence** (scénario stationnaire du chap. 4.3.3.1). On peut alors considérer que le **SCN** a "appris" sa structure finale.

- À l'inverse, si certaines courbes **oscillent** (montées et descentes successives) ou **changent** de palier à plusieurs reprises, cela indique soit un **phénomène oscillatoire** au sens du chap. 4.3.2.2, soit un **scénario évolutif** (chap. 4.3.3.2) où la synergie S se modifie (arrivée de nouvelles entités, rafraîchissement des embeddings, etc.), empêchant un état unique de s'installer définitivement.

Cette **distinction** (plateau vs. oscillations/progression continue) s'avère cruciale pour **comprendre** le régime dans lequel le **SCN** se situe :

- **Régime stationnaire** : la topologie des liens se "fige" et on peut alors extraire de manière stable les clusters.

- **Régime non stationnaire** : les courbes ne se stabilisent pas, reflétant un **réseau** "vivant" ou "en apprentissage continu", potentiellement réactif aux nouveaux flux de données.

Un **bénéfice** direct de tracer ces courbes est d'observer, "pas à pas", **comment** un cluster se **construit** dans le SCN.

Si l'on voit **plusieurs** liaisons $\omega_{i,j}(t)$ associées à un **même** groupe d'entités $\mathcal{C}$ (par exemple $\{1,2,3\}$) **augmenter** quasi simultanément, c'est la **marque** qu'un bloc se **renforce**. Dans le même temps, les liens de ces entités vers l'extérieur peuvent **stagner** ou **décroître**.

Cela illustre de manière **graphique** le phénomène d'**isolement** d'un cluster : $\omega_{1,2}(t), \omega_{1,3}(t)$ deviennent grandes, alors que $\omega_{1,k}$ pour $k \notin \{2,3\}$ restent faibles. La dynamique DSL favorise les "intra-liens" cohérents et punit (ou néglige) les "extra-liens" divergents.

Exemple d'évolution

- **Étapes initiales** : $\omega_{i,j}(0) \approx 0$. Pendant quelques itérations, toutes les courbes sont proches de zéro.

- **Montée** : pour les paires à haute synergie, leurs courbes **décollent** plus ou moins vite (selon η, τ) et peuvent atteindre une zone de plateau intermédiaire (ex. 0.5–0.7).

- **Stabilisation** : si rien ne change, on voit un palier final (par ex. 0.8) pour ces liens forts, alors que les liens "faibles" conservent une valeur modeste (0.1 ou <0.05).

- **Cluster** : on interprète $\omega_{1,2}, \omega_{1,3}, \omega_{2,3}$ comme hautes, signe d'un **cluster** $\{1,2,3\}$. Les liaisons de ces entités vers d'autres indices $\{4,5,\dots\}$ ne dépassant pas 0.15, on obtient un "mur" séparant $\{1,2,3\}$ du reste.

Selon la **présence** ou non d'inhibition (compétition pour les liaisons sortantes), quelques **tweaks** peuvent survenir :

- *Inhibition latérale* : force un **choix** plus exclusif, où un lien se stabilise un peu au détriment d'un autre. $\omega_{1,3}$ peut donc baisser de 0.5 à 0.3 si $\omega_{1,2}$ grimpe à 0.8, etc.

- *Saturation ou clipping* : empêche une courbe de dépasser $\omega_{\max}$. Visuellement, on voit la courbe buter sur une valeur-limite.

Au final, la **lecture** de ces tracés $\omega_{i,j}(t)$ donne une "**histoire**" de la construction d'un cluster, au lieu de juste observer l'état final $\omega_{i,j}(T)$.

Conclusion

La **mise en forme** des courbes $\omega_{i,j}(t)$ à travers un graphique (multi-lignes) est un **outil** particulièrement instructif pour **visualiser** :

- **La discrimination** progressive des liens : les paires à forte synergie se détachent, tandis que celles à synergie faible restent ou redescendent vers zéro.

- **Le scénario** de convergence : si on voit que les valeurs aboutissent à des plateaux stables, on conclut à un régime stationnaire.

- **Les signes** d'un régime non stationnaire ou oscillant : si les courbes ne parviennent pas à un plateau, on suspecte un **scénario évolutif** ou des **oscillations** (selon la topologie, l'inhibition, la variation des synergies…).

Cette **analyse** offre en outre un **témoignage** direct de l'émergence d'un **cluster** : on voit concrètement *quand* et *comment* les liaisons associées à un groupe s'élèvent en parallèle, donnant une **compréhension** dynamique du processus d'**auto-organisation** au sein du SCN.

4.4. Oscillations, Pseudo-Chaos et Méthodes de Contrôle

Les sections précédentes (4.3) ont montré que la **dynamique** d'un **SCN** (Synergistic Connection Network) pouvait, dans certains cas, conduire à une **stabilisation** (formation de clusters fixes), tandis que dans d'autres, elle pouvait demeurer en mouvement (scénario évolutif). Mais il existe une autre possibilité : le réseau entre dans des **oscillations** plus ou moins régulières ou même un régime "pseudo-chaotique". Dans ce chapitre 4.4, nous allons :

- **Analyser** les **causes** de ces oscillations ou de comportements complexes (4.4.1),
- **Proposer** des **stratégies** pour les **rompre** ou les **contrôler** (4.4.2) lorsque le but est de favoriser la convergence,
- **Présenter** enfin quelques **cas stochastiques** (4.4.3) où l'injection de bruit (recuit simulé, perturbations) peut aider le SCN à échapper à des minima ou à se stabiliser de façon plus globale.

Ce thème s'inscrit dans la continuité de la question des **attracteurs** (4.3.2) : outre des points fixes ou des cycles simples, on peut rencontrer des régimes de type "**oscillations** prolongées" ou "**pseudo-chaos**" si la rétroaction est assez puissante et la non-linéarité suffisamment marquée.

4.4.1. Pourquoi des Oscillations ou un Pseudo-Chaos ?

Malgré la tendance naturelle à la stabilisation (lorsqu'un terme $-\tau\,\omega_{i,j}$ freine la croissance), il peut arriver que la **mise à jour** des pondérations $\omega_{i,j}$ entretienne un **cycle** ou un **mouvement** quasi cyclique. De plus, dans des réseaux de forte dimension et dotés de mécanismes non linéaires, on observe parfois des régimes encore plus complexes, assimilables à du **pseudo-chaos** (une sensibilité marquée aux conditions initiales, une trajectoire ne se répétant pas exactement mais restant dans un comportement "erratique").

4.4.1.1. Rétroactions Positives Trop Intenses, Absence d'Inhibition (2.3.2.2)

Principe Général : un Effet de Boule de Neige

Dans les **formulations** de la mise à jour (qu'elles soient additives ou multiplicatives, voir chapitre 4.2), il est fréquent de disposer d'un terme du type $+\eta\,S(i,j)$ qui **encourage** la croissance du lien $\omega_{i,j}$ dès lors que la **synergie** $S(i,j)$ est élevée. Une fois que ce renforcement positif l'emporte sur la décroissance $-\eta\,\tau\,\omega_{i,j}$ (parce que τ est trop faible ou parce que $S(i,j)$ dépend déjà de $\omega_{i,j}$ de manière auto-amplificatrice), on peut observer un **effet boule de neige** :

$$\omega_{i,j}(t+1) \;>\; \omega_{i,j}(t) \;>\; \ldots \quad \text{pendant plusieurs itérations.}$$

Autrement dit, chaque itération accentue encore plus la croissance d'un lien déjà en essor, de sorte que $\omega_{i,j}$ s'élève presque sans limite. Cette **dynamique** peut conduire à un **emballement** des pondérations et, par ricochet, à l'émergence de **cycles** ou d'**oscillations**, surtout si d'autres liens liés à $\omega_{i,j}$ se trouvent également encouragés à croître.

Absence d'Inhibition et Instabilités

Lorsqu'on n'a **pas** incorporé de mécanismes d'inhibition compétitive (chap. 4.2.2.2) ou de saturation dans la mise à jour, plusieurs liens $\omega_{i,j}$ peuvent simultanément connaître cette croissance soutenue. On se retrouve alors avec un **blocage** : la dynamique part dans un renforcement collectif, où chaque lien élevé en entraîne un autre, modifiant la synergie d'autres paires, etc. Le réseau peut **basculer** successivement d'un groupement à un autre (un cluster domine, puis un autre s'impose), s'inscrivant dans un **cycle** de transitions ou de résonances.

Dans un **système** purement linéaire, la présence d'un **gain** trop important sur les boucles de rétroaction positive suscite des phénomènes de **résonance** ou de divergence. Dans un **système** non linéaire, on peut voir apparaître des attracteurs cycliques ou chaotiques, témoignant d'un équilibre impossible à stabiliser.

Exemple de Rétroaction Positive

Considérons un **mini-système** à deux pondérations ω_1 et ω_2. On suppose :

$$\omega_1(t+1) = \omega_1(t) + \eta\left[S_1\big(\omega_2(t)\big)\tau\,\omega_1(t)\right], \quad \omega_2(t+1) = \omega_2(t) + \eta\left[S_2\big(\omega_1(t)\big)\tau\,\omega_2(t)\right].$$

Si $S_1(\omega_2)$ et $S_2(\omega_1)$ sont des **fonctions croissantes** (par exemple, $S_1(\omega_2) = a\,\omega_2$ avec a grand), alors la croissance de ω_2 **stimule** ω_1, et la croissance de ω_1 **stimule** ω_2. Sans **inhibition**, ce couplage conduit aisément à un **cycle** (un lien monte pendant que l'autre descend, puis vice versa) ou à une **explosion** de leurs valeurs. Dans d'autres cas, on obtient des oscillations autour d'un point ou d'une courbe, et ces oscillations peuvent ne jamais s'amortir si τ est trop faible ou si $\eta\,S_1, S_2$ sont trop grands.

Ce **phénomène** explique l'**emballement** de certaines pondérations dans le réseau si rien n'est là pour les contenir (inhibition, saturation, etc.). Une fois enclenché, la rétroaction positive se nourrit d'elle-même et empêche le système de revenir vers un état stable.

Analyse Mathématique d'Exemple Simplifié

Prenons deux pondérations ω_1, ω_2 avec mise à jour :

$$\begin{cases}\omega_1(t+1) = \omega_1(t) + \eta\,[A\,\omega_2(t) - \tau\,\omega_1(t)],\\ \omega_2(t+1) = \omega_2(t) + \eta\,[B\,\omega_1(t) - \tau\,\omega_2(t)].\end{cases}$$

Ici, $A, B > 0$ modélisent la rétroaction positive, et $\tau\,\omega_i(t)$ la décroissance. Les points fixes ω_1^*, ω_2^* vérifient :

$$\begin{cases}A\,\omega_2^* - \tau\,\omega_1^* = 0,\\ B\,\omega_1^* - \tau\,\omega_2^* = 0,\end{cases}$$

d'où $\omega_1^*(A\,B - \tau) = 0$. On voit qu'il existe un **point fixe non trivial** seulement si $A\,B = \tau$. La stabilité dépendra de la jacobienne :

$$J = \begin{pmatrix}1 - \eta\,\tau & \eta\,A\\ \eta\,B & 1 - \eta\,\tau\end{pmatrix}.$$

Si les **valeurs propres** de J sortent du cercle unité, ω_1, ω_2 peuvent **osciller** ou diverger, engendrant cycles ou chaos.

Dans un **SCN** à grande dimension, ce mécanisme s'étend et des **oscillations** complexes ou pseudo-chaotiques peuvent s'installer si la rétroaction positive est prépondérante ou si aucune forme d'inhibition globale ne vient canaliser la croissance.

Conclusion Partielle : Nécessité d'un Frein ou d'une Inhibition

Lorsque la **rétroaction positive** est **excessive**, le SCN risque de développer des **oscillations** ou un **emballement** où certains liens se rehaussent indéfiniment, souvent en boucle avec d'autres liens croissants. Pour nombre d'applications, c'est un **problème** (on désire la stabilisation de clusters), d'où l'introduction d'un **terme** d'inhibition compétitive (chap. 4.2.2.2) ou de **saturation** (limiter $\omega_{i,j} \leq \omega_{\max}$).

À l'inverse, dans certaines approches exploratoires, on peut **vouloir** conserver un degré de mouvement oscillatoire afin d'explorer diverses partitions temporaires. Mais la plupart du temps, si le but est d'obtenir un **arrangement stable**, il est crucial de **contenir** la rétroaction positive (régler η, τ de sorte que $\eta\,\tau \ll / 1$) ou d'implémenter un mécanisme d'inhibition. Dans la section 4.4.2, on exposera des stratégies pour **endiguer** ces oscillations et viser la **stabilité** désirée.

4.4.1.2. Couplages non linéaires (ex. synergie dépendant des ω elles-mêmes)

Lorsqu'on conçoit un **DSL** (Deep Synergy Learning) de base, on postule souvent que la **synergie** $S(i,j)$ entre deux entités i et j découle de leurs **représentations** internes ($\mathbf{x}_i$ pour le sub-symbolique, R_i pour le symbolique, etc.) sans dépendre de la **valeur courante** du lien $\omega_{i,j}$. Toutefois, dès qu'on autorise un **couplage** où $S(i,j)$ devient **fonction** (directe ou indirecte) de la configuration $\{\omega_{p,q}\}$, on introduit une boucle de **rétroaction** supplémentaire. Cela peut, d'une part, **enrichir** la capacité d'auto-organisation et, d'autre part, **augmenter** le risque d'oscillations ou de comportements pseudo-chaotiques.

A. Principe : $S(i,j)$ dépend de ω

Dans le schéma de mise à jour standard,

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\,\big[S(i,j) - \tau\,\omega_{i,j}(t)\big],$$

on considère $S(i,j)$ comme **exogène**, c'est-à-dire qu'elle ne varie pas selon la valeur de $\omega_{i,j}$. Mais si, au contraire, $S(i,j)$ dépend de $\boldsymbol{\omega}(t)$, alors on obtient :

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\,\big[S\big(i,j,\boldsymbol{\omega}(t)\big) \;-\; \tau\,\omega_{i,j}(t)\big].$$

La **dépendance** $S\big(i,j,\boldsymbol{\omega}(t)\big)$ peut être **directe** (S explicitement lié à $\omega_{i,j}$) ou **indirecte** (ex. un module symbolique, activé seulement si certains liens sont déjà au-dessus d'un seuil, modifie la synergie).

On peut imaginer une **synergie** de la forme :

$$S(i,j) \;=\; S_0(i,j) \;+\; \alpha \sum_{k \neq i,j} \omega_{i,k}\; \omega_{k,j}.$$

Cela signifie que plus un *tiers* k connecte déjà i et j, plus la "coopération" potentielle entre i et j grandit. C'est un **effet** d'"auto-renforcement" de type : "Si i et k sont liés, et k et j sont liés, alors la synergie i, j augmente."

Mathématiquement, le **DSL** prend alors la forme d'un système **non linéaire** :

$$\boldsymbol{\omega}(t+1) \;=\; \boldsymbol{\omega}(t) \;+\; \eta\,\big[\mathbf{S}\big(\boldsymbol{\omega}(t)\big) - \tau\,\boldsymbol{\omega}(t)\big].$$

La **variable ω** agit sur $S(\cdot,\cdot)$, qui *en retour* modifie **ω**. Sur le plan **dynamique**, cette boucle interne peut créer des régimes oscillatoires ou amplifiés si rien ne vient limiter cette rétroaction.

B. Conséquences Mathématiques : Non-linéarités et Risques d'Emballement

Dans un DSL où un **lien** $\omega_{i,j}$ peut *lui-même* accroître $S(i,j)$, la "correction" $\Delta\omega_{i,j} \approx \eta\,[S(i,j) - \tau\,\omega_{i,j}]$ risque de devenir **positive** au-delà d'un simple seuil, augmentant **exponentiellement** $\omega_{i,j}$. Sans un mécanisme de **saturation** ou d'**inhibition** (voir 4.4.2), $\omega_{i,j}$ peut monter sans bound ou osciller de façon importante.

Prenons un cas simplifié où :

$$S(i,j,\boldsymbol{\omega}) \;=\; S_0(i,j) \;+\; \alpha\,\omega_{i,j}.$$

Alors la mise à jour additive devient :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\,\big[S_0(i,j) + \alpha\,\omega_{i,j}(t) - \tau\,\omega_{i,j}(t)\big].$$

On obtient :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\,S_0(i,j) + \eta\,(\alpha - \tau)\,\omega_{i,j}(t).$$

- Si $\alpha > \tau$, alors $\alpha - \tau > 0$, et l'on peut avoir $\eta\,(\alpha - \tau) > 1$, créant un **terme** d'amplification qui fait diverger $\omega_{i,j}$.
- À l'inverse, si $\eta\,(\tau - \alpha) > 1$, des oscillations ou un comportement erratique peut se produire selon le signe et la dynamique pas-à-pas.

Ainsi, de **petits** changements de paramétrage peuvent entraîner de grands effets.

Dans un **réseau** de grande dimension, des couplages non linéaires peuvent conduire à des **dynamiques** pseudo-chaotiques, dans lesquelles un **petit** écart initial sur $\boldsymbol{\omega}(0)$ se traduit, au bout de plusieurs itérations, par de grandes différences (sensibilité exponentielle aux conditions initiales). Sur un plan théorique, ce type de comportement exige de solides **dispositifs** (inhibition, limites de croissance, etc.) pour canaliser la rétroaction positive et préserver une trajectoire "stable" ou quasi-stable.

C. Exemples d'Applications

Même si les couplages non linéaires augmentent le risque de comportements complexes, ils **intéressent** certains scénarios :

Réseaux Sociaux ou Viraux

Un lien "viral" entre i et j renforce la probabilité que d'autres liens autour s'activent, ce qui, en retour, augmente la "popularité" de $\omega_{i,j}$.

Sur un plan **DSL**, cela se traduit par $S(i,j,\omega)$ croissant avec $\omega_{i,j}$. Sans mécanismes d'inhibition, on peut voir l'**emballement** d'un cluster "viralisé."

Logique Symbolique Contexte-Dépendante

Certains **règles** ne s'appliquent que si un lien existant $\omega_{i,j}$ dépasse un certain **seuil**. Cela modifie **localement** la valeur de $S(i,j)$ quand la configuration $\{\omega_{k,\ell}\}$ franchit ce seuil.

Sur le plan **dynamique**, on crée des **transitions** d'état quand un lien se met à excéder ledit seuil, ce qui peut enclencher une série de modifications en cascade.

Collaboration Multi-Agent

Dans un essaim robotique, la "**coopération**" entre deux robots peut se trouver *amplifiée* si un troisième robot opère de manière synergique avec chacun d'eux. Cela se modélise par un **terme** $\sum_k \omega_{i,k}\ \omega_{k,j}$ venant augmenter $S(i,j)$.

Risque : **boucles** de renforcement où le trio $\{i,j,k\}$ devient instable ou quasi-oscillatoire.

D. Comment Contrôler ou Réguler ces Couplages ?

Pour éviter des **oscillations** ou une **explosion** des liens, on recourt souvent à :

- **Inhibition** (chap. 4.2.2.2) : Un coût $-\gamma \sum_k \omega_{i,k}$ empêche qu'un nœud i accroisse tous ses liens en même temps.
- **Saturation** ou "clipping" : On borne $\omega_{i,j} \leq \omega_{\max}$.
- **Seuil** adaptatif : On coupe toute $\omega_{i,j} < \theta$.

Ces **outils** limitent la rétroaction positive et bloquent la dérive exponentielle.

On peut analyser la **stabilité** via une étude du **système** dynamique :

$$\omega_{i,j}(t+1) - \omega_{i,j}(t) = \eta\left[S\big(i,j,\boldsymbol{\omega}(t)\big) - \tau\,\omega_{i,j}(t)\right].$$

En dérivant les **conditions** sous lesquelles $\nabla_\omega S(\cdot)$ est dominée par le terme $-\tau\,\mathbf{I}$, on obtient des **critères** (genre $\eta\ \|\ \nabla S\ \| < \tau$) assurant un équilibre stable.

Dans certains cas, on *veut* cette rétroaction positive pour encourager un **cluster** à s'auto-former vigoureusement (ex. pour modéliser des "communautés fortes"). On l'associe à un frein global

(inhibition, competition). Le résultat : l'émergence de **petits** clusters internes très soudés, plutôt que de liens dispersés.

E. Conclusion

Dès lors que la **synergie** $S(i,j)$ dépend de $\omega_{i,j}$ — ou plus globalement, de la configuration **ω** — on bascule dans un **système** dynamique potentiellement **non linéaire** à rétroaction :

Enrichissement : la **cohérence contextuelle** se modèle plus finement (ex. plus un cluster grandit, plus sa synergie interne grandit).

Risque : cette rétroaction positive peut engendrer **oscillations**, **pseudo-chaos**, ou **instabilité**.

Pour y faire face, on introduit des **régulations** (inhibition, saturation, etc.) et on contrôle les **paramètres** de la mise à jour (taux η, coefficient τ) afin de **contenir** la rétroaction. Ainsi, on confère au DSL la possibilité d'exprimer des phénomènes émergents plus riches, tout en préservant une **stabilité** ou une **convergence** raisonnable.

4.4.2. Stratégies pour Rompre les Oscillations

Dans la section 4.4.1, nous avons décrit **pourquoi** des oscillations ou un pseudo-chaos pouvaient apparaître dans un **SCN** (Synergistic Connection Network) : rétroactions positives trop intenses, absence d'inhibition, couplages non linéaires, etc. Pour de nombreuses applications du **DSL** (Deep Synergy Learning), on souhaite cependant **stabiliser** la dynamique, car des oscillations prolongées ou un chaos déterministe peuvent rendre la structure (clusters, attracteurs) peu exploitable.

Heureusement, il existe plusieurs **méthodes** pour **rompre** ou **atténuer** ces oscillations, en introduisant notamment :

- De l'**inhibition latérale** (4.4.2.1),
- Des **règles** de **sparsification** (4.4.2.2),
- Une **adaptation** fine des paramètres η, τ (4.4.2.3).

Ces stratégies visent à **contenir** l'emballement de certains liens et à maintenir la rétroaction dans un domaine maîtrisé. Elles s'appliquent que la mise à jour soit **additive** ou **multiplicative**, et peuvent parfois se cumuler pour un meilleur effet stabilisateur.

4.4.2.1. Inhibition Latérale (2.4.4) : contrainte de ressources, pousse le réseau à choisir un nombre limité de liens forts

L'**inhibition latérale** – parfois appelée **inhibition compétitive** – vise à contrôler la **croissance simultanée** de multiples liens sortant d'une même entité $\mathcal{E}_i$. Au lieu de laisser $\omega_{i,j}$ s'élever librement lorsque la synergie $S(i,j)$ est satisfaisante, on y ajoute un **terme négatif** qui dépend de la **somme** (ou d'un autre agrégat) des connexions $\omega_{i,k}$ sortant du même nœud i. Une **forme** typique de cette mise à jour a été discutée en (4.2.2.2) et peut se formaliser, pour une règle additive, de la façon suivante :

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\left[S(i,j) \;-\; \tau\,\omega_{i,j}(t)\right] \;-\; \gamma \sum_{k\neq j} \omega_{i,k}(t),$$

où $\gamma > 0$ est un **coefficient** d'inhibition, et $\sum_{k\neq j} \omega_{i,k}\,(t)$ désigne la **masse** totale des autres liens sortants de $\mathcal{E}_i$.

A. Principe et influence sur les oscillations

Lorsque $\mathcal{E}_i$ dispose d'un **budget** de ressources limité (biologiquement, on parle d'énergie neuronale, dans un réseau technique de bande passante, etc.), l'inhibition latérale impose une **compétition interne** : si $\omega_{i,j}$ tente de s'élever, l'effet $-\gamma \sum_{k\neq j} \omega_{i,k}$ contraint la croissance simultanée d'autres liens. Sur le plan **comportemental**, cela force $\mathcal{E}_i$ à "**choisir**" un petit nombre de connexions privilégiées, évitant qu'il ne se connecte trop fortement à de multiples nœuds en parallèle.

Dans le DSL sans inhibition, un **lien** $\omega_{i,j}$ peut parfois s'amplifier fortement dès que la synergie $S(i,j)$ est supérieure à la "penalty" $\tau\,\omega_{i,j}$, en particulier s'il existe des boucles positives. L'inhibition latérale introduit un **terme négatif** supplémentaire proportionnel à la somme $\sum_{k\neq j}$ $\omega_{i,k}$, ce qui limite la capacité de $\omega_{i,j}$ à croître de manière exponentielle. Ce **frein** empêche également un ensemble de liens de s'amplifier **simultanément**, l'un d'eux finissant par "dominer" les autres.

De ce fait, toute **oscillation** qui apparaîtrait parce que plusieurs liens tenteraient d'atteindre de grandes valeurs à tour de rôle se trouve en partie **amortie** par la compétition : plus un lien s'élève, plus il pèse sur l'ensemble des autres et inversement. On observe ainsi, mathématiquement, une **dissipation** de l'énergie potentielle d'oscillation.

B. Analyse mathématique simplifiée

Pour voir comment l'inhibition latérale agit sur la **stabilité**, considérons la règle :

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right] \;-\; \gamma \sum_{k\neq j} \omega_{i,k}(t).$$

Un **point fixe** $\omega^*_{i,j}$ s'obtient lorsque $\omega_{i,j}(t+1) = \omega_{i,j}(t) = \omega^*_{i,j}$, i.e. :

$$S(i,j) \;-\; \tau\,\omega^*_{i,j} \;-\; \gamma \sum_{k\neq j} \omega^*_{i,k} \;=\; 0.$$

Sur l'ensemble des entités, cela se traduit par un système d'équations linéairement couplées (ou, selon la forme de S, non linéaires), imposant une **compétition** pour qu'un nœud $\mathcal{E}_i$ ne puisse "allouer" trop de force à plusieurs liens concurrents.

Sur le plan **stabilité**, la matrice jacobienne **J** de la transformation $\omega_{i,j} \mapsto \omega_{i,j} + \cdots$ inclut des **termes croisés** liés au $\sum_{k\neq j} \omega_{i,k}$. En un point fixe, ces dérivées croisées se traduisent par un **effet** de concurrence : quand un lien se renforce, il pèse sur les autres, ce qui rend plus difficile un emballement synchronisé. On diminue ainsi la probabilité de boucles positives incontrôlées, et,

dans la majorité des cas pratiques, on aboutit à un **régime** stable ou faiblement oscillatoire (des oscillations amorties).

C. Impact sur la formation de clusters

L'inhibition latérale incite chaque entité $\mathcal{E}_i$ à ne conserver qu'un **nombre restreint** de liaisons fortes. Cela se rapproche d'un **modèle** "winner-takes-most" : si $\omega_{i,j}$ est très élevé pour un certain j, les autres $\omega_{i,k}$ $(k \neq j)$ subissent une pénalisation accrue. On obtient donc un **réseau économe** en connexions, renforçant la **lisibilité** des structures émergentes (éviter que tout soit moyennement connecté).

Dans les algorithmes de clustering, on recherche souvent des **groupes** ("communautés") clairement définis : des entités fortement reliées **entre elles** et faiblement reliées **aux autres**. L'inhibition latérale contribue à ce "clivage" : un nœud $\mathcal{E}_i$ ayant rejoint un cluster principal est moins susceptible de développer des liens concurrentiels vers un autre cluster, car cette croissance nouvelle se heurterait au coût $\gamma \sum_{k \neq j} \omega_{i,k}$. Cela **stabilise** les partitions émergentes.

D. Conclusion : un mécanisme crucial de stabilité

L'**inhibition latérale**, inspirée des phénomènes **biologiques** (2.4.4), remplit deux rôles essentiels :

Rôle de stabilisation :

En limitant l'**auto-amplification** de multiples liens, on **réduit** le risque d'oscillations ou de comportements chaotiques. La dynamique DSL reste **contrôlée** et converge souvent plus sûrement.

Rôle de structuration :

Chaque nœud se **focalise** sur quelques liens dominants, produisant des **clusters** plus nets et plus faciles à interpréter. La compétition incite le réseau à "faire des choix" tranchés, plutôt qu'à disperser ses connexions.

Dans les sections suivantes (4.4.2.2, 4.4.2.3) et plus loin (chap. 7.4), on verra comment compléter cette inhibition latérale par d'autres techniques (seuil adaptatif, saturation, recuit simulé) pour **approfondir** la maîtrise de la dynamique et **éviter** les fluctuations excessives. L'inhibition demeure toutefois l'un des **leviers** centraux pour maintenir un **SCN** (Synergistic Connection Network) dans un état **auto-organisé**, cohérent et **parcimonieux**.

4.4.2.2. Sparsification : imposer un maximum de k liaisons par entité (k plus forts liens), ε-radius, etc.

La **sparsification** vise à **réduire** le nombre de liaisons $\omega_{i,j}$ actives dans un **SCN** (Synergistic Connection Network), afin de **limiter** la complexité des rétroactions susceptibles de créer des oscillations ou d'autres comportements chaotiques. L'idée est qu'en empêchant chaque nœud $\mathcal{E}_i$ d'entretenir de multiples liens à intensité moyenne, on obtient un **réseau** plus clairsemé (sparse),

ce qui **stabilise** la dynamique et **clarifie** la formation de clusters. Les méthodes les plus courantes sont :

k-liaison : on ne garde, pour chaque entité i, que les k plus forts liens $\omega_{i,j}$.

ε-**seuil** : on annule tout lien $\omega_{i,j}$ dont la valeur tombe sous un seuil ε.

Dans un cas, on contrôle le **degré** sortant du nœud i (exactement k liens), dans l'autre, on impose un **niveau** minimal pour conserver une liaison.

A. Principe général de la sparsification

En l'absence de **restriction**, un nœud $\mathcal{E}_i$ pourrait entretenir des connexions non négligeables vers la quasi-totalité des autres nœuds. Dans les réseaux de grande dimension, cela crée un très grand nombre de **circuits de rétroaction** : le surplus de liens "moyens" accroît la **probabilité** d'oscillations ou de comportements complexes.

La **sparsification** rompt cette possibilité : elle force chaque nœud à "**faire des choix**", c'est-à-dire à ne conserver qu'un **sous-ensemble** restreint de connexions jugées essentielles. En conséquence, de nombreux liens, dont les valeurs sont trop faibles ou moins utiles, sont **coupés** (fixés à 0). On obtient alors un **réseau** moins densément connecté, limitant mécaniquement la complexité dynamique et réduisant le risque d'oscillation.

B. Méthodes pratiques

Après chaque cycle de mise à jour $\omega_{i,j}(t+1)$ ..., on "trie" les liaisons sortantes de chaque entité $\mathcal{E}_i$ par ordre décroissant (ou selon un critère de pertinence). Seuls les k premiers liens $\omega_{i,j}$ (ceux de plus grande intensité) sont **préservés**. Les autres, classés au-delà du rang k, sont mis à 0.

$$\omega_{i,j}(t+1) \;=\; \begin{cases} \omega_{i,j}(t+1), & \text{si } \omega_{i,j} \text{ est dans le top } k, \\ 0, & \text{sinon.} \end{cases}$$

Cette **stratégie** garantit que le **degré** (sortant) de $\mathcal{E}_i$ ne dépasse pas k. On parle parfois de "k-sparsification". Cette forme de contrainte a pour effet de **réduire** fortement le nombre de liens actifs si $k \ll n$. Cela **simplifie** la dynamique : chaque nœud ne s'occupe que d'un petit set de voisins jugés les plus forts.

Ici, on fixe un $\varepsilon > 0$. Après la mise à jour, si $\omega_{i,j} < \varepsilon$, alors on **coupe** le lien $\omega_{i,j}$ en le mettant à 0 :

$$\omega_{i,j}(t+1) \;=\; \begin{cases} \omega_{i,j}(t+1), & \text{si } \omega_{i,j}(t+1) \;\geq\; \varepsilon, \\ 0, & \text{sinon.} \end{cases}$$

On obtient un "ε-radius" : seuls les liens **au-dessus** de ε subsistent. Le **degré** de chaque nœud n'est pas contrôlé directement (cela peut fluctuer), mais on coupe efficacement tout petit lien demeuré trop faible, évitant la prolifération de connexions moyennes. Cette technique se rapproche du "hard thresholding" (voir 7.4.3) : on impose qu'en deçà d'un certain seuil, le lien n'a plus de raison d'exister.

On peut :

- **Appliquer** la coupe (k-liaison ou ε-seuil) à chaque itération (version stricte),
- Ou **périodiquement** (tous les p cycles, ou quand la densité du réseau dépasse un certain seuil).

De plus, on peut **faire varier** k ou ε au fil du temps : autoriser d'abord un plus grand nombre de liens, puis "resserrer" la contrainte. Cela s'apparente à un recuit "inhibiteur", où la parcimonie augmente à mesure que le réseau s'organise.

C. Impact sur la limitation des oscillations

Lorsqu'un **grand** nombre de liens coexistent dans un SCN, la **rétroaction** non linéaire peut entraîner des **oscillations** localisées ou globales (voir 4.3.2). En imposant un k-liaison ou un ε-seuil, on **casse** de nombreux arcs, réduisant ainsi drastiquement les chemins de rétroaction possibles. Moins de boucles = moins d'opportunités d'osciller.

Par ailleurs, chaque nœud n'a plus la "capacité" de faire croître simultanément un grand nombre de liens, car la plupart seront coupés s'ils sont trop faibles. La **dynamique** se concentre alors sur un ensemble restreint d'arcs — souvent, ceci rend la **trajectoire** du réseau plus lisible et moins sujette à des allers-retours.

D. Exemples d'usage et bénéfices

- **Clustering** plus franc : avec peu de liaisons conservées, les **clusters** deviennent plus évidents, car on voit nettement qui se connecte à qui.
- **Calcul plus rapide** : dans un réseau volumineux (n grand), manipuler une matrice ω dense peut être coûteux ($O(n^2)$ liens). Si, pour chaque i, on ne garde que k liens, on tombe à $O(n\,k)$, fréquemment $O(n)$ si k est constant. Cela accélère la mise à jour, en plus de calmer la dynamique.
- **Contrôle** de la densité : si le réseau tend à trop se connecter (souvent constaté dans l'entraînement DSL), la sparsification agit comme un "filet" coupant la majorité des petites connexions.

E. Conclusion : un atout majeur pour calmer le SCN

La **sparsification** se présente comme un **outil** essentiel, complémentaire à l'**inhibition latérale** (4.4.2.1) pour :

- **Réduire** le risque d'oscillations ou de comportements chaotiques,
- **Structurer** un SCN en clusters lisibles (éviter la dispersion de liaisons moyennes),
- **Abaisser** le coût computationnel, surtout pour de grandes n.

Dans un **DSL**, imposer "k-liaison", "ε-seuil" ou toute autre forme de coupe (périodique ou continue) permet, au besoin, d'assurer la **stabilité** de la dynamique. Les sections suivantes (4.4.2.3, etc.) détailleront également la façon d'ajuster η et τ pour achever la **réduction** des phénomènes oscillatoires et améliorer la convergence vers un **arrangement** stable du SCN.

4.4.2.3. Adaptation de η, τ : **régulation fine** de la **vitesse** de mise à jour et de la **décroissance**.

La **dynamique** d'un **SCN** (Synergistic Connection Network) repose en grande partie sur deux **paramètres** fondamentaux : le **taux d'apprentissage** η et le **taux de décroissance** τ. Ces paramètres apparaissent dans l'équation de mise à jour (chap. 4.2) :

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\,\big[S(i,j) \;-\; \tau\,\omega_{i,j}(t)\big],$$

ou, dans une version multiplicative, sous une forme proche de

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \times [1 + \eta(\cdots)] \;-\; \ldots\,.$$

D'un point de vue **mathématique**, η contrôle la **vitesse** à laquelle chaque lien $\omega_{i,j}$ se met à jour ; τ détermine la **force** de "rappel vers zéro", autrement dit la **décroissance** qui empêche le lien de grandir sans limite. Un mauvais choix de η ou de τ peut générer des **oscillations**, voire un **emballement** ; à l'inverse, des valeurs trop "timides" peuvent **freiner** exagérément la formation des clusters.

Dans les sections précédentes (4.4.2.1, 4.4.2.2), nous avons décrit l'**inhibition latérale** et la **sparsification** comme des solutions pour **endiguer** les oscillations. Ici, nous montrons qu'un **réglage** précis de η et τ est un levier tout aussi crucial, souvent complémentaire à ces mécanismes.

A. Rôle de η : vitesse de mise à jour

Dans le cas **additif** :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\,\big[S(i,j) - \tau\,\omega_{i,j}(t)\big],$$

l'analyse (4.2.1.2) révèle que pour garantir une convergence locale (sans oscillations), il faut

$$|1 - \eta\,\tau| < 1,$$

ou au moins $\eta\,\tau < 2$. Au-delà de cette borne, le système peut entrer en **oscillation** et, pour $\eta\,\tau$ encore plus grand, **diverger**.

Lorsque le SCN comprend **plusieurs** entités et couplages, l'analyse de stabilité se raffine par l'étude de la **matrice jacobienne** globale. Néanmoins, l'idée essentielle subsiste : un η trop fort accélère énormément la correction, risquant de "sauter" par-dessus la valeur d'équilibre, créant des "rebonds" successifs (oscillations). En revanche, un η trop faible retarde la mise en place des **clusters**, rendant la convergence (ou la pseudo-convergence) plus lente.

À l'instar de nombreuses méthodes d'apprentissage, on peut **décroître** η au fil des itérations (un "annealing" du learning rate). On commence avec un η suffisamment élevé pour **différencier** rapidement les liens (ceux qui ont une synergie élevée se renforcent plus vite), puis on réduit η afin de **stabiliser** la configuration et éviter que les fluctuations ne perdurent.

B. Rôle de τ : force de la décroissance

Le paramètre τ agit comme un **frein** : à chaque mise à jour,

$$\eta\,\big[S(i,j) - \tau\,\omega_{i,j}(t)\big]$$

inclut la composante $-\eta\,\tau\,\omega_{i,j}(t)$, qui ramène $\omega_{i,j}$ vers zéro si la synergie $S(i,j)$ n'est pas assez grande. Si τ est trop **faible**, les liens risquent de croître exagérément ; si τ est trop **fort**, la plupart des liens restent proches de zéro, le SCN échouant à **exploiter** la synergie potentielle.

- $\tau \approx 0.1$: peut être trop faible si, par exemple, $S(i,j) \approx 1$. Dans ce cas, la décroissance (multipliée par $\omega_{i,j}$) n'est pas assez forte pour empêcher un emballement (surtout si η n'est pas minime).
- $\tau \approx 10$: écrase presque toute **croissance** des liens, car $\tau\,\omega_{i,j}(t)$ dépasse rapidement $S(i,j)$. Les liaisons finissent par stagner très près de zéro.

Comme pour η, on peut **faire évoluer** τ : commencer avec un τ modéré (laissant le réseau former de premiers clusters), puis accroître τ pour consolider la configuration en bloquant les variations tardives. Dans un SCN évolutif, un module de surveillance détecte d'éventuelles oscillations : si elles sont trop fortes, on **relève** τ pour "amortir" davantage la dynamique.

C. Lien avec l'inhibition et la sparsification

Ces mécanismes se **superposent**. Avec **inhibition latérale** (4.4.2.1), la mise à jour inclut un terme

$$-\gamma \sum_{k \neq j} \omega_{i,k}(t),$$

qui limite aussi la croissance simultanée de plusieurs liens. Cela peut autoriser un η un peu plus élevé ou un τ plus bas, puisque la compétition prend en charge une partie de la **stabilisation**.

Si l'on **coupe** régulièrement les liens trop faibles (ε-seuil) ou ne garde que les k plus gros liens sortants, la dimension effective du système diminue (moins de boucles de rétroaction). Par conséquent, le couple (η, τ) peut être choisi un peu plus librement. Dans un réseau très dense, il faut souvent "serrer" η et τ de manière prudente.

D. Approche algorithmique pour η, τ : essais, annealing, jacobienne

Essais empiriques : La méthode la plus fréquente : on **tente** plusieurs paires (η, τ), on **observe** le SCN. S'il y a trop d'oscillations, on réduit η ou on monte τ. Si, au contraire, la formation de clusters est trop lente ou partielle, on augmente η ou on diminue τ. Pour des **grands** SCN, on pratique souvent un calibrage sur un **sous-problème** réduit.

Annealing : On **démarre** avec un η modérément élevé pour accélérer la **différenciation** des liaisons, puis on le **baisse** de façon linéaire ou exponentielle quand le réseau commence à se structurer. On peut également monter τ progressivement pour fortifier l'effet de décroissance au fur et à mesure que la configuration se stabilise.

Étude de la jacobienne : Sur un système jouet (quelques liens), on peut analyser la **matrice J** $= (\partial \Delta\omega / \partial\omega)$. On vise $\| \mathbf{J} \| < 1$ ou un spectre(**J**) en module < 1. Cela donne des **conditions** plus formelles sur η, τ. Dans des SCN réels (grande échelle), cette étude devient trop complexe, mais la simulation permet de **surveiller** l'évolution de ω et le maintien d'une certaine stabilité.

E. Conclusion

Le **couple** η, τ est un **pivot** de la dynamique DSL :

- η détermine la **vitesse** de réaction des liaisons $\omega_{i,j}$. Trop grand, on risque des oscillations ; trop petit, on s'enlise dans une convergence lente ou imparfaite.
- τ fixe la **décroissance** imposée à chaque lien, empêchant un emballement illimité si elle est bien choisie, ou brisant la croissance si elle est excessive.

Combiné à l'**inhibition latérale** (4.4.2.1) et la **sparsification** (4.4.2.2), un réglage adapté de η et τ :

- **Amortit** les oscillations,
- **Canalise** la dynamique pour qu'elle converge (ou quasi-converge) vers un état de clusters stables,
- **Équilibre** la rapidité d'adaptation et la stabilité globale du SCN.

Cette **régulation** fine de η, τ complète donc les leviers décrits, assurant que la configuration du réseau ne tombe ni dans un chaos oscillatoire ni dans une inertie excessive. On obtient ainsi un SCN plus **souple** et **maîtrisé**, à la fois **réactif** et **stable** face aux variations de $\mathbf{x}_i$, de $S(i,j)$, ou d'autres paramètres internes.

4.4.3. Cas Stochastiques : Recuit, Perturbations

Après avoir présenté (4.4.2) diverses stratégies **déterministes** (inhibition, sparsification, ajustement η, τ) pour rompre ou limiter les oscillations, nous abordons à présent des **approches stochastiques**. L'idée est parfois **d'injecter** du **bruit** (des perturbations aléatoires) dans la dynamique, afin de sortir de configurations sous-optimales ou d'éviter une oscillation stable qui ne nous satisfait pas (chap. 2.4.5.1 abordait déjà ce concept). On retrouve ici l'analogie avec le **recuit simulé** en physique statistique, qui secoue la configuration $\boldsymbol{\omega}(t)$ avant de "refroidir" pour se stabiliser dans un attracteur potentiellement plus global.

4.4.3.1. Injection volontaire de "bruit" pour échapper aux minima (2.4.5.1)

Il existe des situations où le **Synergistic Connection Network** (SCN) se trouve **piégé** dans un **minimum local** ou entretient une **oscillation** persistante qui empêche la dynamique de converger vers une configuration plus satisfaisante. Dans un tel contexte, l'idée d'introduire un **terme stochastique** dans la mise à jour des pondérations peut sembler contradictoire, d'autant plus que l'on cherche souvent à **diminuer** l'**instabilité** ou la **non-linéarité**. Cependant, cette démarche conserve une **logique** solide : de la même manière que l'on applique un **recuit** en métallurgie pour franchir une barrière d'énergie, l'**injection de bruit** agit ici comme un **apport d'énergie** aléatoire qui peut aider le SCN à se libérer d'un attracteur local ou à rompre un **cycle** d'oscillation (cf. section 2.4.5.1).

Une **contradiction** apparente peut se lire dans le fait que l'on introduise délibérément des **perturbations aléatoires**, alors même que des mécanismes tels que l'**inhibition** ou la **sparsification** visent à stabiliser la dynamique. Pour comprendre cette démarche, il faut garder à l'esprit que l'objectif premier est de **favoriser** l'**exploration** de configurations nouvelles, qui seraient difficilement atteignables depuis un état trop stable mais de mauvaise qualité. Les **systèmes physiques** fournissent une analogie : un solide en cours de recuit a besoin d'un **excès** d'énergie pour franchir le **seuil** séparant deux **vallées** de potentiel. Dans le cadre d'un **SCN**, le bruit joue un rôle semblable en **déstabilisant** ponctuellement la **trajectoire** des pondérations $\omega_{i,j}$. On peut alors espérer que la structure glisse vers un autre **bassin d'attraction**, plus proche d'une configuration optimale.

Il est fréquent, dans les **réseaux** adaptatifs et non linéaires, de se trouver en présence d'**oscillations** du type "ping-pong", où quelques liaisons alternent périodiquement entre des valeurs élevées et faibles, sans jamais converger vers un point fixe stable. L'**injection** d'un **terme aléatoire** peut dans ce cas **briser** la synchronisation de l'oscillation. Si l'on modélise ce phénomène, on peut considérer qu'à chaque itération t, la mise à jour

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\left[S(i,j) \;-\; \tau\,\omega_{i,j}(t)\right] \;+\; \xi_{i,j}(t)$$

vient remplacer la formule usuelle $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right]$. Le **terme** $\xi_{i,j}(t)$ est un **bruit** gaussien ou uniforme, en général de moyenne nulle, dont l'**amplitude** peut être fixée ou décroissante au fil du temps. Dans certains cas, on peut aussi cibler seulement un **sous-ensemble** de paires (i,j) pour ne pas perturber l'ensemble du réseau.

Sur le **plan mathématique**, on peut relier cette injection de bruit à la **descente de gradient** perturbée, très souvent employée pour **échapper** aux minima locaux. Bien que le **DSL** et son SCN ne correspondent pas toujours à une **descente de gradient** globale (on parle souvent d'une mise à jour locale ne minimisant pas explicitement une fonction ; voir chapitre 2.4.3 sur la notion d'**énergie**), le **raisonnement** demeure valable : tout **puits** local où le réseau serait "coincé" dispose d'une **barrière** à franchir, et le bruit aléatoire fournit la **quantité** d'énergie nécessaire pour la surmonter. Par analogie, on peut dire que l'**état** du SCN suit un "processus stochastique" à l'intérieur d'un **paysage d'énergie**, et qu'en l'absence d'un tel bruit, l'état risquerait de converger dans la première vallée rencontrée, sans avoir la possibilité d'explorer un autre **bassin** à énergie plus basse.

D'un point de vue **dynamique**, on peut observer que dans des conditions de bruit modéré, la **dérive** aléatoire permet d'**éviter** la cristallisation prématurée d'un **mauvais** attracteur local. Si le bruit est trop élevé, on peut toutefois provoquer un **comportement** erratique qui empêche de **stabiliser** réellement la structure. En pratique, on paramètre la **variance** de $\xi_{i,j}(t)$ ou sa **fenêtre** d'uniformité de façon à autoriser quelques **sauts** hors des vallées, mais sans plonger le réseau dans un mouvement totalement brownien.

En **conclusion**, la **méthode** stochastique qui consiste à **injecter** volontairement du **bruit** dans la dynamique $\omega_{i,j}$ répond à deux motivations : permettre au **SCN** de **sortir** d'un **attracteur** local (ou d'une partition sous-optimale) et **rompre** des **oscillations** cycliques. Elle trouve un **équilibre** en se combinant à d'autres approches comme l'**inhibition**, la **sparsification** ou la **compétition** entre liaisons, décrites dans le cadre du **DSL** (voir chapitre 2.4.5.1). Ainsi, même si l'ajout de perturbations aléatoires est a priori contraire à la recherche de **stabilité**, il apparaît en réalité comme

un **complément** essentiel à la dynamique, pour accroître la **capacité** d'exploration et améliorer le **résultat** final lorsque le paysage d'états regorge de minima locaux ou de cycles indésirables.

4.4.3.2. Recuit simulé : un protocole pour secouer la matrice ω avant de "refroidir" et se stabiliser

Le recours à un **recuit simulé** (\emph{simulated annealing}) permet de structurer plus finement l'**injection de bruit** au sein d'un **Synergistic Connection Network** (SCN). Plutôt que d'ajouter des perturbations stochastiques ponctuelles ou uniformément réparties dans le temps, le **recuit** propose de "chauffer" d'abord le réseau à une **température** élevée pour lui donner la capacité d'**explorer** un vaste éventail de configurations, puis de "refroidir" graduellement, laissant le SCN se **figer** dans un état final qu'on espère "globalement optimal" ou, à tout le moins, **meilleur** qu'un minimum local trivial. Cette approche puise ses racines dans la **physique statistique**, où l'on chauffe un matériau (agitation thermique forte) avant de le refroidir lentement, de sorte à franchir des barrières d'énergie et atteindre une structure cristalline stable.

A. Analogies et principes du recuit.

Le recuit s'inspire de la métallurgie : un matériau chauffé voit ses atomes disposer d'une énergie thermique suffisante pour franchir de nombreuses barrières locales et se réarranger. En le refroidissant graduellement, on stabilise la configuration dans une cristallisation souvent plus homogène et moins sujette aux défauts. Transposé à un **SCN**, le "chauffage" correspond à l'adjonction d'un **terme aléatoire** de grande amplitude à la formule de mise à jour des pondérations $\omega_{i,j}$. Ceci permet au réseau de "sauter" entre plusieurs bassins d'attraction, sans se bloquer prématurément dans l'un d'eux. Le "refroidissement" opère en réduisant progressivement ce **bruit**, faisant tendre la dynamique vers une structure où les liens $\omega_{i,j}$ se figent dans une configuration stable.

Dans la perspective d'une **fonction énergie** $\mathcal{J}(\boldsymbol{\omega})$ (voir la section 2.4.3 sur la vision "descente d'énergie"), un important niveau de bruit favorise les remontées énergétiques transitoires, que l'on peut formaliser par un facteur $\exp\left[-\left(\mathcal{J}(\boldsymbol{\omega}') - \mathcal{J}(\boldsymbol{\omega})\right)/T\right]$. Le réglage d'une **température** T régule le degré de tolérance aux hausses d'énergie : à haute température, on accepte aisément de "monter" en énergie pour fuir un puits local médiocre ; à basse température, on rejette de tels mouvements, ce qui stabilise le SCN.

B. Protocole concret dans un SCN.

Dans une formulation pratique, on pose une **température** T dont la valeur décroit au fil des itérations selon une loi typique telle que

$$T(t) = T_0\, \alpha^{\lfloor t/\Delta t \rfloor}, \quad 0 < \alpha < 1,$$

avec un pas de temps Δt. On ajoute alors, à chaque mise à jour $\omega_{i,j}(t+1)$, un **bruit** $\xi_{i,j}(t)$ dont la **variance** dépend de T. Ainsi, on obtient la relation

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right] + \xi_{i,j}(t),$$

où $\xi_{i,j}(t) \sim \mathcal{N}(0, \sigma(t)^2)$ avec $\sigma(t) \propto T(t)$. Dans les premières itérations, $\sigma(t)$ est grande : le SCN est agité, il "saute" volontiers hors des puits locaux. Puis, à mesure que T baisse, $\sigma(t)$ diminue, ce qui fige progressivement la structure émergente.

C. Analyse mathématique du recuit.

Sur le plan formel, l'on connaît des **résultats** garantissant qu'un refroidissement suffisamment lent (c'est-à-dire un α très proche de 1) peut conduire le système à **converger** vers un minimum global de l'énergie $\mathcal{J}$. En pratique, on arbitre entre la qualité de la solution (un refroidissement plus lent favorise l'exploration) et le temps de calcul (un refroidissement trop lent peut être inabordable). Dans un **SCN**, on n'a pas toujours besoin de suivre l'exact protocole de la physique statistique (Markov Chain Monte Carlo, acceptation/rejet), car un simple **bruit additif** "pondéré" par exp $[-\Delta\mathcal{J}/T]$ peut remplir la fonction d'exploration.

En tout état de cause, le recuit simulé fournit un **cadre** méthodique pour **monter** d'abord la "température", explorer le **paysage** d'états plus librement, puis "refroidir" en limitant graduellement les perturbations.

D. Avantages et inconvénients.

Le recuit simulé présente l'**avantage** majeur de \textbf{décoller} le réseau d'un minimum local (ou d'un attracteur) inopportun. Là où un DSL déterministe, limité à

$$\omega_{i,j}(t+1) = \omega_{i,j}(t)\eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right],$$

risquerait de s'arrêter prématurément, la "secousse" apportée par un niveau élevé de bruit autorise l'exploration d'autres **partitions** ou **clusters** potentiels. Cette dimension d'exploration peut aussi, dans les phases "haute température", dévoiler plusieurs organisations concurrentes, dont l'une s'avérera plus globalement satisfaisante.

Ce **bénéfice** se paie par un **coût** temporel accru. D'une part, on doit maintenir des itérations "bruyantes" en nombre suffisant pour vraiment sauter entre bassins d'attraction ; d'autre part, il faut prévoir un **refroidissement** progressif, ce qui rallonge la durée de la simulation. Un autre **risque** vient de la configuration initiale du $\sigma(t)$: si la température de départ est trop basse, on ne profitera pas assez du caractère stochastique ; si elle est trop haute, la trajectoire pourra demeurer chaotique ou écarter certaines bonnes configurations faute d'un contrôle approprié.

E. Conclusion.

Le recuit simulé se révèle un protocole pertinent pour "**secouer**" la matrice $\{\omega_{i,j}\}$ avant de "\textbf{refroidir}" la dynamique, cherchant ainsi à **éviter** les minima locaux et à **briser** des oscillations éventuelles. Il complète les approches plus déterministes (inhibition, saturation) qui visent à **stabiliser** la structure. En veillant à la cohérence du rythme de décroissance de la température, on apporte au **Synergistic Connection Network** un instrument supplémentaire pour **explorer** l'espace des configurations, déjouer les verrous locaux et produire, in fine, une **organisation** potentiellement plus satisfaisante à l'échelle du réseau.

4.4.3.3. Choix de la “température” stochastique, liens avec la physique statistique

Lorsqu’un **recuit simulé** ou plus généralement une **injection de bruit** (voir 4.4.3.1 et 4.4.3.2) est implémenté dans un **Synergistic Connection Network** (SCN), la notion de **température** T joue un rôle central. Cette **température**, inspirée des concepts de la **physique statistique**, contrôle l’**ampleur** des perturbations aléatoires ajoutées aux pondérations $\omega_{i,j}$. Un bon réglage de T est donc déterminant pour que le réseau puisse, dans un premier temps, **explorer** suffisamment l’espace des configurations, puis, à l’inverse, se **figer** à la fin du processus, permettant ainsi une convergence vers un état stable.

A. Rappel : analogie avec la physique statistique.

En physique, la **température** T mesure l’**agitation** des atomes ou des particules. Un solide porté à haute température dispose de suffisamment d’**énergie** pour dépasser des barrières locales et se réorganiser librement. Dans le **SCN**, la “température” joue un rôle similaire : lorsqu’elle est élevée, on autorise l’**injection** de perturbations stochastiques de grande amplitude dans la mise à jour des pondérations. Ces perturbations permettent au réseau de “sauter” hors de puits d’énergie trop étroits ou de configurations peu satisfaisantes. À mesure qu’on réduit progressivement T, l’agitation baisse et le SCN se **stabilise**, à l’image d’un refroidissement qui fige les atomes d’un métal dans une structure cristallisée.

Il existe, dans les modèles inspirés de la physique statistique (Ising, Hopfield, etc.), une relation de type $\exp\left[-\left(\mathcal{J}(\omega') - \mathcal{J}(\omega)\right)/T\right]$ régissant la probabilité d’accepter une modification augmentant l’énergie du système. Même si un **DSL** peut s’écarter de ces protocoles Markoviens classiques, la **logique** reste identique : un niveau de température suffisamment haut donne au SCN la capacité de franchir de petites “vallées” d’énergie et d’explorer son espace d’états de manière étendue.

B. Comment définir T et l’amplitude de bruit.

Le principe consiste à associer l’**injection** de bruit à la **température**. Concrètement, si $\xi_{i,j}(t)$ désigne un bruit gaussien de moyenne nulle, sa variance σ^2 peut être reliée à la température via $\sigma = \sigma_0\, T$. Lorsque T est grand, les fluctuations $\xi_{i,j}(t)$ atteignent une amplitude notable ; à l’inverse, pour une température faible, elles deviennent minimes. Dans un **recuit simulé**, la température elle-même suit un schéma de décroissance contrôlé, par exemple

$$T(t) \;=\; T_0\, \alpha^{\lfloor t/\Delta \rfloor},$$

avec $\alpha \in (0,1)$. Ainsi, au démarrage, T_0 est suffisamment élevé pour permettre une exploration étendue, puis, au fil du temps, on abaisse σ et on réduit graduellement le mouvement aléatoire autour de la configuration courante.

Cette stratégie reproduit l’idée que la **dynamique** d’un SCN doit d’abord être **libérée** (phase de haute température) pour échapper à des attracteurs locaux trop précoces, puis **ralentie** (phase de basse température) afin de stabiliser la structure. Dans certains contextes, on maintient même un petit bruit résiduel pour conserver un léger “tremblement” qui peut se révéler bénéfique, par exemple pour empêcher de retomber dans des oscillations cycliques.

C. Sélection de T_0 et vitesse de décroissance.

Le choix de la **température initiale** T_0 a un impact crucial. Si elle est trop basse, la perturbation stochastique est insuffisante pour sortir le réseau de son attracteur initial ; si elle est trop élevée, toute structuration précoce de $\omega_{i,j}$ est détruite, et le SCN passe son temps en réarrangements chaotiques. Il convient, en pratique, d'ajuster T_0 par essais, parfois en s'appuyant sur une estimation du "coût" des barrières typiques.

Le rythme de **refroidissement** (vitesse à laquelle α fait décroître T) détermine la portée de l'exploration. Un refroidissement extrêmement lent peut tendre vers un optimum global, selon les théorèmes classiques du recuit simulé, mais cela s'avère onéreux en temps de calcul. Les approches plus pragmatiques s'accommodent d'une décroissance modérée qui équilibre la qualité de la solution et l'efficacité de la simulation.

Les **critères** d'arrêt (seuil minimal de température, durée maximum d'itérations, stabilité de la matrice ω) contribuent à définir la durée du recuit. Il faut laisser au SCN le temps d'**explorer**, tout en évitant de prolonger trop inutilement une phase de bruit intense qui empirerait le coût de calcul.

D. Liens formels avec la physique statistique.

Dans un **DSL** disposant d'une fonction "énergie" $\mathcal{J}(\boldsymbol{\omega})$, on peut imaginer un protocole de type Metropolis–Hastings, où l'on calcule la différence $\Delta\mathcal{J} = \mathcal{J}(\boldsymbol{\omega}') - \mathcal{J}(\boldsymbol{\omega})$ et on accepte, selon la probabilité $\min\{1, \exp[-\Delta\mathcal{J}/T]\}$, une modification qui augmente l'énergie. Cet artifice modélise la possibilité de s'extraire d'un puits local, contrôlée par la température. Bien que la mise à jour d'un SCN ne soit pas toujours décrite en ces termes, la **logique** demeure : un niveau de T élevé inflige des **sauts** plus fréquents et plus importants, tandis qu'un faible T "gèle" l'état.

En physique, on observe des **transitions** de phase (paramagnétique à ferromagnétique, par exemple). Parallèlement, dans un SCN, un fort bruit peut maintenir un ensemble de liaisons "déstructuré", puis un abaissement de T induit une **cristallisation** sous forme de **clusters** affirmés.

E. Conclusion.

Le **choix** de la "température" T est un **levier essentiel** pour piloter l'équilibre entre l'exploration (grâce à un bruit significatif) et la stabilisation (grâce à un bruit quasi nul). En début d'apprentissage, on privilégie une "phase chaude" (élevée T_0) qui facilite les sauts hors des minima locaux et la rupture d'oscillations non désirées. Puis, en réduisant progressivement $\sigma(t) \propto T(t)$, le réseau se "fige" dans une configuration plus stable, comparable à un métal cristallisé après un recuit lent.

Cette démarche **stochastique**, inspirée de la **physique statistique**, complète les mécanismes plus déterministes (inhibition, parsimonie) déjà déployés pour maîtriser la dynamique du SCN. Le paramétrage judicieux de T_0, de α et du temps de refroidissement offre un **contrôle** fin sur l'ensemble du processus et peut considérablement améliorer le **résultat** final, notamment dans les situations où la fonction énergie $\mathcal{J}$ possède de nombreuses vallées locales ou où le SCN développe des **oscillations** difficiles à résorber autrement.

4.5. Héritage de la Physique Statistique et des Systèmes Dynamiques

4.5.1. Notions de Contraction, Point Fixe, Attracteurs

- 4.5.1.1. Rappels mathématiques : $\| \mathrm{D}F(\boldsymbol{\Omega}^*) \| < 1$, etc.
- 4.5.1.2. Critères de stabilité locale ou globale, cas non linéaires.
- 4.5.1.3. Illustrations : petit SCN (3 ou 4 entités) pour montrer stabilisation exponentielle ou oscillations.

4.5.2. Énergie ou Pseudo-Énergie

- 4.5.2.1. Cas simple : $\mathcal{J}(\boldsymbol{\Omega}) = -\sum \omega_{i,j} S(i,j) + \tau/2 \sum \left(\omega_{i,j}\right)^2$ (2.4.3).
- 4.5.2.2. Descente locale vs. paysage évolutif si $S(i,j)$ change.
- 4.5.2.3. Cas stationnaire : $\Delta\omega_{i,j} = 0 \rightarrow \omega_{i,j}^* = \frac{S(i,j)}{\tau}$.

4.5.3. Renvoi vers Chapitre 2.3 et 2.4

- 4.5.3.1. Bref rappel des sections 2.3.2 (attracteurs multiples), 2.4.3 (notion d'énergie).
- 4.5.3.2. Comment on operationalise ces idées maintenant dans un cadre "appliqué".

4.5. Héritage de la Physique Statistique et des Systèmes Dynamiques

Tout au long des sections précédentes (4.2, 4.3, 4.4), nous avons souligné la **dynamique** d'un **SCN** (Synergistic Connection Network) : comment les pondérations $\omega_{i,j}$ évoluent, comment des clusters se forment ou des oscillations se produisent, et comment on peut les contrôler (inhibition, sparsification, recuit simulé). Derrière ces mécanismes se cachent des **concepts** issus de la **physique statistique** (modèles d'énergie, recuit, transitions de phase) et de la **théorie des systèmes dynamiques** (points fixes, cycles, attracteurs). Cette section (4.5) met en évidence cet **héritage** et explique pourquoi on parle souvent de "descente d'énergie" ou de "contraction" locale pour caractériser la stabilité d'un point fixe.

4.5.1. Notions de Contraction, Point Fixe, Attracteurs

La grande force d'un **SCN** (Deep Synergy Learning) réside dans sa **capacité** à faire émerger des **états** (clusters, arrangements) stables ou quasi stables. Pour analyser ces états, on s'appuie sur des outils mathématiques liés aux **points fixes** et aux **attracteurs** d'une application **F** (le "règlement" qui, à chaque itération, met à jour $\omega_{i,j}(t)$).

4.5.1.1. Rappels mathématiques : $\| \mathrm{D}F(\boldsymbol{\Omega}^*) \| < 1$, etc.

Dans l'analyse des dynamiques d'auto-organisation dans un **Synergistic Connection Network (SCN)**, il est essentiel de disposer d'un cadre mathématique permettant d'étudier la stabilité des

pondérations $\omega_{i,j}$. Ce cadre s'appuie sur la notion d'**équilibre** ou de **point fixe** d'une fonction de mise à jour globale, ainsi que sur la condition de **contraction locale** caractérisée par l'opérateur dérivé de cette fonction. Nous rappelons ici les principaux éléments de ce formalisme.

A. Formalisation de la mise à jour

On considère que la mise à jour globale des pondérations peut être décrite par une fonction

$$\mathbf{F}: \mathcal{W} \to \mathcal{W},$$

où $\mathcal{W}$ désigne l'espace des matrices $\{\omega_{i,j}\}$, souvent assimilé à $\mathbb{R}^{n\times n}$ si l'on compte n entités dans le SCN. Ainsi, à l'itération t, le vecteur (ou la matrice) des pondérations est mis à jour selon

$$\boldsymbol{\omega}(t+1) = \mathbf{F}\big(\boldsymbol{\omega}(t)\big).$$

Un **point fixe** $\boldsymbol{\omega}^*$ est défini par la condition

$$\boldsymbol{\omega}^* = \mathbf{F}(\boldsymbol{\omega}^*),$$

ce qui signifie que si le système atteint cet état, aucune mise à jour ultérieure ne le modifie. Ce concept constitue la base pour étudier la convergence et la stabilité du SCN.

B. Condition de contraction locale

Pour qu'un point fixe $\boldsymbol{\omega}^*$ soit stable, il est nécessaire que la fonction de mise à jour contracte les petites perturbations autour de ce point. En termes mathématiques, on examine la **matrice jacobienne** de $\mathbf{F}$ évaluée en $\boldsymbol{\omega}^*$, notée $\mathrm{D}\mathbf{F}(\boldsymbol{\omega}^*)$. On impose qu'il existe une **norme** $\|\cdot\|$ sur l'espace $\mathcal{W}$ telle que

$$\| \mathrm{D}\mathbf{F}(\boldsymbol{\omega}^*) \| < 1.$$

Cette inégalité indique que toute petite perturbation δ autour de $\boldsymbol{\omega}^*$ est réduite par l'application de $\mathrm{D}\mathbf{F}$, ce qui se traduit par un retour progressif vers le point fixe. La condition de contraction, souvent résumée par $\| \mathrm{D}F(\boldsymbol{\Omega}^*) \| < 1$, est donc le critère de **stabilité locale**.

C. Exemple élémentaire en dimension 1

Considérons un système unidimensionnel où la mise à jour s'exprime par

$$\omega(t+1) = F\big(\omega(t)\big).$$

Un point fixe ω^* satisfait

$$\omega^* = F(\omega^*).$$

La condition de stabilité locale pour ce système s'écrit alors

$$|F'(\omega^*)| < 1,$$

ce qui assure qu'une petite perturbation autour de ω^* sera contractée par l'itération. En dimension supérieure, cette condition se généralise par l'exigence que la norme de la matrice jacobienne $\mathrm{D}\mathbf{F}(\boldsymbol{\omega}^*)$ soit strictement inférieure à 1.

D. ***Lien avec la dynamique additive***

Dans le DSL, la mise à jour des pondérations se fait généralement par une règle additive de la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big],$$

où $S(i,j)$ est la synergie entre les entités $\mathcal{E}_i$ et $\mathcal{E}_j$, η le taux d'apprentissage, et τ le coefficient de décroissance. Pour analyser la stabilité autour du point fixe, nous supposons que $S(i,j)$ est constant et que la dynamique converge vers $\omega^*_{i,j}$. En posant

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) = \omega^*_{i,j},$$

nous avons

$$\omega^*_{i,j} = \omega^*_{i,j} + \eta\big[S(i,j) - \tau\,\omega^*_{i,j}\big].$$

En simplifiant, nous obtenons

$$S(i,j) - \tau\,\omega^*_{i,j} = 0 \quad \Rightarrow \quad \omega^*_{i,j} = \frac{S(i,j)}{\tau}.$$

Pour étudier la stabilité locale, on examine la dérivée locale de la fonction de mise à jour. Dans ce schéma simplifié, la dérivée par rapport à $\omega_{i,j}$ s'exprime par

$$\frac{d}{d\omega_{i,j}}\big[\omega_{i,j} + \eta\big(S(i,j) - \tau\,\omega_{i,j}\big)\big] = 1 - \eta\,\tau.$$

La condition de convergence locale exige que

$$|1 - \eta\,\tau| < 1.$$

Cette inégalité traduit la **contraction** autour du point fixe, c'est-à-dire que toute perturbation sera atténuée au fil des itérations, garantissant la stabilité du système.

Conclusion partielle

Le critère de contraction, symbolisé par l'inégalité

$$\| \mathrm{D}\mathbf{F}(\boldsymbol{\omega}^*) \| < 1,$$

constitue la base pour vérifier la **stabilité** d'un point fixe dans un SCN. Lorsque l'on applique la règle de mise à jour additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big],$$

et que l'on considère un cas stationnaire où $S(i,j)$ est constant, le système converge vers un équilibre donné par

$$\omega^*_{i,j} = \frac{S(i,j)}{\tau}.$$

La dérivée locale $1 - \eta\,\tau$ doit être inférieure à 1 en valeur absolue, ce qui assure que toute perturbation autour de l'équilibre est contractée, garantissant ainsi la stabilité locale du SCN. Cette interprétation en termes de **descente d'énergie** ou de **gradient** offre une vision intuitive de la manière dont le DSL équilibre le renforcement des liens avec la nécessité de réguler leur croissance, évitant ainsi une explosion des pondérations. Bien entendu, dans des cas plus complexes où $S(i,j)$ varie dans le temps, l'analyse se complique, mais le principe fondamental demeure : la stabilité du point fixe repose sur la condition de contraction de la fonction de mise à jour.

4.5.1.2. **Critères de stabilité locale ou globale, cas non linéaires**

Dans l'analyse des systèmes dynamiques qui sous-tendent un **Deep Synergy Learning (DSL)**, la question de la stabilité des points fixes et des attracteurs est centrale. En effet, pour garantir que les pondérations $\omega_{i,j}$ convergent vers des valeurs qui reflètent correctement la **synergie** entre entités, il convient de s'appuyer sur des critères mathématiques issus de la théorie des systèmes dynamiques. Ces critères se distinguent en deux catégories : la **stabilité locale**, qui examine le comportement du système à proximité d'un point fixe, et la **stabilité globale**, qui concerne la convergence de la dynamique depuis presque toute condition initiale. La présence de **non-linéarités** telles que l'inhibition, les couplages multiplicatifs ou les mécanismes de saturation complique l'analyse, mais le principe de base demeure celui de la contraction autour de l'attracteur.

*A. **Stabilité locale : La contraction au voisinage du point fixe***

Pour étudier la stabilité locale d'un point fixe $\mathbf{\Omega}^*$ dans un SCN, on considère la fonction de mise à jour globale

$$\mathbf{F}: \mathcal{W} \to \mathcal{W},$$

définie sur l'espace $\mathcal{W}$ des matrices de pondérations. À chaque itération, le vecteur de pondérations se met à jour selon

$$\boldsymbol{\omega}(t+1) = \mathbf{F}\big(\boldsymbol{\omega}(t)\big).$$

Un point fixe $\mathbf{\Omega}^*$ satisfait alors la condition

$$\mathbf{\Omega}^* = \mathbf{F}(\mathbf{\Omega}^*).$$

La **stabilité locale** se caractérise par la capacité du système à ramener toute perturbation infinitésimale autour de $\mathbf{\Omega}^*$ vers ce point fixe. Pour cela, on linéarise $\mathbf{F}$ autour de $\mathbf{\Omega}^*$ en introduisant la **matrice jacobienne** $\mathrm{D}\mathbf{F}(\mathbf{\Omega}^*)$. La condition de contraction est alors exprimée par

$$\| \mathrm{D}\mathbf{F}(\mathbf{\Omega}^*) \| < 1,$$

où $\|\cdot\|$ désigne une norme appropriée (par exemple, la norme spectrale) et où le rayon spectral $\rho\big(\mathrm{D}\mathbf{F}(\mathbf{\Omega}^*)\big)$ est inférieur à 1. En d'autres termes, toute perturbation δ autour du point fixe satisfait, pour une itération :

$$\boldsymbol{\omega}(t+1) - \mathbf{\Omega}^* \approx \mathrm{D}\mathbf{F}(\mathbf{\Omega}^*)(\boldsymbol{\omega}(t) - \mathbf{\Omega}^*),$$

et la norme de l'écart se contracte à chaque itération, garantissant que

$$\lim_{t\to\infty} \| \boldsymbol{\omega}(t) - \boldsymbol{\Omega}^* \| = 0.$$

Cette condition de contraction, analogue à la condition en dimension 1 $|F'(x^*)| < 1$, assure que dans un voisinage de $\boldsymbol{\Omega}^*$, la dynamique est **attractive** et les petites perturbations sont rapidement annulées par le processus itératif.

B. Stabilité globale : Convergence depuis presque toute condition initiale

La **stabilité globale** implique que, quel que soit l'état initial $\boldsymbol{\omega}(0)$ (à l'exception d'un ensemble de mesure nulle), la trajectoire converge vers le point fixe $\boldsymbol{\Omega}^*$, c'est-à-dire

$$\lim_{t\to\infty} \| \boldsymbol{\omega}(t) - \boldsymbol{\Omega}^* \| = 0.$$

Pour démontrer la stabilité globale, il faut que la fonction de mise à jour **F** soit **contractante** sur l'ensemble de l'espace $\mathcal{W}$, et non seulement dans un voisinage de $\boldsymbol{\Omega}^*$. Cette condition est beaucoup plus forte et rarement vérifiée dans des systèmes non linéaires à haute dimension, où l'on observe souvent la coexistence de plusieurs attracteurs ou de **bassins d'attraction** distincts. Dans ces cas, le système présente une **multi-stabilité** et la convergence dépend fortement de la condition initiale.

C. Cas non linéaires et phénomènes complexes

Les systèmes réels de DSL intègrent fréquemment des **non-linéarités** telles que des mécanismes d'inhibition, des couplages multiplicatifs ou des saturations. Par exemple, une mise à jour du type multiplicatif peut être exprimée par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) \exp\big(\eta \left[S(i,j) - \tau\, \omega_{i,j}(t)\right]\big),$$

ou encore des termes d'inhibition peuvent être ajoutés pour moduler la dynamique en fonction des interactions globales, donnant lieu à une mise à jour plus complexe du type

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\, \omega_{i,j}(t)\big] - \gamma \sum_{k\neq j} \omega_{i,k}\,(t).$$

Ces non-linéarités rendent l'analyse de la stabilité plus subtile. La **matrice jacobienne** de la fonction de mise à jour devient alors un opérateur qui intègre des termes croisés et dont les valeurs propres peuvent être complexes ou dépasser 1 en module, ce qui conduit à des phénomènes tels que les **oscillations**, les **bifurcations** ou même le **chaos**. Dans ces régimes, le système ne converge pas nécessairement vers un point fixe stable, mais peut osciller ou présenter des attracteurs étranges. Les méthodes de **contrôle** telles que l'inhibition, la **sparsification** ou l'ajustement dynamique des paramètres η et τ sont alors nécessaires pour maintenir une dynamique raisonnablement stable.

D. Conclusion

Les **critères de stabilité**, qu'ils soient locaux ou globaux, reposent sur l'analyse de la **contraction** de la fonction de mise à jour par le biais de la matrice jacobienne, avec la condition fondamentale

$$\| \mathrm{D}\mathbf{F}(\boldsymbol{\Omega}^*) \| < 1.$$

En situation stationnaire, cette condition se traduit par la convergence vers un point fixe $\boldsymbol{\Omega}^*$ où, dans le cas d'une règle additive simple,

$$\omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

Lorsque la dynamique est **non linéaire** – en raison de mécanismes d'inhibition, de couplages multiplicatifs ou d'autres sources de non-linéarité – l'analyse devient plus complexe, et la stabilité locale peut être assurée dans un voisinage de l'attracteur, tandis que la stabilité globale n'est garantie que sous des hypothèses fortes de contraction sur tout l'espace. En pratique, ces critères, ainsi que des phénomènes comme les oscillations et les bifurcations, illustrent l'héritage des **systèmes dynamiques** et de la **physique statistique** dans l'étude du DSL. Cette approche permet non seulement de garantir la convergence locale des pondérations, mais aussi d'adapter le système en présence de perturbations, en utilisant des techniques de contrôle et de régulation pour préserver la **robustesse** et la **cohérence** de l'auto-organisation du SCN.

4.5.1.3. **Illustrations : Petit SCN (3 ou 4 entités) pour montrer stabilisation exponentielle ou oscillations**

Afin de rendre concrètes les notions de stabilité, d'attracteurs et de convergence évoquées dans les sections précédentes (notamment 4.5.1.1 et 4.5.1.2), il est instructif d'étudier des cas à petite échelle, par exemple un SCN constitué de trois ou de quatre entités. Ces illustrations permettent de mettre en évidence, de manière numérique et graphique, comment la dynamique de mise à jour des pondérations $\omega_{i,j}$ peut converger de manière exponentielle vers un point fixe ou, au contraire, présenter des oscillations périodiques, en fonction du choix des paramètres d'apprentissage et de décroissance.

A. Cas à 3 entités : Stabilisation exponentielle

Considérons un SCN formé de trois entités, notées $\mathcal{E}_1$, $\mathcal{E}_2$ et $\mathcal{E}_3$. Pour simplifier l'analyse, nous supposerons que la **synergie** $S(i,j)$ entre chaque paire d'entités est stationnaire, c'est-à-dire constante au cours du temps. Par exemple, nous définissons les synergies par les valeurs suivantes :

$$S(1,2) = 0.8, \quad S(1,3) = 0.7, \quad S(2,3) = 0.2,$$

et par symétrie, $S(2,1) = 0.8$, $S(3,1) = 0.7$ et $S(3,2) = 0.2$. Dans ce scénario, la mise à jour des pondérations est réalisée selon la règle additive classique

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big],$$

où η est le **taux d'apprentissage** et τ est le **coefficient de décroissance**. Pour illustrer la stabilisation exponentielle, nous choisissons par exemple $\eta = 0.05$ et $\tau = 1.0$. L'initialisation des pondérations est faite avec des valeurs proches de zéro, par exemple, $\omega_{i,j}(0) \approx 0$ (avec une légère perturbation aléatoire pour simuler du bruit).

Au cours des itérations, les pondérations $\omega_{1,2}(t)$ et $\omega_{1,3}(t)$ augmentent progressivement car $S(1,2) = 0.8$ et $S(1,3) = 0.7$ sont relativement élevées, tandis que $\omega_{2,3}(t)$ croît plus lentement en

raison d'une synergie moindre $S(2,3) = 0.2$. Par l'analyse de la règle de mise à jour, nous pouvons observer qu'en supposant une convergence stationnaire, c'est-à-dire en posant $\omega_{i,j}(t+1) = \omega_{i,j}(t) = \omega_{i,j}^*$, nous obtenons :

$$\omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

Ainsi, pour le lien entre $\mathcal{E}_1$ et $\mathcal{E}_2$, on a $\omega_{1,2}^* = 0.8$; pour le lien entre $\mathcal{E}_1$ et $\mathcal{E}_3$, $\omega_{1,3}^* = 0.7$; et pour le lien entre $\mathcal{E}_2$ et $\mathcal{E}_3$, $\omega_{2,3}^* = 0.2$. En outre, la dynamique de convergence se caractérise par une décroissance exponentielle de l'écart, puisque la mise à jour linéaire entraîne que la différence $\left|\omega_{i,j}(t) - \omega_{i,j}^*\right|$ diminue en proportion d'un facteur constant $(1 - \eta\,\tau)$ à chaque itération. Graphiquement, en traçant $\omega_{1,2}(t)$ en fonction de t, on observe une courbe exponentielle ascendante se rapprochant asymptotiquement de 0.8. Ce cas illustre ainsi la **stabilité locale** dans un SCN simple, où la dynamique converge de manière prévisible vers des points fixes déterminés par la synergie stationnaire.

B. Cas à 4 entités : Apparition d'oscillations

Pour examiner un comportement différent, considérons maintenant un SCN comportant quatre entités, notées $\mathcal{E}_1$, $\mathcal{E}_2$, $\mathcal{E}_3$ et $\mathcal{E}_4$. Supposons des synergies stationnaires qui varient de façon à créer des interactions compétitives, par exemple :

$$S(1,2) = 0.9, \quad S(2,3) = 0.8, \quad S(3,4) = 0.7, \quad S(4,1) = 0.8,$$

et pour certaines paires moins influentes, $S(1,3) = 0.1$ et $S(2,4) = 0.2$. Dans ce scénario, nous adoptons une **mise à jour multiplicative** afin d'exacerber les effets non linéaires, en utilisant une règle du type

$$\omega_{i,j}(t+1) = \omega_{i,j}(t)\left[1 + \eta\,S(i,j) - \eta\,\tau\,\omega_{i,j}(t)\right].$$

Nous choisissons ici des paramètres plus agressifs, par exemple $\eta = 0.8$ et $\tau = 0.2$, et nous initialisons les pondérations avec une valeur faible, par exemple $\omega_{i,j}(0) \approx 0.01$. Dans ce régime, en raison du taux d'apprentissage élevé et de la faible décroissance, le système peut entrer dans une phase oscillatoire. Concrètement, les pondérations des liens avec des synergies élevées, telles que $\omega_{1,2}(t)$, $\omega_{2,3}(t)$, $\omega_{3,4}(t)$ et $\omega_{4,1}(t)$, vont d'abord augmenter rapidement. Cependant, lorsque l'un de ces liens atteint un niveau trop élevé, l'effet multiplicatif couplé à la rétroaction négative induite par $-\eta\,\tau\,\omega_{i,j}(t)$ provoque un sur-correction qui fait diminuer ce lien, tandis qu'un autre lien compense en augmentant, créant ainsi un cycle de **ping-pong** entre les différentes pondérations. Par exemple, $\omega_{1,2}(t)$ peut augmenter jusqu'à un certain point puis décroître brusquement lorsque $\omega_{2,3}(t)$ prend le relais, et vice versa. Ces oscillations se traduisent par des cycles quasi-périodiques dans la dynamique du SCN, et la convergence n'est pas atteinte de manière statique, mais le système évolue autour d'un attracteur cyclique.

Ce comportement oscillatoire, qui peut être observé en traçant les valeurs de $\omega_{i,j}(t)$ en fonction de t sur un graphique, démontre que le système, en l'absence d'inhibition ou de régulation supplémentaire, peut se comporter de manière non monotone, illustrant ainsi la sensibilité du DSL aux paramètres η et τ, et l'importance de mécanismes de contrôle supplémentaires (tels que l'inhibition latérale ou le clipping) pour éviter des comportements indésirables.

*C. **Illustration par simulation en Python***

Pour mettre en évidence ces deux régimes de comportement, on peut se référer à un exemple de simulation en Python. Dans le cas à 3 entités, la mise à jour additive est appliquée avec un taux d'apprentissage modéré et un coefficient de décroissance suffisant pour assurer une convergence exponentielle. En revanche, dans le cas à 4 entités, un modèle multiplicatif avec des paramètres agressifs engendre des oscillations. Voici un extrait de pseudo-code illustratif pour le cas à 3 entités :

```
import numpy as np
import matplotlib.pyplot as plt

# Définition des synergies stationnaires pour 3 entités
S = {(1,2): 0.8, (1,3): 0.7, (2,3): 0.2}
# Assurer la symétrie
for (i, j), val in list(S.items()):
    S[(j, i)] = val

eta = 0.05   # Taux d'apprentissage
tau = 1.0    # Coefficient de décroissance
num_iterations = 30

# Initialisation des poids (petit bruit autour de 0)
np.random.seed(42)
weights = {(i, j): np.random.uniform(-0.01, 0.01) for (i, j) in S.keys()}

# Enregistrement de l'évolution des poids pour chaque paire
trajectories = {key: [] for key in weights.keys()}

for t in range(num_iterations):
    for (i, j) in weights.keys():
        omega = weights[(i, j)]
        delta = eta * (S[(i, j)] - tau * omega)
        weights[(i, j)] += delta
        trajectories[(i, j)].append(weights[(i, j)])

# Tracé des trajectoires
plt.figure(figsize=(10, 6))
for (i, j), traj in trajectories.items():
    plt.plot(traj, label=f'$\omega_{{{i},{j}}}(t)$')
plt.axhline(0.8, color='blue', linestyle='--', label='$S(1,2)=0.8$')
plt.axhline(0.7, color='orange', linestyle='--', label='$S(1,3)=0.7$')
plt.axhline(0.2, color='green', linestyle='--', label='$S(2,3)=0.2$')
plt.title("Stabilisation exponentielle dans un SCN à 3 entités")
plt.xlabel("Itérations")
plt.ylabel("Pondération $\omega_{i,j}(t)$")
plt.legend()
```

```
plt.grid()
plt.show()
```

Pour le cas à 4 entités, une variante multiplicative avec des paramètres plus élevés (par exemple, $\eta = 0.8$ et $\tau = 0.2$) et l'introduction de termes non linéaires peut être utilisée pour simuler des oscillations, révélant un comportement cyclique dans la mise à jour des pondérations.

D. ***Conclusion***

Les exemples illustratifs sur de petits SCN, comportant 3 ou 4 entités, démontrent de manière concrète comment la dynamique de mise à jour des pondérations $\omega_{i,j}$ peut soit converger de manière exponentielle vers un point fixe (dans le cas stationnaire où $S(i,j)$ est constant et les paramètres η et τ sont bien choisis), soit présenter des oscillations persistantes (dans un régime non linéaire ou avec des paramètres trop agressifs). Ces comportements illustrent l'importance des choix paramétriques et des mécanismes additionnels (comme l'inhibition et le clipping) pour garantir la stabilité et la robustesse de l'auto-organisation du SCN dans un DSL. Ces exemples à petite échelle servent de laboratoire expérimental permettant de vérifier théoriquement la condition de contraction et de comprendre, par analogie, comment la dynamique se généralisera dans des systèmes de plus grande taille comportant de nombreux attracteurs et des interactions complexes.

4.5.2. Énergie ou Pseudo-Énergie

Une façon très féconde de **comprendre** la dynamique d'un **SCN** (Synergistic Connection Network) est d'introduire l'idée d'une **fonction d'énergie** (ou **pseudo-énergie**) qui, en quelque sorte, **mesure** la qualité de la configuration **ω**. Cette idée, inspirée de la **physique statistique** (chap. 2.4), permet d'envisager la mise à jour $\boldsymbol{\omega}(t)$ comme une forme de "**descente**" (ou de quasi-descente) dans un **paysage** énergétique — ou du moins comme un processus régulé par un critère qui ressemble à une minimisation.

4.5.2.1. **Cas simple :** $\mathcal{J}(\boldsymbol{\Omega}) = -\sum\omega_{i,j}\, S(i,j) + \frac{\tau}{2}\sum\left(\omega_{i,j}\right)^2$ **(2.4.3)**

Dans cette section, nous développons de manière approfondie le modèle énergétique qui sous-tend la mise à jour des pondérations dans un **Deep Synergy Learning (DSL)**. L'idée centrale est de définir une **fonction d'énergie** ou **fonction potentielle** $\mathcal{J}(\boldsymbol{\Omega})$ sur l'ensemble des pondérations **Ω** du **Synergistic Connection Network (SCN)**, de telle sorte que la dynamique d'auto-organisation puisse être interprétée comme une descente vers un minimum de cette énergie. La forme simple considérée ici est

$$\mathcal{J}(\boldsymbol{\Omega}) = -\sum_{(i,j)} \omega_{i,j}\; S(i,j) + \frac{\tau}{2}\sum_{(i,j)} (\,\omega_{i,j})^2,$$

où $S(i,j)$ désigne la **synergie** entre les entités $\mathcal{E}_i$ et $\mathcal{E}_j$, $\omega_{i,j}$ représente la **pondération** ou la force du lien entre ces entités, et τ est un paramètre qui module la décroissance des pondérations. Cette fonction $\mathcal{J}$ traduit l'intuition que l'on souhaite **maximiser** la synergie globale entre les entités tout en empêchant une croissance excessive des pondérations.

D'un point de vue interprétatif, le terme $-\sum \omega_{i,j}\, S(i,j)$ favorise l'augmentation des liens lorsque la synergie est élevée. En effet, plus $S(i,j)$ est grand, plus le produit $\omega_{i,j}\, S(i,j)$ contribue négativement à $\mathcal{J}$; puisque l'objectif de la descente consiste à minimiser $\mathcal{J}$, la dynamique incite à augmenter $\omega_{i,j}$ pour réduire la valeur absolue de ce terme négatif, ce qui correspond à renforcer les connexions entre entités fortement synergiques. À l'inverse, le terme de régularisation $\frac{\tau}{2}\sum\left(\omega_{i,j}\right)^2$ pénalise la croissance excessive des pondérations, en augmentant l'énergie si les valeurs de $\omega_{i,j}$ deviennent trop grandes. Ce compromis permet ainsi d'éviter une **croissance non contrôlée** des liens, garantissant que la solution convergera vers un équilibre.

Pour rendre cette intuition plus formelle, nous pouvons envisager une analyse variationnelle dans un cadre continu. Supposons que la dynamique de mise à jour soit modélisée par une équation différentielle approchée

$$\frac{d\,\omega_{i,j}}{dt} = \eta\left[S(i,j) - \tau\,\omega_{i,j}\right],$$

où η est le taux d'apprentissage, et où l'on suppose que $S(i,j)$ ne dépend pas de $\omega_{i,j}$. Dans ce cas, la dérivée partielle de $\mathcal{J}$ par rapport à $\omega_{i,j}$ est donnée par

$$\frac{\partial \mathcal{J}}{\partial \omega_{i,j}} = -\,S(i,j) + \tau\,\omega_{i,j}.$$

La descente de gradient, qui tend à minimiser $\mathcal{J}$, correspond à une dynamique

$$\frac{d\,\omega_{i,j}}{dt} = -\eta\,\frac{\partial \mathcal{J}}{\partial \omega_{i,j}} = \eta\left[S(i,j) - \tau\,\omega_{i,j}\right].$$

Cette équation confirme que la dynamique de mise à jour des pondérations est en effet équivalente à une **descente de gradient** sur $\mathcal{J}$. Au point fixe stationnaire, on pose $\frac{d\,\omega_{i,j}}{dt} = 0$ et l'on obtient

$$S(i,j) - \tau\,\omega_{i,j}^{*} = 0 \quad \Rightarrow \quad \omega_{i,j}^{*} = \frac{S(i,j)}{\tau}.$$

Ainsi, la solution d'équilibre, ou **attracteur**, est directement liée à la synergie $S(i,j)$ et régulée par le coefficient τ.

Il convient toutefois de noter que cette formulation constitue un **cas simple** dans lequel la synergie $S(i,j)$ est supposée constante au cours du temps et indépendante des pondérations. Dans des situations réelles, notamment lorsque des mécanismes non linéaires (inhibition latérale, couplages multiplicatifs, etc.) interviennent, la fonction $\mathcal{J}$ peut inclure des termes supplémentaires, et la descente de gradient n'est plus exacte. C'est pourquoi on parle parfois de **pseudo-énergie**, car la dynamique réelle du DSL tente de réduire $\mathcal{J}$ mais peut ne pas garantir une décroissance monotone à chaque itération en raison de la présence de rétroactions complexes.

La présence du terme de régularisation $\frac{\tau}{2}\sum\left(\omega_{i,j}\right)^2$ joue un rôle essentiel, car il agit comme un **coût** associé au maintien d'un lien fort. Ce mécanisme empêche une croissance illimitée des pondérations et assure qu'en l'absence de synergie suffisante, les liens se désactivent

progressivement. Ainsi, l'équilibre atteint, $\omega_{i,j}^{*} = \frac{S(i,j)}{\tau}$, représente le compromis optimal entre le renforcement des liens et la régulation nécessaire pour préserver la **stabilité** du système.

En résumé, la fonction d'énergie

$$\mathcal{J}(\mathbf{\Omega}) = -\sum_{(i,j)} \omega_{i,j}\ S(i,j) + \frac{\tau}{2}\sum_{(i,j)} (\omega_{i,j})^2,$$

formalise l'intuition selon laquelle le système cherche à **maximiser** la synergie globale $\sum \omega_{i,j}\, S(i,j)$ tout en **pénalisant** la croissance excessive des liens via le terme quadratique. La dynamique induite par la mise à jour additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\, \omega_{i,j}(t)\big]$$

correspond alors à une descente de gradient sur $\mathcal{J}$ dans un cas simple. Cette approche permet d'atteindre un point fixe stable caractérisé par

$$\omega_{i,j}^{*} = \frac{S(i,j)}{\tau},$$

ce qui offre une interprétation claire en termes d'**équilibre** entre l'activation induite par la synergie et la régulation imposée par la décroissance. Cette formulation constitue la base théorique sur laquelle s'appuient des mécanismes plus complexes dans le DSL, et sert de référence pour analyser la stabilité et la convergence des pondérations dans le réseau, même lorsque des non-linéarités supplémentaires sont introduites.

4.5.2.2. **Descente locale vs. paysage évolutif si $S(i,j)$ change**

Dans l'analyse de la dynamique d'un **Deep Synergy Learning (DSL)**, la fonction d'énergie $\mathcal{J}(\boldsymbol{\omega})$ joue un rôle central pour comprendre comment la mise à jour des pondérations $\omega_{i,j}$ tend à converger vers un minimum. Dans la partie précédente, nous avons étudié le cas simple où $S(i,j)$ est considéré comme constant, ce qui conduit à une descente locale dans un paysage énergétique fixe. Cependant, dans de nombreux systèmes réels, la synergie $S(i,j)$ varie au fil du temps, en raison de la mise à jour continue des représentations (embeddings) ou de l'évolution des règles logiques. Cette variabilité entraîne la formation d'un **paysage évolutif** où la fonction d'énergie se modifie continuellement, et, par conséquent, la dynamique d'auto-organisation ne se stabilise pas de manière statique. Nous détaillons ci-après les deux scénarios et leurs implications.

*A. **Descente locale lorsque $S(i,j)$ est constant***

Lorsque la représentation des entités est stable, la synergie $S(i,j)$ est considérée comme une valeur fixe, ce qui signifie que, pour toute paire d'entités, le score $S(i,j)$ reste inchangé au cours des itérations. Dans ce cas, la fonction d'énergie

$$\mathcal{J}(\boldsymbol{\omega}) = -\sum_{i,j} \omega_{i,j}\ S(i,j) + \frac{\tau}{2}\sum_{i,j} (\omega_{i,j})^2$$

devient indépendante du temps. La mise à jour des pondérations est alors réalisée par la règle additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big],$$

et le système se comporte comme une descente de gradient sur $\mathcal{J}$. En effet, en posant l'équilibre stationnaire $\omega_{i,j}(t+1) = \omega_{i,j}(t) = \omega^*_{i,j}$, on obtient immédiatement

$$S(i,j) - \tau\,\omega^*_{i,j} = 0 \quad \Rightarrow \quad \omega^*_{i,j} = \frac{S(i,j)}{\tau}.$$

Dans ce contexte, toute perturbation autour de ce point fixe est contractée par le système, de sorte que la dynamique converge de manière exponentielle vers $\omega^*_{i,j}$. La vitesse de convergence est essentiellement dictée par le facteur $1 - \eta\,\tau$, et l'écart à l'équilibre se réduit comme $|\omega_{i,j}(t) - \omega^*_{i,j}| \sim (1 - \eta\,\tau)^t$. Ce cas de **descente locale** est idéal pour l'analyse théorique, car il offre un cadre simple et linéaire où le paysage énergétique $\mathcal{J}$ est fixe, et où les mécanismes d'auto-organisation peuvent être décrits de manière analytique.

*B. **Paysage évolutif lorsque** $S(i,j)$ **change***

En réalité, de nombreux **DSL** opèrent dans des environnements non stationnaires, où la synergie $S(i,j)$ est recalculée à chaque itération. Cette situation peut se produire lorsque :

- Les **embeddings** des entités sont continuellement mis à jour par des processus de fine-tuning ou d'apprentissage continu, entraînant ainsi une modification de $S(i,j)$ basée sur la proximité vectorielle.
- De nouvelles entités sont introduites dans le système, modifiant la distribution globale et redéfinissant ainsi les relations entre les entités existantes.
- Les **règles symboliques** évoluent (par exemple, par ajout ou suppression d'axiomes), ce qui modifie la composante logique du score de synergie.

Dans un tel scénario, la fonction d'énergie devient dépendante du temps et s'écrit de manière implicite comme

$$\mathcal{J}(\boldsymbol{\omega}, t) = -\sum_{i,j} \omega_{i,j}\; S(i,j,t) + \frac{\tau}{2}\sum_{i,j} (\,\omega_{i,j})^2,$$

où $S(i,j,t)$ varie à chaque itération. La dynamique de mise à jour s'adapte alors en temps réel :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j,t) - \tau\,\omega_{i,j}(t)\big].$$

Dans ce **paysage évolutif**, le minimum de $\mathcal{J}$ n'est pas fixe mais se déplace continuellement. Le système tente alors de « suivre » ce minimum mouvant, ce qui peut conduire à des ajustements constants des pondérations. Si les variations de $S(i,j,t)$ sont **douces**, la dynamique peut suivre le déplacement du minimum local, et le SCN se réorganise progressivement sans perdre sa cohérence. Cependant, si $S(i,j,t)$ subit des variations brutales – par exemple, en cas d'insertion massive de nouvelles entités ou de modifications importantes dans les représentations – le paysage énergétique

peut se transformer de manière significative, et le système peut alors se retrouver dans une situation de **rebondissement** où les pondérations oscillent ou se réajustent continuellement, sans parvenir à une stabilisation définitive.

Ces phénomènes peuvent être analysés en considérant la dérivée temporelle de $\mathcal{J}$ et en observant que, contrairement au cas stationnaire, le gradient $-\nabla\mathcal{J}(\boldsymbol{\omega}, t)$ varie avec t. Le système est alors contraint de s'adapter à un environnement énergétique qui évolue, et la dynamique d'auto-organisation peut présenter des comportements complexes, tels que des transitions de phase ou des oscillations persistantes.

C. Conséquences pratiques et stratégies d'adaptation

Dans un DSL opérant dans un paysage évolutif, la stabilité du SCN est moins garantie que dans le cas stationnaire. La non-stationnarité de $S(i,j,t)$ requiert des mécanismes supplémentaires pour assurer la **robustesse** du système. Parmi ces mécanismes, on peut citer :

- **Lissage des embeddings** : en réévaluant les représentations de manière graduelle, par exemple en utilisant une mise à jour du type

  $$\mathbf{x}_i(t+1) \leftarrow \alpha\, \mathbf{x}_i(t) + (1-\alpha)\, g_{t+1}(\mathbf{d}_i),$$

  où g_{t+1} représente la fonction d'extraction mise à jour et $\alpha \in [0,1]$ un coefficient de lissage, on atténue les variations brusques de $S(i,j,t)$.

- **Réindexation partielle** : lorsqu'un grand changement se produit (nouveaux modèles, nouvelles entités), il peut être nécessaire de recalculer globalement ou partiellement les pondérations $\omega_{i,j}$ pour réaligner le SCN avec la nouvelle configuration du paysage énergétique.

- **Utilisation de mécanismes de contrôle** : des stratégies inspirées du recuit stochastique ou de la régulation adaptative peuvent être mises en place pour permettre au système de "suivre" le minimum local tout en évitant des oscillations excessives ou des divergences.

Ces stratégies permettent d'adapter le SCN à un environnement où $S(i,j,t)$ évolue continuellement, assurant ainsi une **adaptabilité** optimale du DSL même lorsque le paysage énergétique est en mouvement.

D. Conclusion

En résumé, lorsque la synergie $S(i,j)$ est constante, la dynamique de mise à jour des pondérations $\omega_{i,j}$ s'apparente à une **descente locale** dans un paysage énergétique fixe, conduisant à la convergence vers un point fixe défini par

$$\omega_{i,j}^{*} = \frac{S(i,j)}{\tau}.$$

Cette situation permet une analyse simple de la stabilité du système via la descente de gradient. En revanche, lorsque $S(i,j)$ varie au fil du temps – en raison de mises à jour des représentations ou d'insertion de nouvelles données – le paysage énergétique $\mathcal{J}(\boldsymbol{\omega}, t)$ devient mobile, et le SCN doit s'adapter continuellement. Dans ce cas, le système ne se fige pas dans un attracteur fixe, mais réorganise ses pondérations pour suivre le minimum local en évolution. Ce comportement

dynamique, qui peut se traduire par des réajustements constants voire des oscillations, souligne l'importance de mécanismes de contrôle supplémentaires pour garantir la **robustesse** du DSL dans des environnements non stationnaires. La distinction entre descente locale et paysage évolutif permet ainsi de mieux comprendre comment la variabilité de $S(i,j)$ influence la **stabilité**, la **convergence** et l'**adaptabilité** du système, fournissant une base théorique pour l'analyse des performances d'un DSL dans des contextes évolutifs et réels.

4.5.2.3. **Cas stationnaire :** $\Delta\omega_{i,j} = 0 \Rightarrow \omega_{i,j}^{*} = \frac{S(i,j)}{\tau}$

Dans ce cas, nous nous intéressons à la situation idéale dans laquelle la **synergie** $S(i,j)$ entre chaque paire d'entités est considérée comme **constante** au cours du temps, ce qui conduit à une dynamique de mise à jour des pondérations qui ne dépend que de la valeur actuelle de $\omega_{i,j}$ et des paramètres du système. Cette hypothèse simplificatrice permet de décrire le comportement stationnaire du système, c'est-à-dire le point fixe vers lequel tend la dynamique d'auto-organisation du **Synergistic Connection Network (SCN)**.

Soit la fonction d'**énergie** (ou pseudo-énergie) définie par

$$\mathcal{J}(\boldsymbol{\omega}) = -\sum_{i,j} \omega_{i,j}\, S(i,j) + \frac{\tau}{2}\sum_{i,j} (\,\omega_{i,j})^2,$$

où $S(i,j)$ est un score de synergie constant pour la paire (i,j) et τ est le paramètre de décroissance. L'objectif est de minimiser $\mathcal{J}$ afin de trouver une configuration optimale des pondérations. En effet, le premier terme, $-\sum\omega_{i,j}\, S(i,j)$, encourage le renforcement des liens lorsque la synergie est élevée, tandis que le terme de régularisation $\frac{\tau}{2}\sum(\omega_{i,j})^2$ pénalise une croissance excessive des pondérations. Le compromis entre ces deux contributions conduit à un minimum d'énergie qui définit l'équilibre du système.

Pour étudier ce cas stationnaire, nous utilisons la **règle additive** de mise à jour des pondérations donnée par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big],$$

où η représente le **taux d'apprentissage**. Dans un état stationnaire, on suppose que la dynamique ne fait plus évoluer les pondérations, c'est-à-dire que

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) = \omega_{i,j}^{*}.$$

En imposant la condition stationnaire, nous avons :

$$\omega_{i,j}^{*} = \omega_{i,j}^{*} + \eta\big[S(i,j) - \tau\,\omega_{i,j}^{*}\big].$$

La seule solution non triviale de cette équation est obtenue en annulant le terme entre parenthèses, ce qui conduit directement à

$$S(i,j) - \tau\,\omega_{i,j}^{*} = 0 \quad \Rightarrow \quad \omega_{i,j}^{*} = \frac{S(i,j)}{\tau}.$$

Cette équation montre que, dans un environnement stationnaire, la pondération entre deux entités converge vers une valeur proportionnelle à la synergie mesurée, modulée par le paramètre de décroissance τ. En d'autres termes, si la synergie est élevée, le lien se renforce jusqu'à atteindre une valeur d'équilibre donnée par $\omega_{i,j}^{*} = \frac{S(i,j)}{\tau}$. À l'inverse, si $S(i,j)$ est nul ou très faible, alors le lien tendra à disparaître (i.e., $\omega_{i,j}^{*}$ sera proche de 0).

Pour approfondir cette analyse, on peut interpréter la règle de mise à jour comme une **descente de gradient** sur la fonction d'énergie $\mathcal{J}$. En effet, la dérivée partielle de $\mathcal{J}$ par rapport à $\omega_{i,j}$ est donnée par

$$\frac{\partial \mathcal{J}}{\partial \omega_{i,j}} = -S(i,j) + \tau\, \omega_{i,j}.$$

La dynamique de mise à jour du DSL, écrite en termes continus, s'exprime par

$$\frac{d\, \omega_{i,j}}{dt} = -\eta\, \frac{\partial \mathcal{J}}{\partial \omega_{i,j}} = \eta\big[S(i,j) - \tau\, \omega_{i,j}\big],$$

ce qui est exactement la forme de notre règle de mise à jour. Par conséquent, l'évolution des pondérations est guidée par le gradient de $\mathcal{J}$ et tend à réduire l'énergie du système jusqu'à ce que le gradient soit nul, c'est-à-dire quand $\omega_{i,j}$ atteint l'équilibre.

Un **exemple numérique** simple peut illustrer ce point. Supposons que pour une paire d'entités, la synergie stationnaire est $S(i,j) = 0.8$ et que $\tau = 1.0$. Si l'on initialise $\omega_{i,j}(0)$ à une valeur proche de zéro, la mise à jour s'effectuera de manière à faire croître $\omega_{i,j}$ vers

$$\omega_{i,j}^{*} = \frac{0.8}{1.0} = 0.8.$$

On observe alors, par exemple, une convergence exponentielle telle que

$$\omega_{i,j}(t) \approx 0.8[1 - (1 - \eta\tau)^{t}],$$

ce qui traduit la réduction exponentielle de l'écart par rapport à l'équilibre. Graphiquement, une courbe de $\omega_{i,j}(t)$ en fonction de t montrerait une ascension rapide suivie d'un plateau asymptotique à 0.8, confirmant que la dynamique d'auto-organisation a atteint un état stationnaire.

Dans le cadre de cette approche, le terme $\tau\, \omega_{i,j}^{2}$ dans la fonction d'énergie sert de **pénalité** qui empêche la croissance excessive des pondérations. C'est ce terme régulateur qui assure la **stabilité** du système en imposant un coût à la maintenance de liens trop forts, et qui, conjointement avec le terme de synergie $\omega_{i,j}\, S(i,j)$, détermine l'équilibre optimal.

Conclusion

En conclusion, dans un **cas stationnaire** où la synergie $S(i,j)$ reste constante, la règle de mise à jour additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\, \omega_{i,j}(t)\big]$$

permet d'atteindre un point fixe $\omega_{i,j}^*$ donné par

$$\omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

Cette solution correspond à un minimum local de la fonction d'énergie

$$\mathcal{J}(\boldsymbol{\omega}) = -\sum_{i,j} \omega_{i,j}\, S(i,j) + \frac{\tau}{2}\sum_{i,j} (\omega_{i,j})^2,$$

et reflète l'équilibre entre l'effet de renforcement induit par la synergie et la pénalité qui empêche la croissance illimitée des liens. La dynamique de mise à jour agit alors comme une descente de gradient sur $\mathcal{J}$, assurant que toute perturbation autour du point fixe est progressivement amortie, ce qui garantit la stabilité du SCN dans un environnement stationnaire. Cette analyse fournit ainsi une base théorique solide pour comprendre la convergence et la stabilité des pondérations dans un DSL, dans des conditions idéales où le paysage énergétique reste fixe.

4.5.3. Renvoi vers Chapitre 2.3 et 2.4

Les sections précédentes (4.5.1 et 4.5.2) ont souligné l'importance de certains **concepts** (attracteurs, point fixe, énergie ou pseudo-énergie) pour comprendre la dynamique d'un **SCN** (Synergistic Connection Network). Mais l'origine de ces notions se situe déjà en **Chapitre 2**, lorsque nous abordions les **fondements théoriques** du DSL (Deep Synergy Learning) : la possibilité de multi-stabilité (2.3.2), l'analogie avec la physique statistique et les systèmes dynamiques (2.4.3). Cette sous-section (4.5.3) vise à **faire le lien** explicite avec ces passages et à voir **comment** on exploite concrètement (i.e. de manière "appliquée") ces idées dans un DSL.

4.5.3.1. **Bref rappel des sections 2.3.2 (attracteurs multiples) et 2.4.3 (notion d'énergie)**

Dans l'analyse théorique du Deep Synergy Learning (DSL), les chapitres 2.3.2 et 2.4.3 fournissent respectivement les fondations conceptuelles concernant la co-existence d'attracteurs multiples et la formulation d'une fonction d'énergie sur l'espace des pondérations. Ces deux approches, bien que relevant de domaines différents de la théorie des systèmes dynamiques, se rejoignent pour expliquer la manière dont le Synergistic Connection Network (SCN) se structure et converge vers des configurations stables.

A. ***Attracteurs multiples (Chap. 2.3.2)***

La notion d'attracteurs multiples trouve son origine dans la théorie des systèmes dynamiques, où l'on démontre que, pour un système non linéaire, il peut exister plusieurs points fixes ou cycles stables, c'est-à-dire plusieurs attracteurs locaux. Mathématiquement, si l'on définit la fonction de mise à jour globale du SCN par

$$\mathbf{F}: \mathcal{W} \to \mathcal{W},$$

un point fixe $\boldsymbol{\Omega}^*$ vérifie

$$\boldsymbol{\Omega}^* = \mathbf{F}(\boldsymbol{\Omega}^*).$$

Dans des systèmes complexes tels que le SCN, il est souvent constaté que l'espace des pondérations se divise en différents bassins d'attraction. Ainsi, l'initialisation de $\boldsymbol{\omega}(0)$ détermine vers quel attracteur le système convergera. Cette multi-stabilité se manifeste par la formation de **clusters alternatifs** ou de configurations distinctes, chacune étant localement stable. Par analogie avec les modèles de spin (tels que le modèle d'Ising ou de Potts) et les réseaux de Hopfield, ces attracteurs multiples illustrent la possibilité pour le système d'atteindre des états globalement différents, bien que chacun satisfasse la condition d'équilibre local.

L'existence de plusieurs attracteurs implique également que la dynamique du SCN peut être sensible aux perturbations et que de légères variations dans l'état initial peuvent conduire à des résultats radicalement différents. Cette propriété, bien que complexe à analyser de manière exhaustive dans des espaces de grande dimension, constitue la base de nombreux phénomènes de **phase** et de **transition** observés dans les systèmes d'apprentissage non supervisé.

*B. **Notion d'énergie (Chap. 2.4.3)***

Le chapitre 2.4.3 introduit l'idée d'une fonction d'énergie ou fonction potentielle $\mathcal{J}(\boldsymbol{\omega})$ définie sur l'ensemble des pondérations du SCN. Une forme simple de cette fonction est donnée par

$$\mathcal{J}(\boldsymbol{\omega}) = -\sum_{i,j} \omega_{i,j}\, S(i,j) + \frac{\tau}{2}\sum_{i,j} (\,\omega_{i,j})^2,$$

où $S(i,j)$ représente la synergie entre les entités $\mathcal{E}_i$ et $\mathcal{E}_j$ et τ est le coefficient de décroissance, jouant ainsi le rôle d'un terme de régularisation. Cette fonction d'énergie est conçue de manière à ce que, lorsque l'on cherche à la minimiser (ou, de manière équivalente, à maximiser $\sum \omega_{i,j}\, S(i,j)$ tout en pénalisant la croissance excessive des pondérations), le système converge vers un état stable où les pondérations $\omega_{i,j}$ atteignent des valeurs optimales.

En effet, dans le cas stationnaire où $S(i,j)$ est constant, la descente de gradient sur $\mathcal{J}$ conduit à la condition d'équilibre

$$\frac{\partial \mathcal{J}}{\partial \omega_{i,j}} = -S(i,j) + \tau\, \omega_{i,j} = 0,$$

ce qui implique que

$$\omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

Cette relation fournit une interprétation élégante du compromis entre le renforcement d'un lien (favorisé par une synergie élevée) et la régulation de la croissance des pondérations (imposée par le terme quadratique). La fonction d'énergie permet également de visualiser le **paysage** dans lequel évolue le SCN. Dans ce paysage, chaque attracteur correspond à un minimum local de $\mathcal{J}$. Des analogies fortes peuvent être établies avec des systèmes physiques, tels que les réseaux de neurones de Hopfield, où la dynamique du système converge vers des états d'énergie minimale.

C. Lien entre attracteurs multiples et fonction d'énergie

L'héritage théorique posé dans le Chapitre 2 montre que la co-existence de plusieurs attracteurs découle naturellement de la forme de la fonction d'énergie. En effet, dans un système non linéaire, $\mathcal{J}(\boldsymbol{\omega})$ peut présenter plusieurs vallées ou minima locaux, chacun correspondant à un ensemble stable de pondérations. Ainsi, en fonction de l'initialisation du système, le SCN convergera vers l'un de ces attracteurs, ce qui explique la **multi-stabilité** observée. Les attracteurs multiples sont alors interprétés comme des **clusters alternatifs** ou des configurations distinctes du réseau, chaque minimum local traduisant une organisation particulière des liens, qui peut être analysée en termes de similarité des entités et de la cohérence de leur interaction.

Ces concepts se retrouvent dans de nombreux modèles d'auto-organisation et d'apprentissage non supervisé. Par exemple, dans les réseaux de Hopfield, la dynamique converge vers des états mémorisés qui représentent des minima de l'énergie, tandis que dans les modèles de spin, la configuration du système est déterminée par l'équilibre entre les interactions locales et l'influence de l'environnement.

D. Résumé du lien entre Chapitre 2 et Chapitre 4

Le **Chapitre 2** pose les fondations théoriques en introduisant la notion d'attracteurs multiples et en définissant une fonction d'énergie $\mathcal{J}(\boldsymbol{\omega})$ qui guide la dynamique d'auto-organisation du SCN. Cette approche conceptuelle permet d'interpréter la dynamique de mise à jour des pondérations comme une **descente d'énergie** qui, dans le cas stationnaire, conduit à des points fixes stables, et dans des systèmes plus complexes, à une multi-stabilité ou à des comportements oscillatoires.

Le **Chapitre 4** poursuit cette analyse en développant de manière plus "ingénierie" les mécanismes de mise à jour concrets, tels que la règle additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big],$$

et en explorant les stratégies de contrôle (inhibition, clipping, etc.) qui permettent d'ajuster la dynamique du SCN pour éviter des comportements indésirables. Ainsi, la théorie des attracteurs et de la fonction d'énergie fournie au Chapitre 2 constitue la **base conceptuelle** sur laquelle s'appuie la mise en pratique détaillée dans le Chapitre 4, illustrant ainsi la continuité entre une vision théorique et une application pratique dans le cadre d'un DSL.

Conclusion

En somme, le rappel des sections 2.3.2 et 2.4.3 met en lumière deux aspects complémentaires de l'analyse des systèmes dynamiques dans un DSL. D'une part, la notion d'attracteurs multiples permet de comprendre comment différentes configurations stables peuvent coexister, en fonction des conditions initiales et des interactions locales entre entités. D'autre part, la fonction d'énergie $\mathcal{J}(\boldsymbol{\omega})$ offre un cadre mathématique pour interpréter la dynamique de mise à jour des pondérations comme une descente vers un minimum, où le compromis entre renforcement par la synergie et régularisation par la décroissance est clairement établi. Ces concepts théoriques servent de socle à la mise en œuvre pratique, telle que développée dans le Chapitre 4, et illustrent comment un SCN peut évoluer de manière stable ou osciller, selon la structure du paysage énergétique et les mécanismes de contrôle appliqués.

4.5.3.2. **Comment on opérationnalise ces idées maintenant dans un cadre "appliqué"**

Dans cette section, nous développons en détail comment les concepts théoriques introduits dans le Chapitre 2 — tels que la notion d'énergie $\mathcal{J}(\boldsymbol{\omega})$, les attracteurs multiples et la descente d'énergie — sont traduits en procédures concrètes dans un système de **Deep Synergy Learning (DSL)**. L'objectif est de montrer comment, à travers des algorithmes implémentés en code, on peut mettre en œuvre la mise à jour des pondérations $\omega_{i,j}$ de manière à faire émerger des clusters stables ou des configurations attractives, tout en adaptant le système aux évolutions de la représentation.

*A. **Mise en œuvre de la descente d'énergie***

L'idée fondamentale est de considérer la mise à jour des pondérations comme une descente de gradient sur une fonction d'énergie, qui dans sa forme simple est donnée par

$$\mathcal{J}(\boldsymbol{\omega}) = -\sum_{i,j} \omega_{i,j}\ S(i,j) + \frac{\tau}{2}\sum_{i,j} (\,\omega_{i,j})^2,$$

où $S(i,j)$ représente la synergie entre les entités $\mathcal{E}_i$ et $\mathcal{E}_j$ et τ contrôle la pénalité appliquée aux grandes pondérations. Dans un cadre appliqué, cette fonction d'énergie sert à guider la mise à jour des poids par une procédure itérative. Plus précisément, l'algorithme se base sur la règle de mise à jour additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big],$$

qui peut être interprétée comme une descente de gradient sur $\mathcal{J}$ lorsque l'on observe que

$$\frac{\partial \mathcal{J}}{\partial \omega_{i,j}} = -S(i,j) + \tau\,\omega_{i,j}.$$

En pratique, on initialise le vecteur des pondérations $\boldsymbol{\omega}(0)$ (par exemple, en partant d'un bruit faible autour de zéro) et on exécute la boucle itérative suivante :

4. **Calcul du gradient local :** Pour chaque paire (i,j), le gradient local $-S(i,j) + \tau\,\omega_{i,j}(t)$ est évalué. Ce terme indique la direction dans laquelle il faut modifier la pondération pour réduire l'énergie.

5. **Mise à jour :** On ajuste la pondération selon

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) - \eta\,\frac{\partial \mathcal{J}}{\partial \omega_{i,j}} = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big].$$

6. **Clipping et régularisation :** Pour éviter des valeurs aberrantes ou une croissance non contrôlée, on applique des mécanismes tels que le clipping (par exemple, forcer $\omega_{i,j}(t+1) \in [0, \omega_{\max}]$) et éventuellement des termes d'inhibition.

L'ensemble de ces opérations s'exécute de manière itérative jusqu'à ce que la variation entre deux itérations consécutives, $\| \boldsymbol{\omega}(t+1) - \boldsymbol{\omega}(t) \|$, devienne inférieure à un seuil prédéfini, indiquant ainsi que le système a atteint un état quasi-stationnaire.

B. ***Gérer la multi-stabilité et l'adaptation dans un environnement évolutif***

Dans des applications réelles, le système peut être amené à évoluer, que ce soit par l'insertion de nouvelles entités ou par l'actualisation des représentations (embeddings ou règles). Cela se traduit par une modification du paysage énergétique $\mathcal{J}(\boldsymbol{\omega}, t)$ au fil du temps. Pour gérer ce phénomène, plusieurs stratégies opérationnelles peuvent être mises en place :

- **Mécanismes de lissage temporel :** Afin de limiter les variations brutales des synergies $S(i,j)$ dues à des mises à jour continues des représentations, on peut utiliser un schéma de lissage, par exemple en mettant à jour l'embedding $\mathbf{x}_i$ de manière progressive selon la formule

$$\mathbf{x}_i(t+1) \leftarrow \alpha\, \mathbf{x}_i(t) + (1-\alpha)\, g_{t+1}(\mathbf{d}_i),$$

  où g_{t+1} est la fonction d'extraction mise à jour et $\alpha \in [0,1]$ est un coefficient de lissage. Cela permet de faire évoluer $S(i,j,t)$ de manière graduelle, de sorte que la descente de gradient sur $\mathcal{J}$ ne soit pas perturbée par des sauts trop brusques dans le paysage énergétique.

- **Réindexation et recalibration :** En cas de modification significative du système (par exemple, l'arrivée de nombreuses nouvelles entités), il peut être nécessaire de réinitialiser ou recalibrer partiellement les pondérations $\omega_{i,j}$ pour aligner le SCN avec la nouvelle configuration des représentations.

- **Utilisation de techniques de recuit stochastique :** Pour aider le système à sortir des minima locaux peu pertinents et explorer de nouveaux attracteurs plus profonds, on peut injecter du bruit contrôlé dans la mise à jour des pondérations. Ceci est analogue à un recuit stochastique où la "température" décroît progressivement afin de permettre au système d'échapper aux attracteurs peu stables tout en se stabilisant finalement dans un minimum de $\mathcal{J}$.

Ces mécanismes assurent que le SCN reste **adaptatif** et capable de suivre l'évolution des représentations, même lorsque le paysage énergétique est en constante mutation.

C. ***Mise en œuvre opérationnelle dans un cadre appliqué***

Pour traduire ces idées dans un cadre appliqué, il convient de développer un pipeline algorithmique qui intègre à la fois la descente de gradient sur $\mathcal{J}$ et les stratégies d'adaptation nécessaires à un environnement non stationnaire. Concrètement, le pipeline peut être structuré en plusieurs étapes clés :

- **Initialisation des pondérations :** Définir $\boldsymbol{\omega}(0)$ avec des valeurs faibles, éventuellement avec un bruit aléatoire pour simuler des conditions réelles.
- **Calcul de la synergie :** Pour chaque paire d'entités, calculer $S(i,j)$ en fonction des représentations actuelles. Dans un environnement appliqué, ce calcul peut être basé sur des embeddings extraits par des modèles pré-entraînés ou sur la vérification de règles logiques pour des entités symboliques.
- **Boucle itérative de mise à jour :**
  - **Évaluation du gradient :** Pour chaque lien, calculer la variation $\Delta\omega_{i,j} = \eta\big[S(i,j) - \tau\, \omega_{i,j}(t)\big]$.

- **Application de la mise à jour :** Actualiser $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta\omega_{i,j}$.
- **Régulation :** Appliquer des mécanismes de clipping et, le cas échéant, des termes d'inhibition ou de recuit pour contrôler la croissance.

- **Surveillance de la convergence :** Calculer la différence $\| \boldsymbol{\omega}(t+1) - \boldsymbol{\omega}(t) \|$ et suivre l'évolution de la valeur de $\mathcal{J}(\boldsymbol{\omega}(t))$. Une fois ces valeurs stabilisées, le système est considéré comme convergent.
- **Adaptation en ligne :** En cas de changement dans les représentations (mise à jour des embeddings ou des règles), relancer le processus d'actualisation, éventuellement en réutilisant un schéma de lissage pour éviter des réajustements trop brusques.

Ces étapes peuvent être implémentées dans un cadre logiciel (par exemple en Python, avec des bibliothèques de calcul scientifique comme NumPy et des frameworks de deep learning), permettant ainsi d'opérationnaliser les concepts théoriques issus du Chapitre 2 dans un environnement appliqué.

*D. **Conclusion***

La mise en œuvre opérationnelle des idées théoriques de la descente d'énergie et des attracteurs dans un DSL se traduit par la programmation d'un algorithme de mise à jour itérative qui cherche à minimiser une fonction d'énergie $\mathcal{J}(\boldsymbol{\omega})$. Dans un cadre appliqué, ce processus inclut non seulement le calcul de la mise à jour des pondérations par descente de gradient, mais aussi des stratégies d'adaptation pour gérer la variabilité des représentations – par exemple, via le lissage des embeddings, la réindexation ou des techniques de recuit stochastique. Le résultat final est un SCN capable de former des clusters stables ou adaptatifs, en fonction de la stationnarité ou non stationnarité du paysage énergétique. Ainsi, les notions de multi-stabilité, de descente d'énergie et de convergence théorique posées au Chapitre 2 sont opérationnalisées dans un algorithme concret, permettant d'assurer à la fois la robustesse et l'adaptabilité du DSL en environnement réel. Cette approche permet également d'intervenir avec des mécanismes de contrôle (inhibition, clipping) pour orienter la dynamique vers des configurations optimales en fonction des besoins de l'application.

4.6. Extraction et Visualisation des Clusters Émergents

Au cours de la dynamique d'un **SCN** (Synergistic Connection Network), les pondérations $\omega_{i,j}$ évoluent et finissent souvent par révéler un **partitionnement** implicite (voir chapitres précédents). Une question cruciale consiste alors à **observer** et **visualiser** ces pondérations, de sorte à comprendre **comment** les clusters apparaissent et **quand** la structure du réseau se stabilise ou se modifie. Cette section (4.6) présente les principales **méthodes** pour :

7. **Suivre** l'évolution de la matrice $\{\omega_{i,j}\}$ en temps réel (4.6.1),
8. **Identifier** de façon automatique les clusters à partir de cette matrice (4.6.2),
9. **Analyser** les évolutions temporelles éventuelles de ces regroupements (4.6.3).

4.6.1. Méthodes d'Observation

La première étape pour extraire et comprendre les **clusters** consiste à **observer** la matrice $\{\omega_{i,j}\}$ et ses changements au fil du temps. Cette approche est particulièrement importante lorsque la dynamique est complexe (multiplicative, inhibition, etc.) ou qu'il y a un **apprentissage continu** (synergie $S(i,j)$ qui se met à jour).

4.6.1.1. Suivi de la Matrice $\{\omega_{i,j}\}$ au Fil du Temps

Dans la dynamique d'**auto-organisation** d'un **Synergistic Connection Network** (*SCN*), la matrice $\{\omega_{i,j}(t)\}$ joue un rôle fondamental en reflétant l'état courant du réseau à chaque itération t. Les pondérations $\omega_{i,j}(t)$, ajustées selon une équation locale (ex. $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right]$), traduisent la **force de liaison** entre deux entités $\mathcal{E}_i$ et $\mathcal{E}_j$. Observer leur évolution au fil du temps permet d'**analyser** la progression du SCN dans sa quête de **clusters** ou de **sous-réseaux** cohérents, tout en signalant d'éventuels déséquilibres (phénomènes d'oscillation, croissance excessive de liens, inertie). Cette démarche de *monitoring* revêt donc une valeur heuristique considérable pour comprendre si le réseau tend vers un **état stable** ou si, au contraire, il oscille ou se fragmente.

Le **suivi** de $\omega_{i,j}(t)$ consiste à archiver ou à sonder la matrice à intervalle régulier, afin de détecter comment les liens les plus significatifs se renforcent ou s'affaiblissent. Dans un SCN de taille modeste (n de l'ordre de quelques centaines), il est envisageable de **stocker** l'intégralité de la matrice $\{\omega_{i,j}(t)\}$ à chaque itération, pour reconstituer a posteriori l'évolution complète. On dispose ainsi d'un véritable "film" de la progression, ce qui facilite la mise en évidence de la formation progressive de **blocs** ou de "zones actives" où les pondérations montent en flèche. À l'inverse, dans un **réseau** plus volumineux (n allant jusqu'à plusieurs milliers ou dizaines de milliers), l'espace requis pour conserver la totalité des $O(n^2)$ liens à chaque étape s'avère prohibitif, et l'on recourt alors à un **échantillonnage** partiel (en ne conservant la matrice que toutes les 10 ou 50 itérations, ou en ne stockant qu'un sous-ensemble restreint de liens jugés pertinents). On peut aussi suivre des **indicateurs statistiques** (moyenne, variance, quantiles de $\omega_{i,j}$) pour dresser un portrait global de la dynamique, sans conserver chaque élément de la matrice.

Cette observation régulière renseigne non seulement sur la **formation** des **clusters** (par la concentration de liens forts à l'intérieur de certaines sous-parties), mais aussi sur la **différenciation** progressive des liens faibles et des liens forts. Les résultats typiques montrent qu'au départ, toutes les $\omega_{i,j}$ se situent dans un intervalle homogène (souvent proche de zéro si la matrice est initialisée à un faible bruit), puis la distribution se "bimodalise" : certains liens demeurent à des niveaux faibles, tandis que d'autres s'élèvent, révélant l'émergence de groupes denses. Parfois, un suivi fin permet de repérer des **oscillations** récurrentes dans la force de certains liens, symptôme d'une configuration non stabilisée ou d'un paramétrage problématique (par exemple, un η trop grand ou un τ trop faible). Dans ces cas, le *monitoring* constitue un outil de **diagnostic** : si le réseau ne converge pas ou oscille sur une large amplitude, on peut ajuster les paramètres ou vérifier que la fonction de synergie $S(i,j)$ ne présente pas de discontinuité excessive.

Le simple **observationnel** des pondérations ne suffit toutefois pas à extraire automatiquement les **communautés** ni à fournir une répartition claire de l'espace. C'est la raison pour laquelle, dans des sections ultérieures (telles que la section 4.6.2), l'on recourra à des méthodes plus **systématiques** d'extraction de clusters ou de détection de structures. Néanmoins, l'examen visuel ou statistique régulier de la matrice $\omega_{i,j}(t)$ garde tout son intérêt en phase de développement, de **test** ou de **recherche de bugs**, puisqu'il permet de diagnostiquer l'évolution globale du SCN et de repérer s'il se dirige vers un état stable, un cycle, ou un brouillard de valeurs moyennes peu discriminantes. Cette première étape de *monitoring* jette ainsi les bases d'une analyse plus poussée : sans elle, on risquerait de recourir à des algorithmes de détection de communautés ou de hiérarchisation sans s'apercevoir que le réseau ne s'est pas encore stabilisé, ou qu'une oscillation mine l'interprétabilité des clusters.

4.6.1.2. Visualisations en Heatmap, Courbes de Liaisons, ou Graphes Dynamiques

Le **Synergistic Connection Network (SCN)** met en jeu un ensemble de **pondérations** $\omega_{i,j}(t)$ dont l'évolution permet de comprendre la genèse de **clusters** ou de communautés internes. Lorsque le nombre d'entités n reste modéré, la simple consultation de la matrice $\boldsymbol{\omega}(t)$ peut suffire, mais dès que l'on dépasse quelques centaines de nœuds, cette matrice devient difficile à interpréter si on se limite à un tableau numérique. Il est donc souvent essentiel d'adopter des **représentations visuelles** pour saisir le comportement du réseau, apprécier la montée ou la chute des liaisons $\omega_{i,j}(t)$ et identifier la formation progressive de blocs.

A. Heatmap (Matrice Colorée)

L'une des techniques les plus directes pour **visualiser** la matrice $\boldsymbol{\omega}(t)$ consiste à en dresser une **carte colorée**. Les lignes représentent les entités $\mathcal{E}_i$ et les colonnes $\mathcal{E}_j$, tandis que chaque cellule (i,j) est teintée selon la valeur de $\omega_{i,j}(t)$. Une **échelle de couleurs** (du clair au foncé, par exemple) traduit l'intensité du lien : plus la cellule est saturée, plus la pondération est élevée. Il est possible de générer cette carte pour différentes itérations t, de manière à constituer une séquence ou un "film" illustrant l'émergence ou l'extinction de régions de forte connectivité. Cette approche convient bien lorsque le **nombre** d'entités n demeure dans des limites raisonnables, comme quelques centaines de nœuds. On observe alors la formation de **blocs** compacts correspondant à des **clusters**, identifiables par des carrés colorés le long de la diagonale (ou, après réordonnancement, dans d'autres zones de la matrice). Lorsque n devient plus grand, il est courant

de ne conserver qu'un **échantillonnage** ou de n'afficher que les liaisons dépassant un certain seuil. La heatmap reste alors un outil macroscopique, utile pour distinguer rapidement un état quasi homogène (aucun bloc net), une structure bipartite, ou plusieurs communautés distinctes.

B. Courbes de Liaisons

Une autre méthode consiste à **sélectionner** un sous-ensemble de liens $\omega_{i,j}(t)$ et à tracer leur valeur au cours du temps. Cette démarche permet un suivi plus **fin** d'un petit nombre de paires qui peuvent être jugées représentatives : certains liens présumés intra-cluster, d'autres inter-cluster, ou encore des liaisons ayant un rôle spécifique. On produit alors, pour chaque lien (i,j) choisi, une courbe indiquant $\omega_{i,j}(t)$ en fonction de t.

Cette représentation est particulièrement instructive pour **comprendre** la dynamique de montée ou de descente d'un lien donné. Un lien **intra-cluster** se met à croître rapidement jusqu'à un plateau, alors qu'un lien **inter-cluster** peut demeurer faible ou décroître progressivement. Parfois, on décèle des **oscillations** si $\omega_{i,j}(t)$ monte puis redescend en boucle, symptôme d'un déséquilibre dans la règle de mise à jour ou d'un paramètre mal calibré (taux d'apprentissage η trop grand, par exemple). Les *courbes de liaisons* offrent donc un moyen succinct de diagnostiquer la formation de blocs ou les difficultés de convergence, sans afficher la totalité de la matrice.

C. Graphes Dynamiques

Il est également possible de **représenter** le SCN comme un graphe dont les nœuds incarnent les entités $\{\mathcal{E}_i\}$, et où chaque arête (i,j) est dotée d'un poids $\omega_{i,j}(t)$. On peut alors faire évoluer l'épaisseur de l'arête ou sa couleur en fonction de $\omega_{i,j}(t)$, voire employer un algorithme de placement (layout) dynamique pour rapprocher les nœuds fortement reliés et éloigner ceux qui sont faiblement connectés.
Cette **visualisation** en "graphe dynamique" offre une vision très intuitive : on assiste littéralement au rassemblement progressif des nœuds en sous-ensembles denses. Un nœud peut être intégré simultanément à plusieurs blocs s'il dispose de liaisons importantes réparties, ou bien on voit apparaître des "communautés" distinctes dans différentes régions du plan si le layout *force-directed* est utilisé. Lorsque les pondérations $\omega_{i,j}(t)$ convergent, le graphe se fige en clusters stables ; si des oscillations surviennent, les liens changent périodiquement de force, et le graphe ne stabilise pas sa topologie.

Cette technique devient toutefois coûteuse pour de grands n. On doit le plus souvent filtrer les liaisons (ne garder que celles au-dessus d'un seuil, ou les k plus fortes par nœud) afin de limiter l'encombrement visuel. Dans un DSL de taille moyenne, l'animation du graphe dynamique constitue un atout pédagogique puissant pour visualiser la **coalescence** de communautés internes.

D. Intérêt pour l'Analyse de la Dynamique

Le choix de la **représentation** visuelle dépend du nombre d'entités, de la granularité qu'on veut observer et de la possibilité de filtrer ou non les liens. Les **heatmaps** indiquent la globalité de la matrice, les **courbes de liaisons** donnent un portrait plus détaillé de certaines paires, et les **graphes dynamiques** procurent une image structurale et esthétique du réseau en mutation. Toutes ces méthodes servent un même objectif : **appréhender** la progression du SCN, **détecter** les moments critiques (croissance explosive de certaines pondérations, effondrement d'autres) et **repérer** des régimes d'oscillation ou de convergence.

Dans la suite (section 4.6.2), des outils plus formels d'**identification automatique** des clusters seront présentés. Les approches de visualisation décrites ici restent néanmoins un complément précieux pour la **compréhension** et la **validation** empirique de ce qui se produit dans un DSL. Elles confèrent une dimension exploratoire indispensable pour ajuster les hyperparamètres (η, τ) ou évaluer la pertinence de la fonction de synergie $S(i,j)$. Elles constituent en somme un pont entre l'analyse intuitive (repérer des blocs colorés, des arcs épais) et l'**extraction** algorithmique de structures.

4.6.2.1. Seuil sur $\omega_{i,j}$: si > θ, on considère l'arête comme "active" → composantes connexes

Le recours à un **seuil** θ appliqué aux pondérations $\omega_{i,j}$ représente l'une des stratégies les plus élémentaires pour **extraire** des **clusters** au sein d'un **Synergistic Connection Network (SCN)**. L'idée consiste à convertir la matrice $\{\omega_{i,j}\}$ en un graphe binaire en imposant une valeur critique θ : toute liaison (i,j) dont $\omega_{i,j}$ dépasse θ est qualifiée de "active" (et prend la valeur 1), tandis que les autres sont ramenées à 0. Cette binarisation aboutit à un réseau non pondéré où l'on peut aisément repérer des **composantes** connexes, chacune reflétant un **cluster** potentiel.

A. Définition du seuil et matrice binaire

La règle de transformation repose sur la relation

$$A_{i,j} = \begin{cases} 1 & \text{si } \omega_{i,j} > \theta, \\ 0 & \text{sinon.} \end{cases}$$

Ce passage d'une représentation pondérée $\omega_{i,j} \in \mathbb{R}^+$ à une **matrice** binaire **A** simplifie considérablement l'analyse. Dès qu'une liaison franchit la **barre** θ, elle est considérée comme un **arc actif** dans le graphe, sans plus distinguer les nuances de valeurs au-dessus de θ. Ce procédé reste très intuitif : seules les connexions jugées "fortes" ou "significatives" sont conservées, et on écarte la multitude de liens résiduels dont le poids, souvent faible, pourrait brouiller la lisibilité de la structure.

B. Rôle de θ et formation des clusters

Ce paramètre θ joue un rôle crucial, car il influe directement sur la **taille** et le **nombre** de composantes qu'on verra émerger dans le graphe binaire. Si l'on choisit un seuil trop élevé, la plupart des $\omega_{i,j}$ seront en deçà de θ ; on risque alors d'obtenir de nombreuses petites composantes, voire des nœuds totalement isolés. Au contraire, un seuil trop faible a pour effet de conserver trop de liaisons, transformant le graphe en un réseau quasi complet qui ne laisse apparaître aucune partition claire. La distribution des valeurs $\{\omega_{i,j}\}$ peut guider l'ajustement de θ : on peut par exemple fixer θ au percentile 80 ou 90 % des valeurs non-nulles, ou bien étudier l'histogramme pour repérer un coude permettant de trancher entre des liaisons fortes et des liaisons faibles.

C. Détection des composantes connexes

Une fois la matrice binaire établie, le **réseau** se conçoit comme un graphe simple où $\omega_{i,j} \in \{0,1\}$. L'extraction des **composantes** connexes devient alors un problème classique en théorie des graphes. On peut recourir à un algorithme de parcours en profondeur (*Depth-First Search*) ou en

largeur (*Breadth-First Search*) : pour chaque nœud, on visite les nœuds qui lui sont reliés par des arêtes actives, puis les nœuds reliés à ceux-ci, et ainsi de suite. L'ensemble de nœuds atteints de la sorte forme une **composante**, souvent qualifiée de cluster dans ce contexte d'auto-organisation. Il suffit alors de relancer cette procédure pour chaque nœud non encore visité afin de recenser la totalité des composantes. La **cohésion** interne d'une composante reflète la densité d'arêtes ayant dépassé le seuil θ. Ainsi, si θ est bien choisi, on obtient en général une partition reflétant très fidèlement les groupes de nœuds fortement reliés (intra-cluster), tout en isolant les liens inter-cluster jugés insuffisants.

D. Avantages et limites

La principale **force** de cette méthode réside dans sa **simplicité** et sa **transparence** : il suffit de couper tous les liens en deçà d'un certain poids. Les *clusters* détectés comme composantes connexes sont alors très lisibles, ne requièrent pas de calcul sophistiqué (un simple algorithme DFS ou BFS), et assurent une séparation nette entre les liens censés traduire une véritable affinité et ceux considérés comme parasites ou négligeables. Toutefois, cette binarisation implique qu'on ne nuance plus les liaisons supérieures au seuil : un lien à 0,95 est traité de la même manière qu'un lien à 0,55, alors que leur intensité réelle peut différer de manière significative. En outre, l'existence d'un **unique** paramètre θ rend la solution très sensible à ce choix : si ce seuil n'est pas adapté à la distribution réelle des $\omega_{i,j}$, on risque soit un trop grand nombre de micro-clusters, soit un unique bloc, ce qui n'est pas nécessairement conforme à la structure sous-jacente du SCN.

E. Conclusion

La méthode fondée sur un **seuil** θ appliqué aux pondérations $\omega_{i,j}$ constitue l'un des procédés les plus rudimentaires et rapides pour **identifier** des **clusters** dans un SCN. L'on retire tous les liens dont la valeur n'atteint pas ce niveau critique, puis on recherche les **composantes** connexes du graphe binaire obtenu. Cette approche révèle souvent la partition macroscopique que la dynamique d'auto-organisation cherchait à faire émerger, à condition de sélectionner judicieusement le seuil θ. Malgré l'absence de prise en compte des différences de poids au-dessus de θ, cette technique assure une lecture intuitive de l'organisation en groupes fortement soudés. Lorsque l'on désire un partitionnement plus *graduel*, il reste possible de se tourner vers des **méthodes pondérées** ou vers des indices de **modularité** (section 4.6.2.2) qui exploitent toute la richesse des valeurs $\omega_{i,j}$.

4.6.2.2. Mesures de Modularity (Inspirées des Graphes) ou "Community Detection" dans le SCN

Les **méthodes** décrites précédemment (en particulier le recours à un **seuil** θ pour binariser la matrice $\omega_{i,j}$) constituent des approches rapides pour **délimiter** des blocs dans un **Synergistic Connection Network** (*SCN*). Cependant, elles peinent parfois à exploiter la **graduation** subtile des valeurs $\omega_{i,j}$. Dès lors que l'on souhaite distinguer des liens à 0.95 de ceux à 0.55, il est naturel de se tourner vers des **outils** de *community detection* issus de la **théorie des graphes pondérés**. Cette famille d'algorithmes recourt souvent à une **mesure de modularité**, initialement conçue pour des graphes non pondérés, mais aisément étendue à des poids réels.

L'idée générale est de **maximiser** un indicateur, la **modularité** Q, qui évalue dans quelle mesure une partition en communautés s'écarte d'une configuration purement aléatoire. Appliquée à un

SCN, où $\{\omega_{i,j}\}$ traduit la **synergie** entre entités, la modularité fournit un critère solide pour **valider** la cohésion de chaque bloc interne tout en préservant l'hétérogénéité des liens.

A. Rappel de la Notion de Modularity dans un Graphe Pondéré

Dans un graphe non pondéré, la modularité (Q) se définit par le formalisme de Newman-Girvan. On compare la fraction de liens **intra-communautés** réelle d'une partition à la fraction **attendue** si la distribution des degrés était maintenue de façon aléatoire. En notation succincte, pour un graphe non pondéré, on a

$$Q = \frac{1}{2m}\sum_{i,j}\left[A_{i,j} - \frac{k_i\,k_j}{2m}\right]\delta(c_i, c_j),$$

où $A_{i,j}$ décrit la matrice d'adjacence binaire ($A_{i,j} = 1$ s'il existe une arête, 0 sinon), m est le nombre total d'arêtes, et k_i le degré du nœud i. Le terme $\delta(c_i, c_j)$ vaut 1 si les nœuds i et j appartiennent à la **même** communauté, 0 dans le cas contraire.

Dans le cadre **pondéré** pertinent pour un **SCN**, on remplace $\{A_{i,j}\}$ par $\{\omega_{i,j}\}$, et on définit le degré pondéré comme $d_i = \sum_j \omega_{i,j}$. La somme totale de poids W s'identifie alors à la moitié de $\sum_{i,j}\omega_{i,j}$ pour un graphe non orienté. La **modularité** prend la forme

$$Q = \frac{1}{2W}\sum_{i,j}\left[\omega_{i,j} - \frac{d_i\,d_j}{2W}\right]\delta(c_i, c_j).$$

Le raisonnement demeure identique : on quantifie la somme des poids "intra-communauté" qui excède la somme que l'on obtiendrait dans un régime aléatoire respectant les degrés $\{d_i\}$.

B. Application dans un Synergistic Connection Network

Dans un **SCN**, chaque entité $\mathcal{E}_i$ possède une somme pondérée $d_i = \sum_j \omega_{i,j}$. La totalité des poids est $W = 1/2\sum_{i,j}\omega_{i,j}$. Considérer $\omega_{i,j}$ comme la matrice de poids d'un *graphe pondéré* rend possible l'emploi direct des algorithmes de *community detection* tels que Louvain, Leiden ou d'autres techniques spectrales ou basées sur la betweenness.

Leur but est de **maximiser** la modularité Q. Une "communauté" au sens modulaire se conçoit donc comme un ensemble de nœuds (entités) dont les liens internes $\omega_{i,j}$ sont significativement supérieurs à ce que proposerait un modèle d'attachement aléatoire. Intuitivement, si des entités forment un cluster dans le SCN, on doit y constater un renforcement local $\{\omega_{i,j}\}$, ce qui se traduira par un niveau élevé de modularité intra-communauté.

C. Avantages pour le SCN

Comparer cette méthode de *community detection* à la binarisation via un **seuil** θ souligne deux atouts majeurs. D'abord, la modularité prend **pleinement** en compte les variations de poids entre différents liens, et ne les réduit pas à un simple 0/1. Ensuite, de nombreux **algorithmes** éprouvés – comme Louvain ou Leiden – sont disponibles dans des bibliothèques spécialisées (NetworkX, igraph, etc.), avec une **implémentation** optimisée même pour des graphes de grande taille.

Si l'on souhaite confirmer la cohérence d'un découpage donné (par exemple, si on a une partition candidate trouvée par un autre moyen), on peut en calculer la **modularité** Q. Plus la valeur de Q s'approche de 1, plus la structuration en communautés est considérée "prononcée". Dans les cas extrêmes, un Q très proche de 0 ou négatif indique une absence de véritables communautés au sens de la pondération interne.

D. Limites et Alternatives

La **modularité** est victime d'un phénomène connu sous le nom de "résolution limit", qui tend à faire émerger des communautés parfois plus grandes que la "réalité" de terrain ou, au contraire, à rater certains clusters de taille réduite. D'autres métriques (comme la *partition density*, la *surprise*, ou des méthodes de *infomap*) peuvent pallier partiellement ces écueils, mais la modularité demeure un **standard** pour la *community detection* dans de nombreux scénarios.

En sus, on peut envisager des **méthodes** alternatives pour repérer des blocs pondérés : par exemple, un **algorithme spectral** construit la Laplacienne du graphe $\mathbf{L} = \mathbf{D} - \mathbf{W}$, où $\mathbf{W}$ est la matrice $\{\omega_{i,j}\}$ et $\mathbf{D}$ la matrice diagonale des degrés pondérés. Les *vecteurs propres* associés aux plus petites valeurs propres de $\mathbf{L}$ permettent d'identifier, via un *k*-means, des ensembles fortement connectés. D'autres algorithmes, comme *Girvan–Newman* adapté aux poids, suppriment itérativement les liens de forte betweenness jusqu'à l'apparition de groupes.

E. Conclusion

Les **mesures de modularité**, et plus largement les approches de **community detection** pondérées, constituent une avancée majeure pour l'**identification** de **clusters** dans un SCN. Contrairement à la simple binarisation par θ, on conserve la **graduation** des pondérations $\omega_{i,j}$. Cette nuance se révèle décisive pour des scénarios où la distance entre 0.95 et 0.60 importe, et où l'on ne veut pas se contenter d'un coup de ciseau tranchant. Grâce à des algorithmes tels que Louvain ou Leiden, on peut tirer parti de la **structuration** que la dynamique du DSL a créée, et faire émerger une partition en communautés reflétant véritablement la répartition des poids dans le réseau. L'étude de la modularité Q donne alors un **indicateur** quantitatif du caractère "naturel" ou "forcé" de la segmentation, tout en s'appuyant sur la logique de la pondération globale $\{\omega_{i,j}\}$.

4.6.2.3. Lien avec le Clustering "Offline" (k-means, DBSCAN), mais ici c'est issu d'une Dynamique Interne

Le **clustering** en tant que méthode de segmentation de données se rencontre fréquemment sous forme d'algorithmes "**offline**" tels que **k-means**, **DBSCAN** ou encore l'approche **agglomérative**. Ces techniques s'appliquent habituellement à un **ensemble** de points $\{x_i\} \subset \mathbb{R}^d$, soumis à une distance explicite (euclidienne, cosinus, densité, etc.), dans le but de **découper** l'ensemble en blocs suffisamment cohérents pour un critère global.

Dans un **Synergistic Connection Network (SCN)**, le repérage des **clusters** repose sur une **dynamique interne** : la mise à jour itérative de $\omega_{i,j}(t)$ construit progressivement une structure de poids, révélant des **liaisons** fortes entre entités $\mathcal{E}_i, \mathcal{E}_j$. Il est alors légitime de se demander dans quelle mesure ces approches de clustering classiques recoupent, ou se différencient, des blocs obtenus par la **lecture** de la matrice $\{\omega_{i,j}\}$ (comme décrit en 4.6.2.1 ou 4.6.2.2).

A. Rappel : k-means, DBSCAN et Clustering Offline

Lorsque l'on applique **k-means** à un ensemble de points $\{x_1, \dots, x_n\} \subset \mathbb{R}^d$, l'algorithme procède par minimisation d'une fonction d'énergie, généralement la somme intra-cluster des distances au centroïde. La taille k de la partition est fixée au préalable, et le résultat final dépend de la position initiale des centroïdes et de la distribution des données. **DBSCAN**, à l'inverse, se fonde sur une notion de **densité** (rayon ε et minPts) afin de regrouper les points "denses" et laisser les autres comme du "bruit".

Ces algorithmes sont dits "**offline**" : un **lot** de données est fourni, et l'on cherche le partitionnement optimal en se basant sur une **distance** statique. Toute correction (changement de paramètres) exige de relancer l'algorithme.

B. Dynamique Interne des Pondérations dans un SCN

Dans un **SCN**, au contraire, les entités $\mathcal{E}_i$ ne sont pas placées dans un espace fixe ; elles se relient par des pondérations $\omega_{i,j}(t)$ qui évoluent au fil d'une **règle** de mise à jour dépendant de la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$. Le SCN ne procède pas par minimisation explicite d'un critère global comme k-means ; il s'appuie plutôt sur un **processus local** qui renforce ou atténue chaque liaison. Les *clusters* émergent alors naturellement, et l'on peut distinguer :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta \left[S(i,j) - \tau\, \omega_{i,j}(t) \right].$$

Les *clusters* sont donc un **résultat** de l'**auto-organisation** et non l'objet d'un algorithme de partition. Il est possible de considérer qu'une fois la matrice $\{\omega_{i,j}\}$ stabilisée (ou lue à un instant donné), on détient un "**graph pondéré**" dont on peut extraire des communautés via différentes méthodes (seuil, modularité, etc.).

C. Possibilité de Relier l'Approche "Offline" et "Online"

Un **DSL** présente d'autres subtilités : l'on peut, de temps à autre, prélever la **matrice** $\{\omega_{i,j}(t)\}$ pour en faire un **traitement** "offline". Par exemple, on peut procéder ainsi :

10. Considérer les vecteurs $\boldsymbol{\omega}_i \in \mathbb{R}^n$ définis par

$$\boldsymbol{\omega}_i = \left(\omega_{i,1}, \omega_{i,2}, \dots, \omega_{i,n}\right).$$

    On obtient ainsi un portrait complet de la manière dont la **synergie** s'exprime entre $\mathcal{E}_i$ et l'ensemble des autres entités.

11. Appliquer un **clustering** typique (ex. k-means) sur l'ensemble $\{\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_n\}$. Les entités qui possèdent un "profil de connexion" similaire (pondérations internes et externes proches) se retrouvent alors dans la même classe.

Cette technique revient à considérer que la *distance* entre deux entités $\mathcal{E}_i, \mathcal{E}_j$ peut être mesurée via une norme $\| \boldsymbol{\omega}_i - \boldsymbol{\omega}_j \|$. Par ce biais, l'on fait un pont entre la **dynamique interne** (qui a fait croître ou décroître les $\omega_{i,j}$) et un **algorithme** de segmentation standard (k-means, DBSCAN) appliqué "après coup".

D. Intérêt et Différences

Il existe un **intérêt** certain à comparer une **partition** issue de cette démarche (ou via la modularité sur le graph pondéré) à celle qu'on obtiendrait en prenant directement les **données brutes** de $\mathcal{E}_i$ (ex. leurs embeddings ou leurs attributs) et en appliquant k-means/DBSCAN. Si les deux partitions coïncident, on peut conclure que la **fonction** de synergie $S(i,j)$ et la **dynamique** du SCN n'ont fait que reproduire la structure spatiale originale. S'il y a un écart notable, cela indique que la logique interne du SCN (éventuellement imprégnée de symbolique, ou d'une synergie non triviale) a façonné une structuration différente, possiblement plus riche.

La différence essentielle est que dans le **clustering offline**, on cherche à **minimiser** explicitement un critère global (somme des distances intra-cluster, densité, etc.), tandis qu'un **SCN** exécute un **processus local** d'**auto-organisation**. Ce processus peut faire intervenir différents paramètres (symboliques ou sub-symboliques), permettant d'autres formes de "convergence" qu'une simple structure euclidienne dans $\mathbb{R}^d$.

Ainsi, si un utilisateur se demande : "Ne pourrais-je pas simplement appliquer k-means (ou DBSCAN) sur mes données plutôt que de passer par la mise à jour des $\omega_{i,j}$?" La réponse tient en la **finalité** : un **DSL** permet de gérer un flux évolutif, une synergie multimodale, ou un couplage symbolique–sub-symbolique. Les **clusters** ne sont pas seulement un partitionnement statique ; ils résultent d'un **processus** vivant, sensible aux interactions internes (renforcements/désactivations).

E. Conclusion

Il existe un **parallèle** entre la volonté de décomposer un ensemble de points en **clusters** (via k-means, DBSCAN) et la recherche de **communautés** à l'intérieur d'un **SCN**. Les deux aboutissent à des **groupes** d'entités. Cependant, dans un clustering traditionnel "offline", on part d'un dataset statique et on applique un **critère** fixé (variance, densité, etc.), alors que dans un SCN, les **liens** $\omega_{i,j}$ proviennent d'une **dynamique** auto-organisée. Les algorithmes de partition basés sur $\{\omega_{i,j}\}$ (seuil, modularité, community detection) tirent avantage d'un "**réseau** pondéré" engendré par la synergie, et non d'une distance spatiale imposée a priori.

Un **SCN** peut donc être vu comme une solution "**online**" ou "dynamique", là où k-means/DBSCAN constituent des méthodes "**offline**" fixes. Les deux approches demeurent complémentaires : il est possible de confronter les **résultats** (partitions) pour mieux comprendre l'agencement de ses données, ou même d'adopter un schéma hybride (ex. utiliser k-means pour initialiser ω, puis laisser la règle d'auto-organisation opérer). Cette **interaction** entre la dynamique interne du SCN et les techniques de clustering classiques illustre la souplesse du DSL, capable de revisiter ou d'étendre la notion même de partition au sein d'un réseau adaptatif.

4.6.3. Évolutions dans le Temps

Une fois les **clusters** identifiés (4.6.2), il est également pertinent de **suivre** leur **évolution** au fil des itérations ou des mises à jour (particulièrement si $S(i,j)$ n'est pas figé). Dans un **SCN** (Synergistic Connection Network), les changements de pondérations $\omega_{i,j}$ peuvent amener un cluster à **apparaître**, à **fusionner** avec un autre, ou encore à se **scinder** en deux ou plusieurs sous-groupes. L'observation de ces **dynamiques** se révèle cruciale pour comprendre la façon dont le réseau

s'auto-organise ou se réorganise lorsque les entités, les synergies ou les paramètres du modèle varient.

4.6.3.1. Les Clusters Peuvent Apparaître, Fusionner, se Scinder

Les **clusters** formés au sein d'un **Synergistic Connection Network (SCN)** ne constituent pas nécessairement des entités fixes et immuables. Sous l'effet de la **dynamique** de mise à jour des pondérations $\omega_{i,j}(t)$ et de l'évolution potentielle de la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ (dans un scénario de données ou de règles changeantes), ces regroupements peuvent se **créer**, se **réunifier** ou se **dissocier** au fil du temps.

A. Apparition de nouveaux clusters

Un groupe restreint d'entités, initialement dispersées au sein du SCN, est susceptible de voir leurs liaisons $\omega_{i,j}$ progresser de manière conjointe. Les liens intra-groupe s'élèvent peu à peu, se stabilisant au-dessus d'un certain **niveau**. À mesure que ces pondérations franchissent un seuil critique (éventuellement implicite) ou qu'elles surpassent nettement les liens externes, ce **groupe** se distingue nettement du reste du réseau. Un nouveau **cluster** "apparaît" alors, témoignant de la coalescence progressive de ses membres autour d'affinités fortes.
Sur le plan mathématique, si les poids $\{\omega_{i,j}\}$ pour (i,j) dans un certain sous-ensemble $\mathcal{C}$ atteignent des valeurs significatives, alors les entités $\mathcal{E}_i \in \mathcal{C}$ forment un bloc cohésif. Il suffit par exemple qu'à l'itération t les liaisons $\omega_{i,j}(t)$ dépassent successivement un seuil θ, rendant évident le fait que $\{i,j\}$ forment désormais un **noyau** ou qu'un triplet se renforce.

B. Fusion de clusters préexistants

Il arrive également que deux ou plusieurs **communautés** déjà consolidées se rapprochent l'une de l'autre. Dans ce cas, un sous-ensemble d'entités appartenant à deux blocs différents se met à voir ses pondérations inter-blocs croître de manière notable. Les liens $\omega_{i,j}$ entre ces deux agrégats, autrefois faibles, s'intensifient à la suite de modifications dans la **synergie** $S(i,j)$.

Lorsqu'un nombre suffisant de liaisons entre ces groupes dépasse un niveau appréciable, le système peut converger vers une situation où les deux ensembles précédents ne forment plus qu'un seul **cluster** fusionné. Sur le plan **graphique**, on interprète la scène comme deux composantes qui se **rejoignent**, des arêtes inter-blocs qui se renforcent, et une nouvelle macro-communauté qui émerge.

C. Scission d'un cluster en plusieurs

Inversement, un **cluster** jusque-là stable peut se **scinder** si, dans sa structure interne, certaines liaisons se mettent à décroître ou si un sous-groupe d'entités voit ses pondérations internes s'intensifier au détriment des liens avec le reste. Le sous-groupe finit par se "détacher" du noyau originel.

Sur un plan formel, on suppose qu'un cluster $\mathcal{C}$ se fragmente en $\mathcal{C}_1 \cup \mathcal{C}_2 = \mathcal{C}$ parce que les valeurs $\{\omega_{i,j}\}_{i\in\mathcal{C}_1, j\in\mathcal{C}_2}$ baissent progressivement, tandis que les pondérations $\{\omega_{i,j}\}_{(i,j)\in\mathcal{C}_1^2}$ demeurent

élevées. La discontinuité se révèle lorsqu'on observe la formation d'un véritable sous-cluster $\mathcal{C}_1$ qui ne conserve presque plus d'attaches fortes avec $\mathcal{C}_2$.

D. Causes et conditions de ces évolutions

Les **transformations** qui mènent à l'apparition, la fusion ou la scission de clusters peuvent s'expliquer par plusieurs facteurs. L'évolution de la **synergie** $S(i,j)$ constitue une première cause : dans un **DSL** multimodal ou symbolique, un changement d'environnement ou la mise à jour des caractéristiques d'une entité entraînent une redistribution de valeurs de similarité ou de compatibilité, modifiant les $\{\omega_{i,j}\}$. Les **paramètres** internes (le taux d'apprentissage η, le coefficient τ, ou encore l'ajout d'une inhibition latérale γ) peuvent aussi favoriser la ségrégation en petits blocs ou, au contraire, la coalescence en ensembles plus vastes. Enfin, des **perturbations** volontaires (recuit, injection de bruit) peuvent redéfinir la topologie du SCN et engendrer de nouvelles partitions.

E. Observation empirique et diagnostic

Sur le plan **pratique**, on suit la matrice $\{\omega_{i,j}(t)\}$ (section 4.6.1) ou on en extrait des communautés (section 4.6.2). À chaque itération ou après un certain nombre de pas, on compare la partition courante à la partition précédente, ce qui met en évidence l'apparition ou la fusion de blocs, la fragmentation d'une communauté, etc. Des mesures telles que la **NMI** (Normalized Mutual Information) ou l'indice de Rand offrent un moyen quantitatif de comparer deux partitions successives. On peut ainsi repérer les instants où un cluster a "**disparu**" (scission) ou où deux communautés se sont unies.

Cette **observation** permet également de comprendre si le **SCN** se stabilise ou maintient un régime oscillatoire. Lorsqu'un cluster apparaît mais se délite quelques itérations plus tard, l'auto-organisation demeure dans un état transitoire, suggérant qu'il n'existe pas encore d'équilibre.

Conclusion (4.6.3.1)

Les **clusters** au sein d'un **Synergistic Connection Network** ne sont pas fixes, mais reflètent la **dynamique** interne de la mise à jour $\Delta\omega_{i,j}$. Ils peuvent naître de la consolidation de quelques entités jusque-là dispersées, se **fusionner** en un bloc unique lorsque leurs sous-ensembles développent des liaisons suffisamment fortes, ou se **scinder** lorsque certaines liaisons internes s'affaiblissent. Cette plasticité témoigne de la nature adaptative d'un **DSL**, exposé à des modifications de la synergie ou des paramètres du réseau. Les sections ultérieures poursuivront cette analyse en considérant comment suivre et "tracker" ces variations de communautés (section 4.6.3.2), afin de les consigner ou de les exploiter dans des scénarios applicatifs.

4.6.3.2. Notation d'un "Cluster Tracking" : Indexer les Entités qui Changent de Communauté

La dynamique d'un **Synergistic Connection Network (SCN)** peut conduire à des **transformations** répétées de la structure de **clusters** (ou communautés) : un groupe d'entités peut naître, deux communautés peuvent fusionner ou, au contraire, un bloc jusqu'alors soudé se scinder en plusieurs parties. Dans l'optique d'analyser un tel système en évolution, il est souvent crucial d'exploiter un mécanisme de **cluster tracking**, c'est-à-dire un dispositif consignant dans quelle

communauté chaque entité se situe à divers instants. Ce mécanisme autorise un suivi temporel des recompositions et, en cas de changements, il rend possible l'évaluation du degré d'instabilité ou de transition dans le SCN.

A. Principe Général du "Cluster Tracking"

Il est fréquent de ne pas recalculer la partition en communautés à chaque itération t, surtout si la dynamique est potentiellement rapide ou coûteuse. On définit plutôt une suite de **paliers** $\{t_0, t_1, \dots, t_K\}$ où l'on exécute une procédure d'**extraction** de clusters (comme décrit en 4.6.2). Chaque entité $\mathcal{E}_i$ reçoit un label de communauté $\text{clusterID}_i(t_k)$ pour chaque instant de référence t_k. La confrontation des partitions successives donne alors des informations précises : si une entité $\mathcal{E}_i$ change de label, on note que cette entité migre d'une communauté à une autre ; si un cluster se scinde, on repère dans la partition au temps t_{k+1} que le groupe qui s'appelait "C1" au temps t_k est maintenant réparti dans deux nouveaux labels.

B. Comparaison Formelle Entre Deux Partitions

Pour comparer la partition $\mathcal{P}(t_k)$ à la partition $\mathcal{P}(t_{k+1})$, il est souvent utile de recourir à des **indices** standard tels que l'**indice de Rand** ou la **NMI** (Normalized Mutual Information). Ces mesures quantifient la similarité entre deux partitions, de sorte qu'un indice élevé indique une partition presque inchangée alors qu'un indice bas reflète une fragmentation ou une fusion importante. Sur le plan **algorithmique**, on doit parfois résoudre un **matching** entre les labels des deux partitions, puisque le cluster 3 dans $\mathcal{P}(t_k)$ peut correspondre majoritairement au cluster 2 dans $\mathcal{P}(t_{k+1})$. Il se peut qu'un algorithme de *Hungarian matching* (ou un équivalent) soit utilisé pour trouver un mapping optimal, minimisant les désaccords entre les deux partitions.

C. Changements Possibles et Suivi dans le Temps

Les changements majeurs identifiés sont :

- **Apparition d'un nouveau cluster** : un sous-ensemble d'entités jusqu'alors éparpillées ou relevant d'autres communautés se regroupent en un bloc cohésif.
- **Fusion de clusters** : deux blocs établis voient leurs liens inter-communautés $\{\omega_{i,j}\}$ croître au point de se confondre en un unique cluster.
- **Scission** : un bloc relativement stable se fragmente en deux sous-blocs distincts, dont les liens internes respectifs se renforcent et se détachent de l'ensemble initial.
- **Migration partielle** : quelques entités se déplacent d'une communauté à une autre, sans que la communauté source ni la communauté cible ne disparaissent ou ne fusionnent.

La connaissance de ces changements successifs permet, par exemple, de constituer un **journal** retraçant l'historique de l'évolution du SCN. Cela se conçoit sous la forme d'une table où l'on indique à quel pas de temps $\mathcal{E}_i$ quitte un label pour un autre, ou sous forme de diagrammes qui représentent visuellement la "trajectoire" de chaque bloc dans un plan temporel, un peu à la manière d'un diagramme de Sankey.

D. Avantages et Limites

Le *cluster tracking* éclaire la **stabilité** du SCN en révélant quelles entités demeurent ancrées dans le même cluster sur une longue période et quelles entités se montrent plus volatiles. Cela facilite

un diagnostic de la **convergence** : si, itération après itération, la partition fluctue très peu, on en déduit une stabilisation. Inversement, si une proportion non négligeable de nœuds changent de label à chaque palier, on soupçonne soit des oscillations internes, soit une évolution rapide de la synergie $\{\omega_{i,j}\}$.

Sur le plan computationnel, le *cluster tracking* exige d'**exécuter** ou de **répéter** régulièrement un algorithme de détection de communautés (ou un seuil, ou une modularité). Cette opération ne doit pas nécessairement être effectuée à chaque itération : on peut opter pour un pas plus lâche, réduisant la charge. Il subsiste un flou potentiel lorsqu'une communauté se scinde en plusieurs morceaux de taille comparable : les algorithmes de matching entre labels peuvent alors décider arbitrairement d'un label pour la majorité, et on perd une correspondance limpide pour l'autre fragment.

Conclusion (4.6.3.2)

La mise en place d'un **mécanisme** de "**cluster tracking**" joue un rôle capital dans un **DSL évolutif**, où la matrice $\omega_{i,j}(t)$ et par conséquent la structure de groupes subissent de potentielles modifications à chaque étape. En consignant la partition $\mathcal{P}(t_k)$ à divers paliers, puis en comparant ces partitions entre elles, on identifie précisément les **migrations** d'entités d'un cluster à l'autre, les **fusions** et **scissions** de communautés. Cette démarche outrepasse la simple photographie à un instant donné : elle offre une vision **dynamique** et **longitudinale** du SCN, révélant non seulement le "qui est avec qui" final, mais aussi la façon dont on y parvient et les délais ou perturbations éventuels. Elle sert ainsi à détecter et à **documenter** le *turn-over* communautaire, informant de la robustesse, de la stabilité et de l'adaptabilité du réseau pondéré.

4.7. Exemples Concrets et Études de Cas

4.7.1. Petite Simulation Numérique

- 4.7.1.1. 10 entités sub-symboliques + 5 entités logiques, synergie artificielle ou semi-aléatoire.
- 4.7.1.2. Mise en œuvre de la dynamique $\omega_{i,j}(t+1)$. Courbes de convergence, émergence de 2 ou 3 clusters.
- 4.7.1.3. Discussion des paramètres : η, τ, γ (inhibition), etc.

4.7.2. Cas Robotique ou Multi-Agent

- 4.7.2.1. Rappel : flottes de robots, ex. 5 robots simulés, synergie = réussite conjointe de tâches.
- 4.7.2.2. Observations : un cluster se forme entre 3 robots complémentaires, les 2 autres se marginalisent ou se lient plus faiblement.
- 4.7.2.3. Visualisation dynamique (cycles, stabilisation).

4.7.3. Liens Pratiques

- 4.7.3.1. Ce qu'on pourrait coder en Python/C++ pour reproduire cette dynamique en quelques centaines de lignes.
- 4.7.3.2. Aperçu du "SCN Dashboard" : par ex. un outil de monitoring des $\omega_{i,j}$ pour détecter la formation des clusters en temps réel.

4.7. Exemples Concrets et Études de Cas

Les sections précédentes (4.5, 4.6) ont présenté la dynamique d'un **SCN** (Synergistic Connection Network) et les moyens d'observer/extraire les clusters. Pour aller plus loin, il est souvent instructif de **mettre en pratique** ces notions sur des **cas concrets** ou des **simulations**. Ce chapitre 4.7 illustre ainsi plusieurs **exemples** de configurations (sous forme de petites études de cas) où l'on voit clairement comment la synergie (artificielle, semi-aléatoire ou adaptée à une tâche) déclenche l'auto-organisation du réseau.

4.7.1. Petite Simulation Numérique

- 4.7.1.1. 10 entités sub-symboliques + 5 entités logiques, synergie artificielle ou semi-aléatoire.
- 4.7.1.2. Mise en œuvre de la dynamique $\omega_{i,j}(t+1)$. Courbes de convergence, émergence de 2 ou 3 clusters.
- 4.7.1.3. Discussion des paramètres : η, τ, γ (inhibition), etc.

4.7.2. Cas Robotique ou Multi-Agent

- 4.7.2.1. Rappel : flottes de robots, ex. 5 robots simulés, synergie = réussite conjointe de tâches.
- 4.7.2.2. Observations : un cluster se forme entre 3 robots complémentaires, les 2 autres se marginalisent ou se lient plus faiblement.
- 4.7.2.3. Visualisation dynamique (cycles, stabilisation).

4.7.3. Liens Pratiques

- 4.7.3.1. Ce qu'on pourrait coder en Python/C++ pour reproduire cette dynamique en quelques centaines de lignes.
- 4.7.3.2. Aperçu du "SCN Dashboard" : par ex. un outil de monitoring des $\omega_{i,j}$ pour détecter la formation des clusters en temps réel.

4.7.1.1. 10 Entités Sub-Symboliques + 5 Entités Logiques, Synergie Artificielle ou Semi-Aléatoire

Il peut être instructif, pour illustrer les mécanismes d'un **Deep Synergy Learning (DSL)** à **taille modeste**, de constituer un petit jeu de données mélangeant plusieurs entités **sub-symboliques** et quelques entités **symboliques**. L'objectif est de mettre en évidence la **cohérence** qui peut émerger entre groupes de vecteurs (sous-cluster A ou B) et blocs logiques (règles, axiomes) au terme de la dynamique de mise à jour $\omega_{i,j}(t)$.

A. Configuration des Entités

L'expérimentation suppose la présence de dix entités **sub-symboliques** $\{\mathcal{E}_1, \dots, \mathcal{E}_{10}\}$, chacune décrite par un **embedding** en dimension $d = 5$. La génération de ces vecteurs peut s'effectuer selon un **tirage** aléatoire contrôlé, en faisant apparaître deux **sous-groupes** (A et B) plus proches intérieurement (typiquement, leurs centres de gravité se distinguent). L'on obtient ainsi une répartition semi-artificielle, de sorte que la *similarité* sub–sub prenne des valeurs significativement plus élevées (par exemple, autour de 0.7) dans chacun des deux sous-groupes, tandis qu'elle tombe plutôt autour de 0.2 pour des paires inter-groupes.

On introduit en complément cinq entités **logiques** $\{\mathcal{L}_1, \dots, \mathcal{L}_5\}$. Chacune incarne un **ensemble de règles** ou un **bloc de prédicats**, susceptibles d'être plus ou moins compatibles avec les caractéristiques de A ou B. On peut, par exemple, faire en sorte que $\{\mathcal{L}_1, \mathcal{L}_2\}$ correspondent à des axiomes particulièrement alignés sur le sous-cluster A, alors que $\{\mathcal{L}_3, \mathcal{L}_4\}$ trouvent une plus grande cohérence avec B, tandis que $\mathcal{L}_5$ demeure relativement **neutre**.

B. Génération de la Synergie

Le calcul de la synergie $S(i,j)$ peut se structurer selon une logique **hybride** (chapitre 3), c'est-à-dire :

$$S_{\text{hybrid}}(i,j) = \alpha\, S_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j) + (1-\alpha)\, S_{\text{sym}}(R_i, R_j),$$

où $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^5$ représentent des embeddings sub-symboliques et $\{R_i, R_j\}$ les blocs logiques associés. Pour deux entités sub-symboliques, on se limite à la **similarité** sub–sub (par exemple un cosinus normalisé ou un noyau gaussien), alors que pour deux entités **symboliques**, un **score** binaire (0 ou 1) ou trinitaire (0, 0.5, 1) quantifie la **compatibilité** des règles. Enfin, pour un couple sub–sym, l'on convient d'une formule adaptée (voir chap. 3.5) ou d'un paramètre β modérant l'influence d'un vecteur sub-symbolique sur un bloc logique.

La distribution semi-aléatoire est paramétrée de telle sorte que, dans chaque sous-groupe sub-symbolique A ou B, on obtienne des similarités sub–sub moyennes de 0.6–0.8, tandis que les paires inter-groupes se maintiennent à des valeurs plus faibles (0.1–0.3). Les entités logiques $\mathcal{L}_1, \mathcal{L}_2$ pourront ainsi être fortement cohérentes avec A ($S_{\text{sym}}(\mathcal{L}_1, \mathcal{E}_i) \approx 1$ si $\mathcal{E}_i \in A$), et moyennement ou faiblement cohérentes avec B, etc.

C. Lecture Attendue du SCN

Lorsque la **dynamique** de pondérations $\omega_{i,j}(t)$ se déroule (au sens de la mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \cdots$), il est prévisible de voir émerger au moins deux **communautés** majeures : la première associant les entités de **sous-cluster A** (car elles sont proches sub-symboliquement) et les entités logiques compatibles $\{\mathcal{L}_1, \mathcal{L}_2\}$, la seconde regroupant les entités de **sous-cluster B** liées à $\{\mathcal{L}_3, \mathcal{L}_4\}$. La cinquième entité logique $\mathcal{L}_5$, dotée d'un profil plus neutre, pourra :

12. Rejoindre l'un des deux groupes si sa compatibilité symbolique penche assez nettement dans ce sens,

13. Rester relativement isolée dans un micro-bloc, ou

14. Établir un pont faiblement pondéré entre les deux macros-clusters.

L'observation de la matrice $\{\omega_{i,j}(t)\}$ devrait mettre en évidence deux **blocs** de fortes pondérations internes, reflétant les deux sous-groupes sub–sym coalisés, et laisser à part une zone de liaisons plus ténues avec $\mathcal{L}_5$.

D. Reproductibilité et Variations

Cette configuration, étant essentiellement **semi-artificielle**, s'apprête bien à des expérimentations comparatives. En changeant la graine aléatoire qui distribue les embeddings sub-symboliques et en modifiant légèrement la logique de $\mathcal{L}_1, \ldots, \mathcal{L}_5$, on peut évaluer :

– La robustesse de la formation des mêmes blocs A–$\{\mathcal{L}_1, \mathcal{L}_2\}$ et B–$\{\mathcal{L}_3, \mathcal{L}_4\}$,

– L'existence de configurations multiples (multi-stabilité), dans lesquelles $\mathcal{L}_5$ intègre tel ou tel bloc en fonction des initialisations $\{\omega_{i,j}(0)\}$.

E. Extension et Généralisation

Ce **mini-prototype** avec dix entités sub-symboliques et cinq entités logiques peut aisément s'agrandir pour mieux faire ressortir le **phénomène**. En portant le nombre de sub-symboliques à 30 ou 50, réparties en deux ou trois sous-groupes artificiels, on intensifie la dimension "clustering sub–sub". Simultanément, on peut augmenter le nombre d'entités logiques en leur associant diverses compatibilités $\{R_i\}$ afin d'obtenir un scénario plus riche, tout en restant dans une échelle

accessible au *monitoring* (tracé de heatmaps, courbes, etc.). Les étapes subséquentes (sections 4.7.1.2, 4.7.1.3) détailleront comment **visualiser** l'évolution des $\omega_{i,j}(t)$ et **régler** des paramètres tels que η et τ pour mettre en évidence la trajectoire menant à la formation de **clusters** logiquement et statistiquement cohérents.

Conclusion (4.7.1.1)

La configuration consistant en **10 entités sub-symboliques** (réparties en deux mini-groupes) et **5 entités logiques** (cohérentes avec l'un ou l'autre sous-groupe) constitue un **exemple minimal** permettant d'**illustrer** la synergie sub–sub, sym–sym et sub–sym. En **comparant** la **matrice** $\{\omega_{i,j}(t)\}$ avant et après la convergence, on s'attend à voir au moins deux grands **clusters** stabilisés, chacun fédérant un bloc sub-symbolique et les blocs logiques compatibles. Cette **simulation** démontre ainsi la manière dont un **SCN** parvient à un **regroupement** cohésif, appuyé sur une double logique sub-symbolique et symbolique, et fournit un terrain de jeu pour étudier plus avant la dynamique, les paramétrages, et la possibilité de comportements multi-stables.

4.7.1.2. Mise en Œuvre de la Dynamique $\omega_{i,j}(t+1)$. Courbes de Convergence, Émergence de 2 ou 3 Clusters

La configuration décrite en (4.7.1.1) – réunissant dix entités **sub-symboliques** et cinq entités **logiques** avec une **synergie** semi-aléatoire – constitue un cadre propice pour illustrer la **dynamique** interne d'un **Synergistic Connection Network (SCN)**. Après avoir fixé la structure des entités (embeddings, règles) et leur **synergie** (sub–sub, sym–sym, sub–sym), la prochaine étape consiste à **exécuter** l'algorithme de mise à jour $\omega_{i,j}(t+1)$. L'observation de la **convergence** (ou non-convergence) et la **visualisation** des pondérations au fil des itérations révèlent la formation potentielle de deux ou trois **clusters** stabilisés.

A. Choix de la Règle de Mise à Jour

Il s'agit ici de mettre en œuvre la **formulation additive**, telle que :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right],$$

où η est le **taux d'apprentissage**, τ la **décroissance** et $S(i,j)$ la **synergie** présumée fixe pour cet exemple (les valeurs issues du modèle sub-symbolique, symbolique ou mixte). Le système démarre avec $\omega_{i,j}(0) \approx 0$ (ou un bruit faible dans $[0,0.05]$). En option, une **inhibition compétitive** peut s'ajouter, du type

$$-\gamma \sum_{k \neq j} \omega_{i,k}\,(t),$$

afin de limiter la prolifération des liens, bien que l'on puisse initialement ne pas l'inclure (ou la fixer à un faible $\gamma \approx 0.01$).

B. Boucle de Simulation

La procédure la plus classique consiste à répéter l'**itération** suivante pour un nombre fixé $T_{\max}$ de pas de temps (par exemple 500). À chaque itération t, on évalue pour toutes les paires (i,j) :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right] - \gamma \sum_{k \neq j} \omega_{i,k}(t) \quad \text{(si l'inhibition latérale est activée),}$$

suivi éventuellement d'un **clipping** $\max\{0,\cdot\}$ si la convention exige que $\omega_{i,j} \geq 0$. Une **mise à jour synchrone** (calcul des $\omega_{i,j}(t+1)$ à partir de toutes les $\omega_{i,j}(t)$) est préférée ici pour la simplicité. Entre deux itérations, on peut **enregistrer** la matrice $\{\omega_{i,j}(t)\}$ ou ses indicateurs sommaires (moyennes, variances).

C. Courbes de Convergence et Émergence de Blocs

En début de simulation, la plupart des pondérations restent très faibles. Puis, à mesure que l'on répète la mise à jour, les **paires** (i,j) pour lesquelles la synergie $S(i,j)$ est élevée s'accroissent nettement, tandis que d'autres stagnent ou régressent. Après plusieurs dizaines ou centaines d'itérations, deux ou trois ensembles **cohérents** ont tendance à se distinguer. Si l'on a prévu :

- Deux **sous-groupes** sub-symboliques (A et B) ;
- Un lot de **règles** $\{\mathcal{L}_1, \ldots, \mathcal{L}_5\}$, dont certaines s'alignent sur A et d'autres sur B ;

alors, on observe qu'un groupe (A + logiques $\mathcal{L}_1, \mathcal{L}_2$) voit ses liens internes monter au-dessus d'un certain plateau (par exemple 0.5–0.7), formant ainsi un **cluster** bien défini, tandis que l'autre groupe (B + logiques $\mathcal{L}_3, \mathcal{L}_4$) évolue vers un bloc distinct d'intensités similaires, et la cinquième entité logique $\mathcal{L}_5$ peut se retrouver en marge ou intégrée faiblement à l'un des deux.

On peut **documenter** la convergence via plusieurs moyens :

> **Courbes** $\omega_{i,j}(t)$: on choisit quelques paires (i,j) caractéristiques (intra-groupe, inter-groupe) et on trace $\omega_{i,j}(t)$ vs. t. Les liaisons intra-groupe montrent souvent une augmentation plus ou moins rapide jusqu'à un palier stable, tandis que les liaisons inter-groupe demeurent faibles ou oscillent en deçà d'un seuil.
>
> **Heatmap** : on extrait régulièrement la matrice $\{\omega_{i,j}(t)\}$ et on la présente sous forme de *carte colorée*. Les "zones" correspondantes aux clusters apparaissent clairement, occupant des blocs plus saturés.
>
> **Algorithmes** de détection de communautés (4.6.2.2) : si l'on souhaite une lecture plus automatisée, on peut appliquer, à chaque instant notable (ou à la fin de la simulation), un critère de modularité ou un algorithme Louvain sur la matrice $\{\omega_{i,j}\}$. En pratique, cela confirmera l'émergence de deux ou trois groupes majeurs.

D. Deux ou Trois Clusters au Final

En théorie, l'expérience montre que l'on obtient souvent **deux** clusters correspondant aux regroupements A et B, chacun couplé aux entités logiques qui partagent leur synergie. Toutefois, des ajustements subtils de la synergie ou des paramètres (η, τ, γ) peuvent introduire :

> **Un troisième** bloc, si une portion de A ou B se détache sous l'influence de certaines logiques, ou si $\mathcal{L}_5$ forme un embryon de communauté autonome.

Un regroupement en un seul super-bloc si la différence sub–sym est insuffisante pour maintenir une réelle scission (par exemple τ trop faible, η trop élevé).

Dans une configuration "parfaite" (sous-cluster A resserré, B bien séparé, et logiques polarisées), la dynamique converge la plupart du temps vers **deux** agglomérations centrales, et éventuellement un "satellite" composé d'entités moins typées.

E. Conclusion

L'exécution concrète de la **règle de mise à jour** $\omega_{i,j}(t+1)$ dans l'exemple décrivant dix entités sub-symboliques + cinq logiques met en relief la **croissance** des liens intra-cluster et la **décroissance** ou la stagnation des liens inter-cluster, aboutissant en pratique à **2** ou **3 clusters** stables. Ces résultats se traduisent mathématiquement par la montée en plateau de $\omega_{i,j}$ pour les paires à **forte** synergie, la fixation d'un noyau **sub–sym**, et la marginalisation de certains nœuds restés en faible affinité. Les méthodes de *visualisation* et de *community detection* (sections 4.6.1 et 4.6.2) confirment l'**émergence** de blocs cohérents. Cela illustre précisément la vocation d'un **SCN** : faire apparaître de manière auto-organisée (et localement régulée) des **groupements** reflétant tant la dimension sub-symbolique (similarité vectorielle) que l'éventuelle cohérence symbolique (règles).

4.7.1.3. Discussion des Paramètres : η, τ, γ (Inhibition), etc.

Le **mini-exemple** présenté aux sections 4.7.1.1 et 4.7.1.2 a montré comment un **SCN** (Synergistic Connection Network) relativement modeste – rassemblant dix entités sub-symboliques et cinq entités logiques – se développe au travers d'une **dynamique** de mise à jour $\omega_{i,j}(t+1)$. L'expérience a simultanément mis en relief l'importance des **paramètres** η, τ et γ. Ces coefficients déterminent la **vitesse** à laquelle les pondérations $\omega_{i,j}$ se réajustent, la **hauteur** qu'elles peuvent atteindre avant saturation et l'**interaction compétitive** qui peut encourager chaque nœud à privilégier certains liens au détriment d'autres. Cet ensemble de réglages exerce une influence directe sur la **structure** finale : nombre de **clusters**, clarté de la partition, potentiel de bifurcation ou d'oscillation.

A. Rôle de η (Taux d'Apprentissage)

La **formulation additive** classique suppose :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right],$$

où η contrôle l'ampleur de la correction appliquée à la pondération $\omega_{i,j}(t)$ à chaque itération. La manière dont η influe sur la **vitesse** et la **stabilité** de la convergence est cruciale :

Un η **trop faible** peut rallonger considérablement la **phase transitoire** : la croissance ou la décroissance de $\omega_{i,j}$ demeurant minime à chaque pas, plusieurs centaines ou milliers d'itérations peuvent être nécessaires pour voir émerger des blocs cohérents.

Un η **trop élevé** peut au contraire causer des **oscillations** ou une divergence si la boucle de rétroaction devient excessive. La matrice $\omega_{i,j}(t)$ se met alors à "sauter" autour de la valeur d'équilibre sans s'y stabiliser, ou elle s'aventure vers des amplitudes anormales.

Un compromis pratique consiste à essayer des valeurs de $\eta \in [0.01,\ 0.1]$. Dans le **scénario** illustré (4.7.1.2), un $\eta = 0.05$ s'avère un point de départ correct permettant d'atteindre un **plateau** en quelques centaines d'itérations sans oscillations majeures.

B. Rôle de τ (Décroissance)

La mise à jour comporte un terme $\tau\, \omega_{i,j}(t)$ que l'on **soustrait** partiellement à la pondération. Cette "force de **décroissance**" empêche $\omega_{i,j}$ de croître indéfiniment en cas de synergie $S(i,j)$ positive. Dans un cadre stationnaire simple, on retrouve un **point fixe** $\omega^*_{i,j} = S(i,j)/\tau$. Ainsi, plus τ est grand, plus ce point d'équilibre demeure **bas**, et inversement.

Lorsque $\tau \approx 0$, le mécanisme se rapproche d'une croissance potentiellement illimitée, risquant des saturations hors de portée ou des oscillations.

Lorsque τ est élevé, la possibilité de liens forts exige que $S(i,j)$ soit vraiment substantiel. Autrement, $\omega^*_{i,j} \approx 0$ pour la plupart des paires.

Dans l'exemple, fixer $\tau = 1$ et $\eta = 0.05$ donne un produit $\eta\,\tau = 0.05$. Dans un schéma additif linéaire, cette configuration garantit généralement une **convergence** stable pour un large éventail de synergies. Pour des schémas plus sophistiqués (inhibition, non-linéarités), la stabilité peut nécessiter des conditions plus élaborées.

C. Rôle de γ (Inhibition Compétitive)

L'**inhibition latérale** (section 4.2.2.2) introduit un terme supplémentaire $-\gamma \sum_{k\neq j} \omega_{i,k}\,(t)$ dans la mise à jour $\omega_{i,j}(t+1)$. L'effet de cette composante est de **forcer** chaque nœud $\mathcal{E}_i$ à limiter le nombre ou la force totale de ses liaisons en compétition. Lorsque γ est nul, chaque nœud peut, en théorie, entretenir de multiples liens "moyennement élevés". Une inhibition plus **forte** ($\gamma \approx 0.05$) incite chaque nœud à se focaliser sur moins de liens, ou à "opter" pour un cluster plus net. Cela réduit les configurations mixtes et favorise l'apparition de **communautés** mutuellement exclusives, au risque de scinder des regroupements potentiellement plus larges si γ atteint un niveau trop élevé.

D. Autres Paramètres Pratiques

Outre η, τ et γ, d'autres choix peuvent impacter la dynamique :

Clipping $\omega_{i,j} \geq 0$. Il est fréquent d'imposer $\max\{0,\cdot\}$ pour éviter que certaines pondérations deviennent négatives, ce qui peut heurter l'interprétation de "lien fort".

Sparsification. On peut décider de conserver seulement les k liaisons les plus fortes par nœud, ou celles dépassant un certain seuil dynamique (voir 4.4.2.2), pour favoriser l'émergence de blocs distincts.

Bruitage ou "recuit" (4.4.3.2). Injecter un léger bruit aléatoire dans la mise à jour $\omega_{i,j}$ aide parfois à échapper à des minima locaux ou à surmonter des phénomènes de multi-stabilité non désirés.

E. Synthèse des Réglages dans l'Exemple

Dans la simulation illustrée en 4.7.1.2, des valeurs modérées $\eta = 0.05$, $\tau = 1.0$ et $\gamma \approx 0$ (ou $\gamma = 0.01$) produisent typiquement la formation de deux à trois **clusters** en moins de 300 itérations, s'accompagnant d'une stabilisation visible dans les courbes $\omega_{i,j}(t)$. En revanche, si η est trop fort (0.2 ou 0.3), le réseau peut osciller ou hésiter plus longtemps avant de trouver un arrangement stable. Si τ est trop petit (0.1 ou 0.2), beaucoup de liaisons gonflent rapidement et les clusters manquent de "relief" ; si τ est trop grand (> 2), la plupart des pondérations restent faibles et l'on ne parvient pas à dégager des blocs cohérents, sauf si $S(i,j)$ est très prononcé. L'ajout d'une inhibition γ plus substantielle (0.05 ou 0.1) va forcer le réseau à "choisir" de manière plus drastique, potentiellement conduisant à une partition plus scindée, avec des blocs plus purs.

Conclusion

Les **paramètres** η (taux d'apprentissage), τ (décroissance) et γ (inhibition) se révèlent fondamentaux pour **guider** l'auto-organisation dans un **DSL**. Leur dosage détermine la **vitesse** de convergence, la **précision** des clusters, l'apparition ou non d'oscillations, ou encore le degré d'exclusivité dans les liens (γ favorisant un "winner-takes-most"). Dans un exemple simple (dix entités sub-symboliques, cinq entités logiques), la bonne combinaison de η, τ, γ permet d'aboutir en quelques centaines d'itérations à une structure claire, typiquement deux (ou trois) blocs majeurs, alignés avec la répartition sous-jacente. L'analyse visuelle des courbes, de la matrice ω et des partitions obtenues offre un retour précieux pour affiner ces paramètres en conséquence.

4.7.2. Cas Robotique ou Multi-Agent

En plus des scénarios de **simulation numérique** purement abstraits (4.7.1), un **SCN** (Synergistic Connection Network) peut s'appliquer à des contextes plus "réels" ou plus "pragmatiques", comme la coordination de **robots** ou d'**agents multiples**. Dans ce type de cadre, la **synergie** $S(i,j)$ n'est plus simplement une similarité vectorielle ou une compatibilité logique ; elle peut représenter la **complémentarité** de tâches ou la **cohérence** d'actions entre robots. On souhaite alors que le **DSL** (Deep Synergy Learning) favorise la formation de **sous-groupes** de robots qui collaborent efficacement, tout en laissant d'autres robots moins impliqués si leur contribution ne s'imbrique pas bien.

4.7.2.1. Rappel : Flottes de Robots, Ex. 5 Robots Simulés, Synergie = Réussite Conjointe de Tâches

Les principes du **Deep Synergy Learning (DSL)**, décrits dans les sections précédentes, peuvent s'appliquer de manière naturelle à des scénarios **multi-agents**, en particulier lorsqu'on considère plusieurs **robots** ou entités dotées de capacités complémentaires. Pour illustrer cette mise en pratique, on imagine une flotte réduite de cinq robots $\{r_1, \ldots, r_5\}$, chacun disposant de capteurs,

effecteurs ou aptitudes particulières, et l'on définit la synergie $S(r_i, r_j)$ sur la base de leur **réussite conjointe** dans des tâches passées ou de leur **compatibilité** fonctionnelle. L'**auto-organisation** d'un **SCN** édifié autour de ces robots peut alors faire émerger un partitionnement clair : un ensemble manipulatoire, un ensemble d'exploration, ou toute autre combinaison reflétant la meilleure configuration de collaboration selon les missions à accomplir.

A. Motivation Générale

Dans une configuration à cinq robots, il existe souvent un **partage** des rôles : certains se spécialisent dans la **manipulation** (bras robotisés), d'autres dans l'**inspection** (caméras, scanners, capacité de déplacement rapide). Les missions collectives imposent parfois une coordination de robots complémentaires, par exemple un robot manipulateur ayant besoin de l'assistance d'un robot d'inspection ou d'un robot de transport. La **synergie** $S(r_i, r_j)$ quantifie alors la probabilité (ou la qualité mesurée) que les robots r_i et r_j réalisent efficacement une tâche ensemble. Pour alimenter cette synergie :

On peut s'appuyer sur la **complémentarité** des modules (capteurs et actionneurs) : la présence d'un bras manipulateur et d'un détecteur de défauts peut s'additionner pour certaines missions.

On peut se baser sur un **historique** de performances : si deux robots ont maintes fois coopéré avec succès, $\mathrm{Hist}(r_i, r_j)$ s'en retrouve valorisée.

On peut inclure la **distance** entre robots si la proximité spatiale influe la communication ou la facilité de travail : $S(r_i, r_j)$ décroît alors quand $\mathrm{dist}(r_i, r_j)$ augmente.

Dans tous les cas, l'idée est de traduire la complémentarité et la réussite en un **score** S au sens du DSL, qui alimentera la mise à jour de $\omega_{i,j}(t)$.

B. Exemple Concret de Configuration

Supposons cinq robots $\{r_1, \dots, r_5\}$ dont trois $\{r_1, r_2, r_3\}$ sont dotés de bras manipulateurs. Les deux autres, $\{r_4, r_5\}$, privilégient l'inspection et la reconnaissance mobile, éventuellement plus rapides mais dépourvus de bras de préhension. Une **matrice** de synergie peut être construite : $S(r_i, r_j) \approx 0.7$ pour deux robots manipulateurs collaborant à une tâche d'assemblage, $S(r_i, r_j) \approx 0.4$ pour un manipulateur joint à un explorateur (un niveau moyen de coopération), et $S(r_i, r_j) \approx 0.1$ ou 0.2 pour deux robots explorateurs qui n'ont pas grand-chose à s'apporter dans une mission de montage.

En pratique, on peut enrichir la formulation :

$$S(r_i, r_j) = \alpha\, \mathrm{Comp}(r_i, r_j) + \beta\, \mathrm{Hist}(r_i, r_j) + \gamma\, \mathrm{Dist}(r_i, r_j),$$

avec Dist prenant la forme d'une fonction décroissante de la distance géographique (si la proximité influe). Par souci de simplicité, on peut rester dans un cas stationnaire où $\{S(r_i, r_j)\}$ ne varie pas au cours du temps, pour observer la formation d'un SCN.

C. Pourquoi un SCN ?

Dans une solution **centralisée**, il serait aisé de programmer explicitement : "Les robots r_1, r_2, r_3 travailleront ensemble pour les missions d'assemblage, r_4, r_5 couvriront l'inspection". Cependant, l'approche DSL propose une **auto-organisation** : chaque liaison $\omega_{i,j}(t+1)$ est mise à jour localement suivant la synergie $S(r_i, r_j)$ et un terme de décroissance ou d'inhibition. Cette mise à jour répétée conduit les robots partageant une forte synergie à renforcer leurs liens, alors que des paires moins avantageuses demeurent ou se rapprochent de zéro. Au final, un **cluster** regroupe naturellement les trois manipulateurs, tandis que les deux explorateurs ou inspecteurs conservent des pondérations réduites avec ceux-ci, ou forment un autre sous-bloc.

D. Exemple Numérique

Si l'on initialise $\omega_{i,j}(0) \approx 0$, puis qu'on applique, par exemple, la **règle additive** déjà évoquée :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(r_i, r_j) - \tau\, \omega_{i,j}(t)\big],$$

avec $\eta = 0.05$ et $\tau = 1.0$, alors les paires $\{(r_1, r_2), (r_1, r_3), (r_2, r_3)\}$ bénéficieront d'un **score** $S \approx 0.7$, ce qui leur permet de croître rapidement. Les paires (r_1, r_4), (r_2, r_5), etc., avec $S \approx 0.2$, demeureront plus modestes ou retomberont près de zéro. Après 100 à 200 itérations, on constate l'apparition d'un bloc $\{r_1, r_2, r_3\}$ relativement isolé du bloc $\{r_4, r_5\}$. Sur le plan **graphique**, la heatmap $\{\omega_{i,j}\}$ exhibe deux carrés plus saturés (ou un seul pour $\{r_4, r_5\}$, si ces deux explorateurs se connectent l'un à l'autre avec un certain niveau de synergie).

E. Analyse et Extensions

Un tel dispositif peut naturellement évoluer. Si l'on introduit un **changement** de mission, rendant la collaboration entre un manipulateur et un explorateur tout à fait pertinente, la synergie $S(r_i, r_j)$ peut progresser pour ce couple, et $\omega_{i,j}$ s'élever au point de faire fusionner des sous-blocs antérieurement disjoints. C'est là que réside la force d'un SCN : l'**adaptation** en continu, sans qu'un planificateur centralisé impose le regroupement. On peut aussi envisager d'intégrer un **terme** d'inhibition latérale γ pour forcer chaque robot à ne cultiver que peu de liens forts, ou un *clipping* pour éviter la saturation hors de propos.

Ce **exemple** à 5 robots démontre la facilité avec laquelle un **DSL** capture une **auto-organisation** multi-agent : la **synergie** définit la probabilité ou la qualité de réussite conjointe, et la mise à jour $\omega_{i,j}$ concrétise la consolidation ou l'abandon progressif d'une liaison. À l'échelle plus large, on peut imaginer des dizaines ou centaines de robots, répartis entre diverses missions. Le modèle DSL s'étend alors de la même façon, exploitant la dimension partiellement locale et réactive de la mise à jour pour assigner de facto les robots à des **clusters** de tâches.

Conclusion

La notion de **flotte de robots** constitue un **scénario** privilégié pour l'application d'un **SCN** : la **synergie** entre deux robots (r_i, r_j) reflète leur capacité à coopérer avec succès, soit par compatibilité d'effecteurs/capteurs, soit par l'historique de missions communes. En déroulant la dynamique de $\omega_{i,j}(t+1)$, on obtient naturellement la constitution d'un **cluster** de robots manipulateurs et un **cluster** de robots explorateurs, ou toute autre configuration correspondant aux

spécificités de leur collaboration. Cette démarche illustre la **puissance** du DSL, qui, sans supervision directe, génère un réseau de pondérations traduisant la **cohérence** (ou l'absence de cohérence) entre agents.

4.7.2.2. **Observations : Un Cluster se Forme entre 3 Robots Complémentaires, les 2 Autres se Marginalisent ou se Lient Plus Faiblement**

Dans le cadre d'une flotte de cinq robots, comme illustré en section 4.7.2.1, la dynamique des pondérations $\omega_{i,j}(t)$ appliquée par le Synergistic Connection Network (SCN) permet de mettre en évidence comment, sous l'hypothèse d'une synergie inégale entre certains robots, un **cluster principal** se forme spontanément. Ce cluster se compose typiquement de trois robots qui, en raison de leurs capacités complémentaires, développent des liens forts et cohérents, tandis que les deux autres robots présentent des synergies faibles avec ce groupe et avec eux-mêmes, conduisant ainsi à leur marginalisation ou à des connexions moins robustes. Nous allons maintenant développer en détail les différents aspects de cette observation en plusieurs parties.

A. Mise en Place et Déroulé de la Simulation

Le déploiement de la dynamique du SCN s'effectue en initialisant les pondérations $\omega_{i,j}(0)$ à des valeurs faibles, souvent proches de zéro, ce qui permet d'introduire un bruit initial qui simule des conditions réelles. La mise à jour des poids se fait selon la formule suivante :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(r_i, r_j) - \tau\,\omega_{i,j}(t)\right] - \gamma \sum_{k \neq j} \omega_{i,k}\,(t),$$

où :

- η représente le **taux d'apprentissage** qui détermine la vitesse d'évolution des pondérations,
- $S(r_i, r_j)$ est la **synergie** entre les robots r_i et r_j, fonction qui varie de 0 à 1 en fonction de la complémentarité de leurs capacités,
- τ est le **coefficient de décroissance** qui pénalise les liens trop forts pour éviter une croissance illimitée,
- γ est le paramètre d'**inhibition** qui sert à limiter la diffusion excessive des pondérations sur plusieurs liens, encourageant ainsi une répartition plus sélective des connexions.

Dans la simulation, les paramètres sont choisis de sorte que $\eta \approx 0.05$, $\tau \approx 1.0$ et $\gamma \approx 0.02$. L'algorithme s'exécute pour environ 200 à 300 itérations, ou jusqu'à ce que les modifications des pondérations deviennent négligeables, indiquant une stabilisation du réseau.

La synergie $S(r_i, r_j)$ est définie de manière à refléter la qualité de la collaboration entre les robots. Par exemple, pour le sous-groupe de trois robots complémentaires, on attribue des valeurs élevées, typiquement comprises entre 0.7 et 0.8, alors que pour les paires impliquant les deux robots restants, les scores peuvent être de l'ordre de 0.2 à 0.3, voire moins. Cette différence dans les

valeurs de synergie conditionne la dynamique : les liens entre les robots complémentaires se renforcent, tandis que les autres restent faibles ou nulle.

B. Le Cluster de Trois Robots Complémentaires

Au cours des itérations, les pondérations reliant les robots du sous-groupe complémentaire — disons r_1, r_2 et r_3 — augmentent progressivement. Par exemple, les liens $\omega_{1,2}(t)$ et $\omega_{1,3}(t)$ vont converger vers des valeurs proches de 0.6 à 0.7, ce qui correspond, en équilibre stationnaire, à la relation

$$\omega_{i,j}^{*} = \frac{S(r_i, r_j)}{\tau}.$$

Avec $\tau = 1$, si $S(1,2) = 0.8$ et $S(1,3) = 0.7$, on obtient respectivement $\omega_{1,2}^{*} \approx 0.8$ et $\omega_{1,3}^{*} \approx 0.7$. Ce phénomène de renforcement des liens parmi ces trois robots traduit la formation d'un **cluster principal**. L'effet d'inhibition, via le terme $-\gamma \sum_{k \neq j} \omega_{i,k}\,(t)$, contribue à concentrer l'énergie de chaque robot sur les connexions les plus profitables. En conséquence, ces robots renforcent préférentiellement leurs liens entre eux, se détachant ainsi du reste de la flotte.

C. Les Deux Autres Robots : Marginalisation ou Liens Faibles

Les robots r_4 et r_5 ne bénéficient pas des mêmes synergies. Deux scénarios principaux peuvent se présenter :

- Dans un premier cas, si la synergie entre r_4 et r_5 est très faible (par exemple, $S(r_4, r_5) \approx 0.1$ ou 0.2) et que leurs liens avec le groupe principal sont également faibles (autour de 0.3), alors les pondérations associées à ces robots restent faibles. Graphiquement, les valeurs de $\omega_{4,j}(t)$ et $\omega_{5,j}(t)$ stagnent près de zéro ou oscillent à de faibles niveaux, indiquant que ces entités ne participent pas activement à la formation du cluster principal.

- Dans un second cas, si r_4 et r_5 présentent une synergie intermédiaire entre eux (par exemple, $S(r_4, r_5) \approx 0.5$), ils peuvent former un petit sous-cluster distinct, avec des pondérations convergeant vers une valeur modérée (par exemple, 0.4 à 0.45). Toutefois, leurs liens avec le trio principal restent inférieurs, ce qui les marginalise vis-à-vis du cluster dominant.

Ces situations illustrent comment, selon la répartition des synergies, le SCN peut organiser hiérarchiquement les interactions entre robots, favorisant les collaborations les plus pertinentes et isolant ou réduisant l'influence des entités moins complémentaires.

D. Stabilisation de la Configuration et Visualisation

Après un nombre suffisant d'itérations (typiquement 200 à 300), la dynamique converge vers une configuration stable où la matrice de pondérations $\{\omega_{i,j}\}$ présente un bloc de fortes connexions entre r_1, r_2 et r_3, et des liens significativement plus faibles pour les interactions impliquant r_4 et r_5. La visualisation sous forme de **heatmap** permet de constater cette structuration : le bloc 3×3 correspondant au trio manipulateur affiche des valeurs proches de 0.6 à 0.8, tandis que les liens inter-clusters ou les connexions entre r_4 et r_5 apparaissent en contraste, souvent en dessous de 0.3. Des méthodes d'extraction de communautés, telles que l'indice de Silhouette ou l'algorithme de

Louvain, confirment que le cluster principal se distingue nettement, tandis que les deux autres robots restent soit isolés, soit regroupés dans un sous-ensemble à faible densité.

E. Conclusion

Les observations issues de cette simulation illustrent clairement que, dans un SCN appliqué à une flotte de robots, les entités ayant une forte **complémentarité** se regroupent naturellement pour former un **cluster principal**. Dans notre cas, les robots r_1, r_2 et r_3 développent des liens solides grâce à des synergies élevées (typiquement entre 0.7 et 0.8), ce qui se traduit par des pondérations $\omega_{i,j}$ élevées. En revanche, les robots r_4 et r_5 présentent des synergies faibles avec le groupe principal et entre eux, et leurs pondérations restent faibles ou convergent vers des valeurs modérées. Le paramètre d'inhibition γ joue un rôle essentiel en encourageant chaque robot à concentrer ses liens sur les partenaires les plus profitables, contribuant ainsi à la formation d'un cluster distinct. Ce résultat démontre la capacité du SCN à réaliser une **auto-organisation** efficace, révélant des clusters qui reflètent la logique de collaboration inhérente aux tâches robotiques, tout en marginalisant les entités dont la synergie ne justifie pas un renforcement important des liens. Cette observation, issue de simulations numériques, constitue un exemple concret de l'efficacité du DSL dans des environnements multi-robots, et offre une base pour ajuster les paramètres du système afin d'optimiser la cohérence et la robustesse des interactions collaboratives.

4.7.2.3. Visualisation Dynamique (Cycles, Stabilisation)

Les **expériences** menées dans le cadre robotique (voir 4.7.2.1 et 4.7.2.2) mettent en évidence la capacité d'un **Synergistic Connection Network** (SCN) à faire émerger, au sein d'une flotte de cinq robots, un **cluster** principal regroupant trois manipulateurs dotés d'une synergie élevée, tandis que les deux autres, moins pertinents à la même tâche, se marginalisent ou conservent des pondérations intermédiaires. Afin de **comprendre** plus finement la dynamique menant à cet état, il est précieux d'adopter des **techniques de visualisation** qui révèlent la progression pas à pas des pondérations $\omega_{i,j}(t)$, la survenue éventuelle de **cycles** ou oscillations, ainsi que le moment précis où se consolide la structure en sous-groupes.

A. Représentation sous forme de "Graph Dynamique"

Pour représenter l'évolution d'un **SCN**, on peut choisir un **graphe** pondéré dont les nœuds sont les robots $\{r_1, r_2, r_3, r_4, r_5\}$ et les arêtes sont les pondérations $\omega_{i,j}(t)$. Le principe est de produire une **animation** en temps discret : à chaque itération ou palier t, on met à jour :

> **L'épaisseur** (ou la couleur) de chaque arête en fonction de $\omega_{i,j}(t)$. Les liaisons fortes apparaissent plus visibles (arêtes plus larges, couleur saturée), les liaisons faibles restant minces ou quasi invisibles.
>
> **Le positionnement** des nœuds dans le plan (si l'on recourt à un layout type *force-directed*), ce qui permet de voir les nœuds fortement connectés se rapprocher progressivement. Certains utilisateurs conservent toutefois un positionnement fixe pour faciliter la comparaison dans le temps.

Cette représentation dynamique clarifie la formation du **cluster** des trois robots manipulateurs : au départ, toutes les arêtes sont fines ($\omega_{i,j} \approx 0$), puis celles reliant $\{r_1, r_2, r_3\}$ s'épaississent nettement, tandis que les connexions $\omega_{4,k}$ et $\omega_{5,k}$ demeurent maigres. Une fois l'itération 100 ou 150 atteinte, on constate un **noyau** visuel constitué de $\{r_1, r_2, r_3\}$, et le graphe se stabilise autour de ce bloc principal.

Cycles ou pseudo-oscillations.

Dans le cas où la dynamique de mise à jour (paramètres η, τ, γ) permet des **oscillations**, on percevrait un "va-et-vient" dans l'épaisseur de certaines arêtes, l'union de robots se renforçant brièvement puis chutant au profit d'un autre regroupement. Cet état se traduirait visuellement par une "valse" des nœuds dans le layout *force-directed*, ou par une alternance de couleurs pour des liens rivaux. Dans la configuration simple où la synergie des trois manipulateurs dépasse nettement celle des deux explorateurs, un tel phénomène survient peu, à moins que η ou γ ne soient mal calibrés.

B. Heatmap et Courbes de Pondérations

En parallèle d'un *graph* dynamique, on peut stocker et afficher la matrice $\{\omega_{i,j}(t)\}$ sous forme de **heatmap** (section 4.6.1.2). Du fait que la flotte compte seulement cinq robots, la matrice 5×5 est aisément lisible. Dès que les liens $\{\omega_{i,j}\}$ intra-cluster ($r_1 \leftrightarrow r_2$, $r_2 \leftrightarrow r_3$, etc.) grimpent, les cellules correspondantes du carré deviennent plus "chaudes" (couleurs saturées), alors que les autres restent plus froides (teintes pâles). On suit ainsi la montée de la diagonale 3×3 correspondant au noyau manipulateur, et la stagnation de $\omega_{4,j}$ et $\omega_{5,j}$.

Un autre niveau de détail s'obtient en **traçant** les courbes $\omega_{i,j}(t)$ pour quelques paires repères. Pour la paire (r_1, r_2) ou (r_2, r_3), une courbe monotone ou faiblement oscillatoire croît de 0 vers un plateau proche de 0.6–0.7, tandis que pour la paire (r_4, r_1), elle plafonne à 0.2 ou 0.3. Le moment où l'on atteint 90 % du plateau renseigne précisément sur l'itération où la **structure** se scelle. Si aucune oscillation n'apparaît, on parle de stabilisation ; si, au contraire, on voit un comportement fluctuant, c'est signe d'un **régime cyclique** ou multi-stable.

C. Intérêt de la Visualisation pour le Diagnostic

Cette **visualisation** dynamique permet de déterminer la **temporalité** de la formation du cluster manipulateur : on aperçoit à quelle itération les liaisons $\omega_{1,2}$ et $\omega_{2,3}$ commencent à surpasser un seuil implicite (ex. 0.5), scellant la communauté $\{r_1, r_2, r_3\}$. Les deux robots restants $\{r_4, r_5\}$ conservent des pondérations plus modestes, se trouvant donc marginalisés en fin de simulation.

En complément, la visualisation dévoile si la **dynamique** traverse une **phase transitoire** agitée (fortes oscillations puis stabilisation tardive) ou se fixe rapidement en un **cluster** ferme. À partir de là, il devient possible d'ajuster η, τ, γ afin d'obtenir un compromis satisfaisant entre **rapidité** de convergence, **contrôle** des oscillations et **clarté** de la séparation des robots.

D. Conclusion

Le recours à un **graph dynamique** (avec épaisseurs d'arêtes) ou à une **heatmap** itérative rend tangibles les phénomènes d'**auto-organisation** dans un **SCN** appliqué à une flotte robotique. Dans l'exemple, on suit l'évolution des pondérations $\{\omega_{i,j}(t)\}$ et on voit émerger un **cluster** de trois

robots manipulateurs solidement reliés, tandis que les deux autres s'écartent par manque de synergie. Selon la distribution initiale de $\omega_{i,j}$ et la différence entre les synergies, on constate un chemin plus ou moins direct vers la stabilisation. Cette approche visuelle est un **complément** essentiel aux seules mesures numériques : elle révèle la nature temporelle des liens, **comment** et **quand** s'établit la structure, et **détecte** aisément la présence éventuelle d'oscillations ou de compétitions prolongées.

4.7.3. Liens Pratiques

Après avoir illustré deux types de cas (4.7.1 : simulation semi-artificielle ; 4.7.2 : application robotique), il est naturel de se demander **comment** mettre en œuvre concrètement un **SCN** (Synergistic Connection Network) pour **reproduire** ces expériences ou en concevoir de nouvelles. La présente section (4.7.3) aborde quelques **pistes pratiques**, notamment :

> **Le code Python/C++ minimal** pour gérer la dynamique $\omega_{i,j}(t+1)$ (4.7.3.1),
>
> **Un outil de monitoring** ("SCN Dashboard") qui offre un **suivi** en temps réel de la formation des clusters (4.7.3.2).

4.7.3.1. Ce qu'on Pourrait Coder en Python/C++ pour Reproduire cette Dynamique en Quelques Centaines de Lignes

Il est souvent instructif de passer d'un **exemple théorique** à une **implémentation pratique**, même minimale, afin de rendre tangibles les principes de mise à jour $\omega_{i,j}(t+1)$ d'un **Synergistic Connection Network (SCN)$. L'**objectif** d'un tel prototype est de **montrer** comment, dans un code concis (Python ou C++), on peut :

1. Définir l'ensemble d'entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$.
2. Construire la **matrice** de synergie $\mathbf{S} = \big(S(i,j)\big)$.
3. Gérer la **matrice** $\boldsymbol{\omega}(t)$ représentant les pondérations.
4. Boucler sur un nombre d'itérations $T_{\max}$ afin d'**exécuter** la dynamique de mise à jour en s'appuyant sur les formules du **DSL**.

De cette façon, il devient possible, en quelques centaines de lignes, de **reproduire** les scénarios décrits précédemment : que ce soit le cas **mixte** (entités sub-symboliques, entités logiques) de la section 4.7.1, ou l'exemple **robotique** de la section 4.7.2, ou tout autre schéma dans lequel on veut explorer la formation de clusters par **auto-organisation**.

A. Structure Globale du Code

Le **cœur** de l'implémentation consiste à :

1. **Définir** le **nombre** d'entités, noté n.
2. **Allouer** et **remplir** la matrice $\mathbf{S} \in \mathbb{R}^{n \times n}$, qui fixe la **synergie** $S(i,j)$.
3. **Initialiser** la matrice $\boldsymbol{\omega}(0) \approx 0$ (ou un léger bruit), éventuellement en imposant $\omega_{i,i} = 0$ s'il n'existe pas de liaison réflexive.
4. Choisir les **paramètres** : η (taux d'apprentissage), τ (décroissance), γ (inhibition compétitive éventuelle) et le **nombre** d'itérations $T_{\max}$.
5. Entrer dans la boucle principale sur $t \in \{0, \dots, T_{\max} - 1\}$, où l'on met à jour $\omega_{i,j}(t+1)$ à partir de $\omega_{i,j}(t)$ et de $S(i,j)$.
6. (Optionnel) Stocker régulièrement des instantanés de la matrice $\boldsymbol{\omega}(t)$ pour les besoins de visualisation ou d'analyse.

Cette démarche se transpose aussi bien en **Python** qu'en **C++**. Dans Python, on utilise souvent *numpy* pour manipuler les matrices ; en C++, on recourra à des tableaux 2D ou à des conteneurs appropriés.

B. Exemple de Pseudo-Code (Version Additive)

Le pseudo-code ci-dessous, essentiellement en Python, illustre la **formulation additive** vue au chapitre 4.2.1 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right] - \gamma \sum_{k \neq j} \omega_{i,k}\,(t)$$

(avec un éventuel γ pour l'inhibition compétitive). Le code est exprimé de manière lisible, sans prétendre à une optimisation poussée :

```
import numpy as np

# 1) Définition des paramètres et données
n = 15              # nombre d'entités
eta = 0.05          # taux d'apprentissage
tau = 1.0           # décroissance
gamma = 0.0         # si on veut ajouter l'inhibition compétitive
T_max = 300         # nombre d'itérations

# Matrice de synergie S (n x n), initialisée à zéro
S = np.zeros((n, n))
# => Ici, on doit remplir S[i,j] selon le scénario :
#    - sub-symbolique / logique mixte,
#    - ou robots, etc.

# Matrice des pondérations w (n x n)
```

```
w = np.zeros((n, n))
# => on peut ajouter un petit bruit :
# w = np.random.uniform(0.0, 0.05, (n, n))
# for i in range(n):
#     w[i,i] = 0.0

# 2) Boucle de mise à jour
for t in range(T_max):
    w_next = np.copy(w)
    for i in range(n):
        for j in range(n):
            if i != j:
                # Terme principal
                delta = eta * (S[i,j] - tau*w[i,j])
                # Inhibition compétitive
                if gamma > 1e-12:
                    # somme de w[i,k] pour k != j
                    inhibition_term = gamma * (np.sum(w[i,:]) - w[i,j])
                    delta -= inhibition_term
                w_next[i,j] = w[i,j] + delta

                # Option : clipping pour éviter w < 0
                if w_next[i,j] < 0.0:
                    w_next[i,j] = 0.0

    w = w_next

    # (Option) Stocker ou imprimer un résumé de la matrice w(t)
    # if t % 10 == 0:
    #     print(f"Iteration {t}: some stats, e.g. mean={w.mean():.3f}, max={w.max():.3f}")

# 3) A la fin, w contient la matrice des pondérations finales
```

Ce **squelette** en Python, d'une centaine de lignes ou moins, fournit déjà le **cœur** d'un SCN dynamique. Pour un **scénario** plus complexe, on étoffe la partie **construction** de la matrice **S** (par exemple, pour des robots ou des entités logiques, on calcule les scores sub–sub, sym–sym, sub–sym). De même, on peut rendre la décroissance ou l'inhibition plus sophistiquées.

C. Extensions et Version C++

Pour reproduire cette simulation en C++, on définit des **tableaux** ou **vecteurs** 2D (par exemple *std::vector<std::vector<double>>*) pour **S** et $\boldsymbol{\omega}(t)$. Le schéma principal reste identique : une double boucle *for i in range(n): for j in range(n): ...* actualise $\omega_{i,j}$. On peut stocker la matrice à la fin de la simulation ou à intervalles réguliers.

Dans les deux langages, un **point crucial** est la manière dont la synergie **S** se construit, selon l'hétérogénéité des entités (partie sub-symbolique, symbolique, etc.). Le code de mise à jour reste lui-même identique, témoignant de la simplicité structurelle d'un SCN une fois que la **fonction** $S(i,j)$ est donnée.

D. Paramétrisation et Scénarios Multiples

Dans un **mini-programme** de ce type, on peut aisément :

Laisser η, τ, γ être passés en **argument** (ou chargés depuis un fichier de config) afin d'expérimenter la sensibilité de la convergence.

Générer **S** selon des distributions semi-aléatoires : par exemple, disjoindre des **sous-groupes** d'entités sub-symboliques avec un niveau de similarité 0.7–0.8 en interne et 0.2–0.3 à l'externe, rattacher à cela quelques entités logiques 0–1, etc.

Introduire un **mode évolutif** où **S** se modifie au fil du temps, ce qui exigerait un recalcul de $S(i,j)$ avant chaque itération.

Ainsi, quelques douzaines ou centaines de lignes de code suffisent à couvrir tous les **scénarios** des exemples (4.7.1) et (4.7.2), qu'il s'agisse de **robots** ou d'entités logiques/sub-symboliques mixtes.

E. Observation et Post-Traitement

Un code minimal comme ci-dessus se borne à mettre à jour ω. Dans la pratique, on procède à des **observations** pour :

Sauvegarder la matrice $\boldsymbol{\omega}(t)$ toutes les 10 ou 20 itérations.

Produire une **heatmap** ou un **graph** (utilisant, par exemple, NetworkX en Python) pour illustrer l'évolution visuelle (4.6.1).

Mesurer la **modularité** ou d'autres indices de partition (4.6.2) afin de vérifier la formation de clusters.

Ces étapes ne dépassent pas quelques lignes supplémentaires, tout en révélant la **dynamique** interne et la structure émergente.

Conclusion

En quelques centaines de lignes Python (ou C++), il est possible de **coder** la mise en œuvre fondamentale d'un **SCN** : on définit **S** (synergie), on initialise $\omega \approx 0$, puis on exécute la boucle de mise à jour $\omega_{i,j}(t+1)$. Malgré la brièveté du code, on parvient à reproduire tous les mécanismes décrits au chapitre 4.2 et dans les exemples (4.7.1, 4.7.2). Il suffit de paramétrer η, τ, γ et de spécifier **S** en fonction du contexte (sub-symbolique, logique, robotique, etc.). En sortie, la matrice

ω obtenue rend compte de l'**auto-organisation** en clusters, démontrant la force et la **simplicité** conceptuelle d'un DSL une fois traduite en code.

B. Conclusion (4.7.3.1)

Le développement d'un **mini-code** (en Python ou en C++) pour mettre en place la **dynamique** d'un **Synergistic Connection Network** (SCN) ne requiert qu'un nombre restreint d'instructions, généralement de l'ordre de quelques dizaines ou centaines de lignes. La logique est en effet assez directe : on se dote d'une **matrice S**, qui contient les synergies $S(i,j)$ entre toutes les entités, puis on déclare une **matrice ω**(0) (initialisée à zéro ou à un faible bruit) et une **boucle** d'itérations qui applique la règle de mise à jour décrite au chapitre 4.2 (additive, multiplicative, ou une variante avec inhibition). Il suffit enfin d'observer (ou d'enregistrer) l'évolution de $\omega_{i,j}(t)$ à chaque pas, afin de visualiser la montée ou la stagnation des liens et, in fine, la formation de **clusters**.

L'écriture du code en **Python** permet un prototypage rapide et une intégration aisée avec des librairies de visualisation (matplotlib, seaborn) ou de community detection (networkx). En **C++**, on gagne en performance si le nombre d'entités n devient important, par exemple au-delà de quelques milliers, tout en conservant le même principe de mise à jour. Dans les deux cas, on peut établir divers **scénarios** en variant la façon dont la synergie **S** est calculée : un mélange sub-symbolique/symbolique (tel que présenté en 4.7.1), une configuration multi-robots (4.7.2), ou toute autre instance s'appuyant sur des scores de complémentarité et d'historique.

Pourvu que le code structure correctement l'initialisation des $\omega_{i,j}$ et la boucle itérative, on peut alors, sans effort notable, y ajouter des **outils** pour la **collecte** ou l'**analyse** des résultats : export des matrices $\omega(t)$ à intervalles réguliers, tracés de courbes, heatmaps itératives, algorithmes de détection de communautés pour identifier les blocs émergents. Cette démarche rend possibles des **expérimentations** plus vastes ou plus complexes, tout en conservant la simplicité fondamentale du SCN. Ainsi, le cœur même de la **dynamique** ne nécessite qu'un petit nombre d'instructions, démontrant la **puissance** et la **généralité** d'un DSL, et la facilité avec laquelle on peut reproduire les scénarios présentés dans ce chapitre 4.7 en quelques centaines de lignes de code.

4.7.3.2. Aperçu du "SCN Dashboard" : Un Outil de Monitoring des $\omega_{i,j}$ pour Détecter la Formation des Clusters en Temps Réel

Dans un cadre de **Deep Synergy Learning (DSL)**, l'implémentation d'un **Synergistic Connection Network (SCN)** nécessite non seulement l'élaboration d'une dynamique de mise à jour des pondérations $\omega_{i,j}$, mais également un outil permettant de **suivre** en temps réel l'évolution de ces poids afin de détecter la formation des clusters et d'identifier les points critiques de réorganisation du réseau. Cet outil, que l'on désigne par le terme **"SCN Dashboard"**, offre une interface de monitoring et de visualisation qui traduit en temps réel la dynamique d'auto-organisation du système. Dans cette section, nous décrivons en détail les principes, l'architecture et les modalités opérationnelles de ce dashboard, en nous appuyant sur des formules mathématiques et des mécanismes algorithmiques concrets.

A. Principes Généraux du SCN Dashboard

Le **SCN Dashboard** a pour objectif de fournir une **vue synthétique** de l'évolution de la matrice de pondérations $\boldsymbol{\omega}(t) = \{\omega_{i,j}(t)\}$ au cours des itérations. Pour ce faire, le dashboard récupère régulièrement des instantanés de la matrice et les présente sous forme de visualisations interactives. Parmi ces visualisations, on retrouve notamment une **heatmap** de la matrice $\boldsymbol{\omega}$, des **courbes d'évolution** pour certaines liaisons représentatives, ainsi que des indicateurs quantitatifs tels que la **moyenne** et la **variance** des poids, ou encore des indices de **modularité** issus d'algorithmes de détection de communautés.

Matériellement, pour chaque itération t de la dynamique définie par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right]$$

(éventuellement complétée par des termes d'inhibition ou de clipping),

le dashboard capte les valeurs de $\omega_{i,j}(t)$ et calcule des métriques résumées, telles que

$$\overline{\omega}(t) = \frac{1}{n(n-1)}\sum_{i\neq j}\omega_{i,j}\,(t)$$

$$\sigma_{\omega}(t) = \sqrt{\frac{1}{n(n-1)}\sum_{i\neq j}(\,\omega_{i,j}(t) - \overline{\omega}(t))^2}.$$

Ces scalaires, ainsi que d'autres indicateurs (par exemple, le nombre de liens dépassant un seuil $\omega_{\min}$), sont affichés en temps réel pour permettre à l'utilisateur d'évaluer la **stabilité** du système.

B. Architecture Technique du Dashboard

Pour construire un **SCN Dashboard** opérationnel, plusieurs architectures techniques sont envisageables. La mise en œuvre peut se faire dans un environnement distribué ou en local, selon le volume des données et la nécessité d'une actualisation en temps réel. Deux approches principales sont :

Serveur intégré avec interface web :

Le système de simulation du SCN s'exécute sur un serveur (par exemple, en Python ou Node.js) et stocke périodiquement les instantanés de la matrice $\boldsymbol{\omega}(t)$ dans une base de données ou un fichier de log. Une interface web, développée en JavaScript (avec des bibliothèques telles que D3.js ou Plotly), interroge régulièrement ce serveur pour mettre à jour les visualisations. Cette approche permet une **interaction** en temps réel, où l'utilisateur peut ajuster dynamiquement les paramètres (par exemple, les valeurs de η, τ ou γ) via des **contrôles** graphiques (sliders, menus déroulants).

Processus séparé et visualisation post-hoc :

Dans un second schéma, la simulation du SCN est exécutée en tant que processus distinct, qui enregistre périodiquement les matrices $\boldsymbol{\omega}(t)$ dans un fichier. Un outil de visualisation,

également développé en Python (par exemple, en utilisant Matplotlib ou Seaborn), lit ces fichiers à intervalles réguliers et met à jour les graphiques. Cette solution est plus adaptée à des analyses postérieures, mais peut être complétée par des mécanismes d'actualisation en temps réel pour des applications nécessitant une surveillance continue.

Dans les deux cas, le dashboard intègre des **mécanismes de filtrage** pour réduire la charge visuelle. Par exemple, lorsque le nombre d'entités n est très élevé, le dashboard peut n'afficher que les liens dont les pondérations dépassent un certain seuil ou utiliser des techniques de **clustering** pour regrouper visuellement les entités.

C. Visualisations et Indicateurs

Le dashboard se doit de présenter plusieurs types de visualisations pour rendre la dynamique du SCN intelligible :

- **Heatmaps** : La matrice $\boldsymbol{\omega}(t)$ est représentée sous forme d'une heatmap, où chaque cellule (i, j) est colorée en fonction de la valeur de $\omega_{i,j}(t)$. Cette représentation permet de visualiser instantanément la formation de clusters (zones de couleur intense) et de détecter des zones de faibles connexions.

- **Graphes dynamiques** : Une représentation sous forme de graphe, avec des nœuds représentant les entités et des arêtes pondérées par $\omega_{i,j}(t)$, offre une perspective plus intuitive de la topologie émergente. Les algorithmes de layout de type « force-directed » repositionnent les nœuds en fonction de la force des liens, de sorte que les clusters apparaissent comme des groupes de nœuds rapprochés.

- **Courbes temporelles** : Pour certaines paires d'entités sélectionnées, le dashboard affiche l'évolution de $\omega_{i,j}(t)$ en fonction du temps, permettant d'identifier des régimes de convergence, d'oscillations ou d'instabilité.

- **Indicateurs scalaires** : Des statistiques globales telles que la moyenne, la variance et des indices de modularité (issus d'algorithmes de community detection) sont calculées et affichées. Ces indicateurs permettent de quantifier la qualité de la partition en clusters et de surveiller la stabilité du système.

L'utilisateur peut également disposer d'**outils interactifs** pour modifier les paramètres du système en direct. Par exemple, des sliders peuvent permettre de varier η, τ ou γ et d'observer immédiatement l'impact de ces modifications sur la convergence ou la formation de clusters.

D. Mise en Pratique et Exemples d'Utilisation

Pour illustrer le fonctionnement du dashboard, considérons l'exemple d'une flotte de robots dont les pondérations évoluent selon la règle du SCN. Le dashboard permet alors de suivre en temps réel la matrice $\boldsymbol{\omega}(t)$ et de détecter, par exemple, la formation d'un cluster principal de robots complémentaires. Au fur et à mesure que la simulation progresse, la heatmap révèle l'émergence d'un bloc de valeurs élevées correspondant aux liens internes du cluster. Parallèlement, un graphe dynamique montre que les nœuds représentant ces robots se rapprochent physiquement, tandis que les robots moins complémentaires se déplacent en périphérie. De plus, des courbes temporelles affichées pour certains liens illustrent la convergence exponentielle vers des points fixes, tandis

que d'autres liens montrent des oscillations indiquant des transitions entre différents régimes d'interaction.

Un tel dashboard ne se contente pas de visualiser la dynamique, il sert également de **outil de diagnostic** et de **pilotage**. Par exemple, si les indices de modularité diminuent brusquement ou si la variance des pondérations augmente, cela peut alerter l'opérateur sur une possible instabilité ou sur la nécessité de réajuster les paramètres du SCN (par le biais d'une inhibition plus forte ou d'un ajustement du taux d'apprentissage). Ce retour d'information en temps réel est particulièrement précieux dans des applications critiques, telles que la coordination multi-robots en environnement dynamique, où l'adhérence à une configuration stable et cohérente est essentielle pour la réussite de la mission.

E. Conclusion

Le **SCN Dashboard** se présente comme un outil de monitoring interactif permettant de visualiser et de contrôler la dynamique des pondérations $\omega_{i,j}$ dans un Deep Synergy Learning. En exploitant des visualisations telles que des heatmaps, des graphes dynamiques et des courbes temporelles, le dashboard offre un aperçu détaillé de la formation des clusters en temps réel. Il permet non seulement de détecter rapidement la consolidation des liens entre entités fortement synergiques, mais aussi d'identifier des signaux précurseurs d'instabilités ou d'oscillations, facilitant ainsi l'ajustement des paramètres du système. En résumé, ce tableau de bord constitue un pont essentiel entre la théorie du DSL et son application pratique, offrant à la fois un outil de recherche pédagogique et une interface de supervision opérationnelle dans des environnements multi-agents ou robotiques.

4.8. Stabilisation Avancée : Couplages avec Chapitre 5 (Architecture SCN)

Les sections antérieures du chapitre 4 ont mis en lumière la **dynamique** d'un SCN (Synergistic Connection Network) : les mises à jour $\omega_{i,j}$, la formation de clusters, la gestion de paramètres comme η, τ, γ. Toutefois, pour qu'un **DSL** (Deep Synergy Learning) fonctionne de manière **pérenne** et **scalable**, il est aussi crucial de **concevoir** l'**architecture logicielle** appropriée, détaillée au **Chapitre 5**. Cette architecture doit non seulement soutenir la **dynamique auto-organisée** mais également permettre :

- Un **déploiement** clair des différents modules (calcul de $S(i,j)$, mise à jour de $\omega_{i,j}$, inhibition globale, etc.),
- Une **gestion** ordonnée des ressources (threads, verrous, données),
- Une **intégration** flexible (possibilité de changer de schéma de mise à jour ou d'inhibition).

4.8.1. Rôle de l'Architecture

4.8.1.1. Comment l'Organisation Logicielle (Chap. 5) Facilite le Déploiement de la Dynamique Auto-Organisée

La mise en œuvre efficace d'un **Synergistic Connection Network (SCN)** repose sur une **infrastructure logicielle** soigneusement conçue qui permet de séparer les responsabilités au sein du système et de gérer la dynamique d'auto-organisation de manière modulable et évolutive. L'organisation logicielle, telle que décrite au Chapitre 5, joue un rôle déterminant dans le déploiement de la dynamique de mise à jour des pondérations $\omega_{i,j}(t)$ et dans l'intégration des divers modules de calcul de la **synergie** $S(i,j)$, de l'**inhibition** et du **contrôle des paramètres**.

*A. **Modularisation et Séparation des Rôles***

L'un des principes fondamentaux de l'**organisation logicielle** appliquée à un SCN est la **modularisation**. Plutôt que d'imbriquer l'ensemble des calculs dans un unique bloc de code, on divise le système en modules indépendants, chacun ayant un rôle spécifique. Par exemple, le **Module de Calcul de Synergie** est chargé d'extraire et de combiner les informations issues de diverses sources (embeddings sub-symboliques, règles symboliques, etc.) afin de produire un score $S(i,j)$. Ce module peut utiliser diverses méthodes, telles que la **similarité cosinus** ou des noyaux RBF, pour obtenir une mesure quantitative de la proximité entre deux entités. De même, le **Module de Mise à Jour des Pondérations** applique la règle d'auto-organisation, généralement de la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big],$$

où η est le **taux d'apprentissage** et τ le **coefficient de décroissance**. Enfin, un **Module d'Inhibition/Régulation** peut être ajouté pour introduire des mécanismes de sparsification ou de limitation (par exemple, via un terme $-\gamma \sum_{k\neq j} \omega_{i,k}\,(t)$) qui garantissent que la structure du réseau reste compacte et que les liens moins pertinents ne perturbent pas la dynamique globale.

Cette séparation permet d'isoler les modifications : un changement dans la méthode de calcul de $S(i,j)$ (par exemple, passer d'un modèle sub-symbolique à une approche hybride) n'impacte pas directement la logique de mise à jour des pondérations, facilitant ainsi la maintenance et l'évolution du code.

B. Flexibilité dans le Choix des Schémas

L'**organisation logicielle** favorise également la flexibilité dans le choix des schémas de mise à jour. Par exemple, la règle de mise à jour additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right]$$

peut être aisément remplacée par une variante multiplicative si le contexte l'exige, comme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t)\exp\left(\eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right]\right).$$

L'architecture logicielle modulaire permet de modifier uniquement le **Module de Mise à Jour des Pondérations** sans avoir à réécrire la totalité du système. Par ailleurs, les paramètres globaux η, τ et γ peuvent être centralisés dans un **Module de Gestion des Paramètres**, qui fournit des interfaces permettant d'ajuster ces valeurs en fonction des besoins de l'application ou des observations en temps réel.

C. Gestion Cohérente des Itérations et du Flux de Données

Un autre aspect clé de l'organisation logicielle est la gestion structurée de la boucle itérative qui orchestre le SCN. Une **boucle centrale** ou **engine** est mise en place pour :

- Appeler le **Module de Calcul de Synergie** pour recalculer $S(i,j)$ (si nécessaire, notamment dans des systèmes non stationnaires),
- Appliquer la règle de mise à jour aux pondérations $\omega_{i,j}$ à chaque itération,
- Activer des mécanismes de **monitoring** et de **visualisation** (par exemple, en enregistrant périodiquement la matrice $\boldsymbol{\omega}(t)$ pour en générer des heatmaps ou des graphes dynamiques).

Ce processus est rendu encore plus robuste par l'introduction d'**observateurs** (observers) qui analysent la convergence, par exemple en mesurant la norme $\| \boldsymbol{\omega}(t+1) - \boldsymbol{\omega}(t) \|$ et en déclenchant des ajustements si la convergence se révèle trop lente ou si des oscillations apparaissent. De plus, la modularité de la boucle permet d'intégrer des mécanismes de contrôle comme le recuit stochastique pour échapper aux minima locaux peu satisfaisants.

D. Stabilisation et Contrôle de la Dynamique

L'**organisation logicielle** favorise une gestion fine de la dynamique d'auto-organisation grâce à des modules spécifiques qui surveillent et ajustent le comportement du SCN. Par exemple, un **Module de Stabilisation** peut intervenir en temps réel pour ajuster les paramètres η et τ en fonction des indicateurs de convergence et des oscillations détectées. Ce module peut également appliquer des techniques de **clipping** et de **sparsification** pour empêcher une prolifération excessive de liens, garantissant ainsi la robustesse et la lisibilité de la matrice $\boldsymbol{\omega}$.

L'approche permet de tester différents scénarios : dans un système multi-entités, la même infrastructure logicielle peut être utilisée pour différents types d'applications (robotique collaborative, agents conversationnels, systèmes de recommandation, etc.), en modifiant uniquement le module de calcul de synergie $S(i,j)$ et en adaptant les paramètres de la boucle d'itération. Cette capacité à réutiliser et à adapter le **moteur d'auto-organisation** est l'un des avantages majeurs de l'organisation logicielle présentée au Chapitre 5.

E. Intégration Multi-Scénario et Maintenance

Enfin, l'architecture logicielle permet une **intégration multi-scénario** efficace. Qu'il s'agisse de systèmes sub-symboliques, symboliques ou hybrides, le même cadre d'auto-organisation peut être déployé avec des modules spécifiques pour le calcul de la synergie et pour la régulation des pondérations. Cette uniformisation facilite non seulement le développement initial mais aussi la maintenance à long terme, puisque chaque composant évolue de manière indépendante. Par ailleurs, les interfaces de monitoring et de contrôle offertes par le SCN Dashboard (voir section 4.7.3.2) permettent aux opérateurs de visualiser en temps réel l'évolution du réseau et d'ajuster les paramètres via des interfaces ergonomiques, rendant ainsi l'ensemble du système adaptable à des environnements en constante évolution.

Conclusion

En résumé, l'**organisation logicielle** présentée au Chapitre 5 facilite le déploiement de la **dynamique auto-organisée** dans un SCN en offrant une **modularisation** claire qui sépare le calcul de la synergie, la mise à jour des pondérations et les mécanismes de régulation. Cette approche permet non seulement d'optimiser la maintenance et l'évolution du code, mais aussi d'adapter le système à divers scénarios applicatifs grâce à des modules interchangeables et à des interfaces de monitoring en temps réel. Les paramètres globaux tels que η, τ et γ sont gérés de manière centralisée, facilitant ainsi leur ajustement en fonction des besoins du système et garantissant une **stabilisation** dynamique efficace. Cette infrastructure logicielle constitue le socle sur lequel repose l'implémentation robuste, évolutive et maintenable d'un DSL, capable de s'adapter aux exigences complexes des environnements multi-agents et robotiques modernes.

4.8.1.2. Notion de Modules Distincts : "Calcul de Synergie", "Update ω", "Inhibition Global ou Local", etc.

Dans la mise en œuvre d'un **Synergistic Connection Network (SCN)**, il apparaît primordial de structurer le code en **modules distincts** afin de garantir la **clarté**, la **flexibilité** et la **maintenabilité** de l'architecture. Cette approche modulaire, qui sera développée plus en détail au Chapitre 5, permet de séparer les responsabilités liées au **calcul de la synergie**, à la **mise à jour des pondérations** $\omega_{i,j}$, et aux mécanismes de **régulation** tels que l'inhibition (globale ou locale). En procédant ainsi, le système se construit comme un ensemble de briques fonctionnelles interopérables, chacune étant aisément remplaçable ou ajustable selon le contexte d'application. Nous détaillons ci-après les principaux modules qui constituent ce cadre opérationnel.

*A. **Module "Calcul de Synergie"***

Le **module de Calcul de Synergie** est chargé de déterminer le score $S(i,j)$ qui mesure la qualité de la relation entre deux entités $\mathcal{E}_i$ et $\mathcal{E}_j$. Ce score peut être obtenu à partir de représentations sub-symboliques (telles que des **embeddings** issus de CNN, Transformers, etc.), de critères symboliques (basés sur des règles logiques, axiomes ou ontologies), ou d'une combinaison hybride de ces deux approches. Mathématiquement, on peut écrire la fonction de synergie de manière générique comme

$$S(i,j) = f\big(\mathbf{r}(i), \mathbf{r}(j)\big),$$

où $\mathbf{r}(i)$ représente la **représentation** de l'entité i (qui peut être un vecteur, un ensemble de règles, ou une concaténation de ces deux types d'informations) et f est une fonction mesurant la similarité ou la compatibilité. Par exemple, dans un contexte purement sub-symbolique, on utilisera typiquement la **similarité cosinus** :

$$S_{\text{sub}}(i,j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\| \mathbf{x}_i \| \| \mathbf{x}_j \|},$$

tandis que dans un contexte symbolique, $S_{\text{sym}}(i,j)$ pourra être définie en fonction de la concordance entre deux ensembles de règles R_i et R_j. Dans un système hybride, on opère une fusion pondérée telle que

$$S_{\text{hybrid}}(i,j) = \alpha\, S_{\text{sub}}(i,j) + (1-\alpha)\, S_{\text{sym}}(i,j),$$

où $\alpha \in [0,1]$ détermine l'importance relative des deux composantes. La modularité de ce calcul permet de modifier ou de substituer le module de synergie sans perturber la structure de la mise à jour des pondérations, rendant ainsi le système adaptable à divers types de données et de contextes.

*B. **Module "Update ω"***

Le **module de mise à jour des pondérations** est le cœur de la dynamique d'auto-organisation du SCN. Il applique les règles d'évolution des liens qui relient les entités entre elles. La règle de mise à jour additive classique s'exprime par :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\, \omega_{i,j}(t)\big],$$

où η est le **taux d'apprentissage** et τ représente un terme de **décroissance** qui pénalise une croissance excessive des liens. Dans certains cas, pour obtenir une dynamique plus exponentielle, on peut opter pour une variante multiplicative :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t)\exp\big(\eta\big[S(i,j) - \tau\, \omega_{i,j}(t)\big]\big).$$

Ce module est conçu pour être indépendant du module de calcul de la synergie ; il reçoit en entrée la matrice $\mathbf{S} = \{S(i,j)\}$ et les paramètres η et τ, et renvoie la nouvelle matrice de pondérations. Par cette séparation, il devient trivial de changer le schéma de mise à jour (passer d'un modèle additif à un modèle multiplicatif, ou encore y intégrer des mécanismes de clipping ou d'inhibition) sans affecter le calcul du score de synergie.

C. Module "Inhibition Global ou Local"

Dans des systèmes complexes, il est souvent nécessaire d'introduire une **inhibition** afin d'empêcher qu'une entité ne se lie de manière excessive à un trop grand nombre de partenaires. Cette inhibition peut être implémentée de deux manières principales :

- **Inhibition locale** : Chaque entité $\mathcal{E}_i$ se voit imposer une contrainte sur la somme des pondérations associées à ses liens. Par exemple, on peut imposer que

$$\sum_{j \neq i} \omega_{i,j} \leq \Omega_{\max},$$

où $\Omega_{\max}$ est un seuil maximal. Cela force l'entité à concentrer ses liens sur les partenaires les plus synergiques.

- **Inhibition globale** : On ajoute directement un terme d'inhibition dans la règle de mise à jour, tel que

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big] - \gamma \sum_{k \neq j} \omega_{i,k}\,(t),$$

où γ est un coefficient d'inhibition qui réduit la pondération si l'entité est déjà fortement connectée à plusieurs partenaires.

Ce module peut être conçu comme une fonction à part entière, qui s'applique en post-traitement de la mise à jour des $\omega_{i,j}$. En isolant ce mécanisme, on peut facilement expérimenter avec différentes stratégies d'inhibition (par exemple, en modifiant le seuil $\Omega_{\max}$ ou en ajustant le coefficient γ) sans impacter la logique de calcul de la synergie ou la mise à jour des pondérations.

D. Synergie d'Ensemble et Ordonnancement

L'architecture logicielle d'un SCN s'organise de manière séquentielle en un pipeline d'itération. À chaque itération t, le **moteur** de l'auto-organisation procède de la manière suivante :

Le **Module de Calcul de Synergie** est appelé pour déterminer les scores $S(i,j)$ pour toutes les paires d'entités.

Le **Module de Mise à Jour** applique la règle de mise à jour sur la matrice $\boldsymbol{\omega}(t)$ pour générer $\boldsymbol{\omega}(t+1)$.

Le **Module d'Inhibition** intervient ensuite pour appliquer des ajustements supplémentaires, tels que la régulation de la somme des liens ou le clipping.

Des **observateurs** ou **logs** sont activés pour suivre l'évolution de la matrice $\boldsymbol{\omega}(t)$ et calculer des indicateurs de convergence, par exemple la norme de la différence $\| \boldsymbol{\omega}(t+1) - \boldsymbol{\omega}(t) \|$ ou des indices de modularité issus d'algorithmes de détection de communautés.

Ce séquençage permet d'intégrer aisément des mécanismes de **contrôle adaptatif**. Par exemple, un **Module de Paramètres** central peut ajuster dynamiquement η et τ en fonction des indicateurs de stabilité, ou activer un **recuit stochastique** pour sortir des minima locaux peu intéressants.

*E. **Avantages de la Modularisation***

La division du système en modules distincts présente plusieurs **avantages** majeurs :

- **Clarté et maintenabilité** : Chaque module est développé et testé indépendamment, ce qui facilite la compréhension globale du système et simplifie les modifications ou mises à jour.
- **Flexibilité** : Il devient trivial de substituer ou d'améliorer un module particulier sans affecter les autres parties du SCN. Par exemple, l'algorithme de calcul de synergie peut être remplacé par une version hybride (fusionnant sub-symbolique et symbolique) tout en conservant la même interface avec le module de mise à jour des ω.
- **Extensibilité** : Dans des scénarios multi-entités ou multi-domaines, le même cadre de mise à jour peut être réutilisé en adaptant uniquement le module de calcul de synergie ou les paramètres de régulation, ce qui permet une intégration homogène de divers types de données et de logiques.
- **Contrôle et adaptation** : La structure modulaire permet l'implémentation de mécanismes de surveillance et de contrôle (inhibition, clipping, recuit) qui peuvent être activés ou désactivés indépendamment, offrant ainsi une gestion fine de la dynamique d'auto-organisation.

Conclusion

L'approche modulaire, telle que présentée dans cette section, est essentielle pour traduire les principes théoriques du SCN en une solution logicielle opérationnelle. En séparant distinctement le **calcul de la synergie**, la **mise à jour des pondérations** et les mécanismes d'**inhibition** ou de **régulation**, le système gagne en **clarté**, en **flexibilité** et en **extensibilité**. Ce découpage permet non seulement de maintenir une architecture propre et évolutive, mais aussi d'expérimenter différents schémas de mise à jour et de contrôler la dynamique du réseau par des ajustements paramétriques précis. En intégrant ces modules dans une boucle d'itération orchestrée par un moteur central, le SCN peut être déployé efficacement pour diverses applications, allant de la robotique collaborative à la gestion de systèmes multi-agents, en garantissant une auto-organisation robuste et adaptable aux exigences spécifiques de chaque domaine.

4.8.2. Gestions des Ressources et Threads

Dans un **SCN** (Synergistic Connection Network) à grande échelle, la boucle de mise à jour $\omega_{i,j}(t+1)$ peut rapidement devenir un **goulet d'étranglement** si elle s'exécute de façon **strictement séquentielle**. On peut souhaiter **paralléliser** certaines opérations ou recourir à des **threads** multiples, surtout si n (le nombre d'entités) se compte en milliers ou plus. Toutefois, cette parallélisation doit être orchestrée avec soin pour éviter les **incohérences** dans le calcul de $\Delta\omega_{i,j}$.

4.8.2.1. **Mise à Jour Parallèle ou Asynchrone : Chaque Entité Calcule Localement $\Delta\omega_{i,j}$**

Dans les systèmes de **Deep Synergy Learning (DSL)**, la mise à jour des pondérations $\omega_{i,j}(t)$ entre entités se réalise traditionnellement par le biais d'une double boucle itérative sur toutes les paires (i,j). Cependant, lorsque le nombre d'entités n devient grand, le coût de calcul, de l'ordre de $\mathcal{O}(n^2)$ opérations par itération, peut rapidement devenir prohibitif. Pour pallier cette limitation, il est naturel d'envisager une **mise à jour parallèle ou asynchrone** des pondérations, dans laquelle chaque entité calcule localement sa contribution $\Delta\omega_{i,j}$ aux mises à jour, permettant ainsi de répartir la charge de calcul entre plusieurs threads ou processeurs. Nous présentons ici de manière détaillée les mécanismes, les avantages et les risques associés à cette approche.

*A. **Principe de la Mise à Jour Parallèle***

L'idée fondamentale consiste à **diviser** le calcul de la matrice des pondérations $\boldsymbol{\omega}(t)$ en sous-tâches indépendantes, qui peuvent être exécutées simultanément sur différents cœurs ou processeurs. Soit n le nombre d'entités, la mise à jour standard s'exprime par :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big],$$

où $S(i,j)$ représente la synergie entre les entités i et j, η est le taux d'apprentissage et τ le coefficient de décroissance. Pour paralléliser ce calcul, plusieurs approches sont possibles. Par exemple, l'on peut découper la matrice $\boldsymbol{\omega}$ en **sous-blocs** ou en attribuant à chaque entité i la responsabilité de mettre à jour tous les liens sortants $\omega_{i,j}$ pour $j = 1, \ldots, n$. Formellement, chaque entité réalise le calcul :

$$\Delta\omega_{i,j} = \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big],$$

puis met à jour localement

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta\omega_{i,j}.$$

L'utilisation d'un **double-buffer** est souvent recommandée pour éviter les conflits d'accès en écriture. On définit ainsi une matrice $\mathbf{w}^{(t+1)}$ dans laquelle chaque thread écrit ses mises à jour, alors que la matrice $\mathbf{w}^{(t)}$ est lue en lecture seule pendant toute l'itération.

*B. **Synchronous vs. Asynchronous Update***

Dans le **mode synchrone**, toutes les entités ou tous les threads réalisent leur calcul de $\Delta\omega_{i,j}$ en se basant sur la même version de la matrice $\omega(t)$. La procédure se décompose en deux phases distinctes :

- **Phase de lecture** : Chaque thread lit la matrice $\omega(t)$ pour calculer les variations locales $\Delta\omega_{i,j}$ pour les indices qui lui sont assignés.
- **Phase de mise à jour** : Une fois que toutes les variations ont été calculées, un **barrier** ou **synchronisation** est appliquée, puis la nouvelle matrice $\omega(t+1)$ est construite en copiant les résultats du buffer de sortie.

Mathématiquement, on définit deux matrices $\mathbf{w}^{(t)}$ et $\mathbf{w}^{(t+1)}$ telles que, pour tous i et j,

$$\mathbf{w}_{i,j}^{(t+1)} = \mathbf{w}_{i,j}^{(t)} + \eta\left[S(i,j) - \tau\,\mathbf{w}_{i,j}^{(t)}\right].$$

Ce mode garantit une **cohérence** absolue puisque chaque entité travaille sur une version identique de $\omega(t)$. Le principal inconvénient réside dans la nécessité d'attendre que tous les threads aient terminé leur calcul, ce qui peut introduire un délai de synchronisation dans des environnements massivement parallèles.

En **mode asynchrone**, chaque entité met à jour ses propres liens dès que son calcul de $\Delta\omega_{i,j}$ est terminé, sans attendre la fin de l'itération pour l'ensemble du réseau. Cela peut se formaliser par l'équation :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right],$$

où la lecture et l'écriture de $\omega_{i,j}$ se font de manière **concurrente** et **lock-free**. Dans ce contexte, il arrive que différentes entités lisent des versions légèrement différentes de la matrice ω (certaines mises à jour étant déjà effectuées tandis que d'autres non). Ce mode peut accélérer le temps de calcul global, surtout dans des architectures distribuées, mais il soulève des problèmes de **cohérence** et de **stabilité**. Par exemple, une mise à jour asynchrone peut être analysée comme une itération de type Gauss-Seidel, où l'**ordre** d'exécution influe sur la convergence. Des techniques telles que le **verrouillage léger** ou l'emploi de structures atomiques peuvent être nécessaires pour minimiser les conflits, sans pour autant imposer une synchronisation complète.

C. Avantages et Risques du Calcul Parallèle/Asynchrone

L'**avantage principal** de la mise à jour parallèle ou asynchrone réside dans la **réduction du temps de calcul** global. En divisant la charge de mise à jour sur plusieurs threads, le coût de $\mathcal{O}(n^2)$ opérations peut être réduit de manière quasi linéaire avec le nombre de processeurs disponibles, permettant ainsi d'aborder des systèmes à grande échelle. Par ailleurs, un mode asynchrone, s'il est correctement contrôlé, permet une **adaptation continue** et une mise à jour plus fluide de ω, même dans des environnements distribués.

Cependant, ces gains de performance ne vont pas sans risques. En mode asynchrone, l'absence d'une synchronisation stricte peut conduire à des **incohérences** où certaines mises à jour sont appliquées sur des données déjà modifiées, générant des écarts imprévus et potentiellement des oscillations dans les valeurs des pondérations. La garantie théorique de convergence devient alors plus complexe, nécessitant l'emploi d'algorithmes de contrôle tels que les techniques basées sur le **Gossip** ou des schémas de verrouillage léger pour assurer que la dynamique globale demeure stable.

D. Stratégies pour la Mise en Œuvre Asynchrone

Afin d'assurer la **stabilité** tout en bénéficiant des avantages du parallélisme, plusieurs stratégies peuvent être mises en œuvre :

Double Buffering :

Chaque itération lit une copie immuable de la matrice $\mathbf{w}^{(t)}$ pour effectuer les calculs et écrit les résultats dans une nouvelle matrice $\mathbf{w}^{(t+1)}$. Cette approche garantit que toutes les mises

à jour de la phase t sont basées sur des données cohérentes, éliminant ainsi les conflits d'accès simultanés.

Utilisation de Verrous Atomiques ou Lock-Free :

Dans des environnements multi-threadés, l'emploi de verrous atomiques permet de s'assurer que les mises à jour sur un élément $\omega_{i,j}$ ne sont pas interrompues ou écrasées par d'autres threads. Bien que cette approche puisse introduire un certain coût en termes de performance, elle offre une **garantie** de cohérence. Des algorithmes lock-free sophistiqués, inspirés des techniques de la programmation concurrente, peuvent également être envisagés pour minimiser les surcharges tout en maintenant la stabilité.

Décomposition en Blocs :

En divisant la matrice **w** en sous-blocs disjoints, chaque thread peut être affecté à un bloc spécifique, réduisant ainsi les risques de conflits. Cette approche est particulièrement efficace si l'on peut isoler des sous-ensembles d'entités ayant peu d'interaction avec d'autres sous-ensembles.

Ajustement Dynamique des Paramètres :

Les paramètres η et τ jouent un rôle crucial dans la vitesse de convergence et la stabilité. Un **Module de Gestion des Paramètres** peut surveiller en temps réel la variation des pondérations et ajuster η de manière à atténuer les oscillations lorsque des mises à jour asynchrones induisent des incohérences.

*E. **Conclusion***

La mise à jour parallèle ou asynchrone dans un SCN représente une approche essentielle pour rendre le déploiement d'un **DSL** scalable et efficace, en particulier lorsque le nombre d'entités n est élevé. En permettant à chaque entité ou à chaque bloc d'entités de calculer localement $\Delta\omega_{i,j}$ et de mettre à jour les pondérations de manière simultanée, on réduit significativement le temps de calcul global. Toutefois, cette approche soulève des défis importants en termes de **cohérence** et de **stabilité** de la dynamique, nécessitant des stratégies telles que le double buffering, l'utilisation de verrous atomiques, la décomposition en blocs et l'ajustement dynamique des paramètres. L'objectif ultime est de garantir que, malgré la nature asynchrone des mises à jour, le système converge vers une configuration stable et fiable, tout en profitant des avantages du parallélisme pour traiter de très grands réseaux d'entités en temps réel. Cette méthode opérationnelle est le pilier qui permet au DSL de s'adapter aux environnements complexes et distribués, assurant ainsi une auto-organisation robuste et scalable.

4.8.2.2. **Conflits Possibles, Besoin de Cohérence Globale (Verrous, ou Algorithmes Lock-Free ?)**

Dans un environnement de mise à jour parallèle ou asynchrone d'un **Synergistic Connection Network (SCN)**, la **cohérence** globale des pondérations $\omega_{i,j}$ est mise à rude épreuve par la concurrence des opérations de lecture et d'écriture. Dès lors que plusieurs threads ou processus

interviennent simultanément pour calculer les incréments $\Delta\omega_{i,j}$ et mettre à jour les valeurs correspondantes, il apparaît impératif d'assurer une synchronisation adéquate afin de préserver l'intégrité de la dynamique globale. Dans cette section, nous analysons les conflits qui peuvent surgir, les approches traditionnelles basées sur des verrous (mutex) et les alternatives lock-free, et nous discutons de leur impact sur la stabilité et la convergence du SCN.

A. Nature des Conflits en Mise à Jour Concurente

Le SCN repose sur la règle de mise à jour additive donnée par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big],$$

où $S(i,j)$ représente le score de synergie entre les entités i et j. Dans un environnement parallèle, la matrice $\boldsymbol{\omega}$ est partagée entre plusieurs threads, et plusieurs d'entre eux peuvent tenter de lire ou d'écrire simultanément la même valeur $\omega_{i,j}(t)$. Par exemple, si un thread lit $\omega_{i,j}(t)$ pour calculer $\Delta\omega_{i,j}$ pendant qu'un autre thread a déjà effectué une mise à jour sur la même case, alors la valeur utilisée pour le calcul peut être obsolète ou partiellement mise à jour. Ce phénomène de **races conditions** (conditions de concurrence) peut conduire à des incohérences, notamment lorsque la somme d'une inhibition globale dépend de la lecture simultanée de plusieurs $\omega_{i,k}(t)$. En d'autres termes, des mises à jour concurrentes non coordonnées peuvent aboutir à une matrice qui ne reflète plus fidèlement la dynamique théorique du système, générant ainsi des déphasages, des oscillations imprévues ou même la divergence de la procédure de convergence.

B. Stratégies de Synchronisation Basées sur les Verrous

La méthode la plus intuitive pour prévenir ces conflits est d'utiliser des **verrous** (*mutex*), lesquels garantissent que lorsqu'un thread est en train d'accéder ou de modifier une partie de la matrice $\boldsymbol{\omega}$, aucun autre thread n'y a accès. Deux approches se distinguent :

- **Verrous Globaux** : On peut appliquer un verrou global sur l'ensemble de la matrice $\boldsymbol{\omega}$ pendant la phase de mise à jour. Autrement dit, dès qu'un thread commence à mettre à jour une valeur $\omega_{i,j}$, il acquiert un verrou qui empêche toute autre modification sur $\boldsymbol{\omega}$ jusqu'à la fin de l'itération. Cette approche garantit une cohérence parfaite, mais elle réduit fortement le parallélisme, car les autres threads doivent attendre la libération du verrou global.

- **Verrous Fins** : Une approche plus raffinée consiste à appliquer des verrous sur des sous-ensembles de la matrice, par exemple sur chaque ligne ou sur des blocs spécifiques. Ainsi, si un thread met à jour la ligne i de la matrice, il acquiert un verrou sur cette ligne, tandis que d'autres threads peuvent mettre à jour des lignes différentes simultanément. Cette granularité fine permet une meilleure **concurrence** tout en assurant une cohérence locale, même si la gestion de multiples verrous peut devenir complexe dans un système à très haute dimension.

La mise en œuvre des verrous s'appuie souvent sur des primitives fournies par les bibliothèques de concurrence (par exemple, le module *threading* en Python ou les classes *std::mutex* en C++). Bien que cette approche soit efficace pour assurer la cohérence, elle peut introduire un surcoût en termes de **latence** et de **contention**, en particulier lorsque le nombre d'entités est grand.

C. Approches Lock-Free et Algorithmes Gossip

Pour surmonter les limitations imposées par les verrous, des approches **lock-free** ont été développées. Ces méthodes reposent sur l'utilisation d'opérations atomiques, telles que le **compare-and-swap (CAS)**, qui permettent de mettre à jour les valeurs de $\omega_{i,j}$ sans verrou explicite. L'idée est de garantir que chaque opération de mise à jour soit effectuée de manière indivisible, de sorte qu'un thread vérifie que la valeur de $\omega_{i,j}$ n'a pas changé depuis sa lecture avant d'y écrire la nouvelle valeur. Formulée de manière schématique, une mise à jour lock-free s'effectue ainsi :

$$\text{if } \omega_{i,j} = \text{old\_value then } \omega_{i,j} \leftarrow \text{new\_value (atomiquement).}$$

Cette approche permet une **concurrence** très élevée, particulièrement dans des environnements distribués ou sur des architectures multicœurs. Toutefois, la garantie de **convergence** en mode lock-free nécessite des conditions strictes, notamment un paramétrage très faible de η et τ pour limiter les déphasages. Par ailleurs, la complexité de l'analyse mathématique augmente, car la dynamique asynchrone introduit des retards variables et des mises à jour non uniformes, qui peuvent, dans certains cas, conduire à des oscillations accrues.

Une autre méthode est l'utilisation d'algorithmes **Gossip-based**, dans lesquels chaque nœud partage ses mises à jour de manière probabiliste et locale avec ses voisins. Cette approche permet de propager l'information de manière distribuée, avec des garanties de convergence sous certaines conditions sur le réseau de communication. Cependant, les algorithmes de type Gossip introduisent également des retards et des incohérences temporaires qui doivent être absorbés par la dynamique globale du SCN.

D. Impact sur la Convergence et la Stabilité Globale

Le choix entre des verrous, des approches lock-free ou des algorithmes Gossip influence directement la **convergence** du SCN. En mode synchrone, la garantie de convergence est aisée à démontrer grâce à l'hypothèse que toutes les mises à jour se font sur une version immuable $\omega(t)$. Le double-buffering, qui consiste à utiliser une matrice $\omega^{(t)}$ en lecture seule et une autre $\omega^{(t+1)}$ pour les écritures, est souvent utilisé pour préserver la **cohérence** tout en permettant une parallélisation. En mode asynchrone, la lecture de valeurs légèrement décalées peut induire des fluctuations dans les mises à jour. Pour assurer la stabilité, il est alors nécessaire d'ajuster le taux d'apprentissage η pour qu'il soit suffisamment faible, garantissant que même en présence de retards, l'opération de mise à jour reste contractante dans la norme choisie (i.e., $|1 - \eta\,\tau| < 1$).

L'utilisation de techniques de contrôle, telles que l'injection de bruit contrôlé (similaire à un recuit stochastique) ou l'application périodique de mécanismes de réindexation, peut aider à compenser les déphasages induits par une mise à jour asynchrone. Ainsi, l'architecture lock-free, bien que plus rapide en termes de débit de mise à jour, requiert une analyse plus fine des conditions de convergence, et peut parfois se traduire par des comportements oscillatoires ou des divergences temporaires qu'il faudra ensuite amortir par des correctifs algorithmique.

E. Conclusion

La mise à jour parallèle ou asynchrone des pondérations $\omega_{i,j}$ dans un SCN offre un gain substantiel en termes de performance lorsque le nombre d'entités est élevé. Cependant, cette accélération

s'accompagne de défis majeurs en matière de cohérence et de stabilité globale. L'emploi de verrous (globaux ou fins) permet d'assurer une cohérence stricte, mais au prix d'une réduction de la parallélisation. À l'inverse, les techniques lock-free et les algorithmes Gossip, tout en maximisant la rapidité d'exécution, introduisent des incohérences temporaires qui nécessitent un paramétrage prudent (réduction de η, ajustement de τ) pour garantir la convergence du système vers des attracteurs stables. Le choix final entre ces stratégies dépend largement de la taille du réseau, du degré de distribution souhaité et des contraintes de performance spécifiques à l'application. En pratique, une approche hybride, souvent basée sur le double-buffering dans un mode synchrone, s'avère être un compromis efficace, car elle assure une cohérence globale tout en tirant parti du parallélisme offert par les architectures modernes. Cette solution opérationnelle constitue le socle sur lequel repose le déploiement robuste et scalable d'un SCN dans des environnements multi-threadés ou distribués.

4.8.3. Transition

Les sections précédentes (4.8.1, 4.8.2) ont souligné l'importance de **l'architecture logicielle** (modularisation, gestion des ressources, parallélisme) pour soutenir la dynamique d'un **SCN** (Synergistic Connection Network). Il apparaît clairement qu'au-delà de la simple mise à jour $\omega_{i,j}$, il faut un cadre organisationnel robuste : comment instancier les modules de synergie, comment gérer l'inhibition, comment orchestrer la boucle de calcul en mode synchrone ou asynchrone, etc.

4.8.3.1. **Ce qu'on Pourrait Coder en Python/C++ pour Reproduire cette Dynamique en Quelques Centaines de Lignes**

Dans cette section, nous présentons une approche concrète pour implémenter la dynamique d'un **Synergistic Connection Network (SCN)** dans un langage de programmation tel que Python ou C++. L'objectif est de traduire la théorie développée dans les chapitres précédents en un code minimaliste mais complet, permettant de simuler la mise à jour des pondérations $\omega_{i,j}(t)$ et d'observer l'émergence de structures telles que des clusters, en quelques centaines de lignes. Cette démarche met en œuvre la règle de mise à jour additive ainsi que des mécanismes complémentaires comme l'inhibition et le clipping, tout en assurant une modularisation pour une meilleure maintenance.

A. Définition des Paramètres et Initialisation

Le système comporte n entités, chacune identifiée par un indice $i \in \{1,2,\dots,n\}$. La dynamique du SCN est régie par la règle de mise à jour :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right] - \gamma \sum_{k \neq j} \omega_{i,k}\,(t),$$

où :

- η représente le **taux d'apprentissage**,
- τ est le **facteur de décroissance** qui module la pénalisation d'un lien trop élevé,

- γ est le **coefficient d'inhibition** destiné à limiter la prolifération des connexions non sélectives,
- $S(i,j)$ désigne le score de **synergie** entre les entités i et j.

Le premier module du code consiste à définir ces paramètres et à initialiser la matrice de pondérations $\boldsymbol{\omega}(0)$. Par convention, on initialise $\omega_{i,j}(0)$ à des valeurs faibles (par exemple, un bruit aléatoire autour de zéro) pour simuler l'état initial d'un système non organisé.

En Python, on utilisera la bibliothèque NumPy pour manipuler efficacement les matrices. Par exemple :

```
import numpy as np

# Nombre d'entités
n = 15

# Paramètres globaux
eta = 0.05      # Taux d'apprentissage
tau = 1.0       # Facteur de décroissance
gamma = 0.02    # Coefficient d'inhibition
T_max = 300     # Nombre d'itérations

# Initialisation de la matrice des pondérations, avec un léger bruit aléatoire
np.random.seed(42)
omega = np.random.uniform(0.0, 0.05, (n, n))
np.fill_diagonal(omega, 0.0)

# Initialisation de la matrice de synergie S (à définir selon le contexte)
S = np.zeros((n, n))
# Par exemple, pour simplifier, on peut définir S(i,j) statique selon des critères prédéfinis
# Ici, S est symétrique et les valeurs sont entre 0 et 1.
for i in range(n):
    for j in range(n):
        if i != j:
            S[i, j] = np.random.uniform(0.1, 0.8)
```

B. ***Implémentation de la Boucle de Mise à Jour***

La dynamique d'auto-organisation est réalisée via une boucle itérative. À chaque itération t, la mise à jour des pondérations se fait de manière synchrone sur l'ensemble des paires (i,j). Afin d'éviter les conflits de lecture-écriture, on utilise le **double-buffering** : la matrice $\omega(t)$ est lue en lecture seule, tandis que les nouvelles valeurs sont écrites dans un buffer temporaire ω_{next}.

La mise à jour additive est alors formulée comme suit :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right] - \gamma \sum_{k \neq j} \omega_{i,k}\,(t).$$

Le pseudo-code Python ci-dessous illustre ce processus :

```
# Liste pour enregistrer l'évolution de la moyenne des pondérations (pour monitoring)
omega_means = []

# Boucle principale de mise à jour
for t in range(T_max):
    omega_next = np.copy(omega)
    for i in range(n):
        for j in range(n):
            if i != j:
                # Calcul de la variation (delta) de omega[i,j]
                delta = eta * (S[i, j] - tau * omega[i, j])
                # Application de l'inhibition : somme des pondérations de la ligne i sauf pour j
                inhibition = gamma * (np.sum(omega[i, :]) - omega[i, j])
                # Mise à jour additive
                omega_next[i, j] = omega[i, j] + delta - inhibition
                # Clipping pour garantir que omega reste positif
                omega_next[i, j] = max(0.0, omega_next[i, j])
    omega = omega_next
    # Enregistrement d'une statistique pour monitoring
    omega_means.append(np.mean(omega))

# À la fin de la boucle, omega contient la configuration finale du SCN.
```

Ce code, bien que succinct, encapsule le cœur de la **dynamique auto-organisée** du SCN. On observe que la mise à jour est effectuée de manière synchrone via un double-buffering, assurant la cohérence de la lecture de $\omega(t)$ pendant toute l'itération.

C. Visualisation et Analyse des Résultats

Pour évaluer la **convergence** et la **formation des clusters**, il est utile de visualiser l'évolution de la matrice ω et des indicateurs tels que la moyenne ou la variance des pondérations. En Python, on peut utiliser des bibliothèques telles que Matplotlib pour tracer des graphiques en temps réel ou après simulation :

```
import matplotlib.pyplot as plt

# Tracé de la moyenne des pondérations au cours des itérations
plt.figure(figsize=(8, 5))
plt.plot(omega_means, label='Moyenne de $\omega(t)$')
plt.xlabel('Itérations (t)')
plt.ylabel('Valeur moyenne de $\omega$')
plt.title('Convergence de la matrice de pondérations')
plt.legend()
plt.grid(True)
plt.show()
```

De plus, pour une analyse plus fine, on peut générer une **heatmap** de la matrice ω à la fin de la simulation pour identifier visuellement les clusters :

```
import seaborn as sns

plt.figure(figsize=(10, 8))
sns.heatmap(omega, cmap='viridis', annot=True, fmt=".2f")
plt.title('Heatmap finale de la matrice $\omega$')
plt.xlabel('Entités')
plt.ylabel('Entités')
plt.show()
```

Ces visualisations permettent de confirmer que les pondérations se sont organisées en clusters stables, conformément aux attentes théoriques.

D. Extensions Possibles et Adaptations en C++

La même logique de mise à jour peut être implémentée en **C++** en utilisant, par exemple, la bibliothèque STL et des structures de données adaptées (comme *std::vector<std::vector<double>>* pour représenter ω). La boucle de mise à jour sera similaire à celle présentée en Python, avec des boucles *for* imbriquées pour parcourir les indices i et j. Voici un exemple schématique en C++ :

```
#include <vector>
#include <iostream>
#include <algorithm>
#include <cstdlib>

// Définition des paramètres
const int n = 15;
const int T_max = 300;
const double eta = 0.05;
const double tau = 1.0;
const double gamma = 0.02;

// Fonction pour initialiser la matrice avec un bruit faible
std::vector<std::vector<double>> initializeMatrix(int n) {
    std::vector<std::vector<double>> omega(n, std::vector<double>(n, 0.0));
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (i != j) {
                omega[i][j] = ((double) rand() / RAND_MAX) * 0.05;
            }
        }
    }
    return omega;
}
```

```
// La fonction principale de mise à jour
void updateOmega(std::vector<std::vector<double>> &omega, const std::vector<std::vector<dou
ble>> &S) {
    int n = omega.size();
    std::vector<std::vector<double>> omega_next = omega;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (i != j) {
                double sum_inhibition = 0.0;
                for (int k = 0; k < n; ++k) {
                    if (k != j) {
                        sum_inhibition += omega[i][k];
                    }
                }
                double delta = eta * (S[i][j] - tau * omega[i][j]) - gamma * sum_inhibition;
                omega_next[i][j] = std::max(0.0, omega[i][j] + delta);
            }
        }
    }
    omega = omega_next;
}

int main() {
    // Initialisation des matrices omega et S
    std::vector<std::vector<double>> omega = initializeMatrix(n);
    std::vector<std::vector<double>> S(n, std::vector<double>(n, 0.0));
    // Remplissage simple de S, par exemple avec des valeurs aléatoires entre 0.1 et 0.8
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (i != j) {
                S[i][j] = 0.1 + ((double) rand() / RAND_MAX) * 0.7;
            }
        }
    }

    // Boucle de mise à jour
    for (int t = 0; t < T_max; ++t) {
        updateOmega(omega, S);
    }

    // Affichage final de la matrice omega
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            std::cout << omega[i][j] << " ";
        }
        std::cout << std::endl;
    }
```

```
    return 0;
}
```

Ce code C++ illustre la structure de base du pipeline de mise à jour, de la même manière que la version Python, et démontre que la **logique** sous-jacente est indépendante du langage de programmation. La modularité du code permet ensuite d'y intégrer facilement des mécanismes de **visualisation** ou d'**exportation** des résultats pour une analyse plus poussée.

*E. **Conclusion***

L'implémentation d'un SCN dynamique à l'aide d'un code en Python ou en C++ repose sur un ensemble de modules bien définis : l'initialisation des matrices de synergie et de pondérations, la boucle itérative de mise à jour appliquant la règle

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big] - \gamma \sum_{k \neq j} \omega_{i,k}\,(t),$$

et enfin des étapes de régulation telles que le clipping et la sparsification. Cette approche permet de reproduire, en quelques centaines de lignes de code, la dynamique auto-organisée décrite théoriquement dans le cadre d'un **Deep Synergy Learning**. La simplicité conceptuelle de la mise à jour est ainsi transformée en une architecture logicielle modulaire, facilitant la maintenance, l'extension à des systèmes plus vastes et l'intégration dans des frameworks multi-agents ou robotiques. Les exemples de pseudo-code présentés en Python et en C++ servent de point de départ pour des développements plus avancés, dans lesquels l'injection de visualisations, la gestion dynamique des paramètres et la persistance des données viendront enrichir l'ensemble et permettre des expérimentations en temps réel ou sur de grands volumes de données.

Voici une proposition détaillée et approfondie, rédigée dans un style académique et structuré, qui intègre de nombreuses formules mathématiques et explications approfondies.

4.8.3.2. Les Concepts de ce Chapitre 4 (Mise à Jour, Inhibition, Clusters Émergents) s'Insèreront dans une Architecture Logicielle Solide

L'ensemble des notions développées dans le chapitre 4 – notamment la dynamique de mise à jour des pondérations $\omega_{i,j}(t+1)$, les mécanismes d'inhibition (locaux ou globaux) et l'émergence des clusters – doit être opérationnalisé dans une infrastructure logicielle modulaire pour permettre un déploiement à grande échelle et une maintenance aisée. Le passage de la théorie à la pratique exige la conception d'un framework où chaque composant de la dynamique est isolé sous forme de modules interchangeables. Cette approche facilite non seulement l'évolution et l'extension du code, mais également l'intégration de nouvelles méthodes de calcul ou de nouveaux algorithmes de contrôle. Nous détaillons ci-après les axes majeurs de cette intégration.

*A. **Intégration Modulaire des Composants de la Dynamique***

Dans un premier temps, il convient de séparer distinctement les responsabilités fonctionnelles de la dynamique du SCN. On distingue principalement trois modules :

- **Module de Calcul de Synergie** : Ce module est chargé de déterminer la fonction $S(i,j)$, qui représente la qualité ou l'intensité de la synergie entre deux entités $\mathcal{E}_i$ et $\mathcal{E}_j$. La fonction $S(i,j)$ peut être définie à partir de critères sub-symboliques (par exemple, la similarité cosinus ou une distance gaussienne entre embeddings) ou symboliques (compatibilité de règles ou axiomes). On peut formuler, pour une approche hybride, la synergie comme

$$S_{\text{hybrid}}(i,j) = \alpha\, S_{\text{sub}}\big(\mathbf{x}_i, \mathbf{x}_j\big) + (1-\alpha)\, S_{\text{sym}}\big(R_i, R_j\big),$$

  où $\alpha \in [0,1]$ module l'importance relative des composantes. Ce module est conçu pour être interfaçable avec différents types de données et peut être mis à jour indépendamment, par exemple en recalculant périodiquement les embeddings ou en ajustant les règles symboliques.

- **Module de Mise à Jour des Pondérations** ω : Ce module encapsule la règle d'auto-organisation appliquée à la matrice $\boldsymbol{\omega}$. Qu'il s'agisse d'un schéma additif classique

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\, \omega_{i,j}(t)\big],$$

  ou d'une version multiplicative (par exemple,

$$\omega_{i,j}(t+1) = \omega_{i,j}(t)\exp\big(\eta\,\big[S(i,j) - \tau\, \omega_{i,j}(t)\big]\big),$$

  ce module reçoit en entrée la matrice S issue du module précédent, ainsi que les paramètres η et τ, et produit la nouvelle matrice $\boldsymbol{\omega}(t+1)$. L'architecture modulaire permet de remplacer aisément la règle de mise à jour en modifiant uniquement le code de ce module, sans impacter le calcul de la synergie.

- **Module de Régulation (Inhibition, Sparsification, Clipping)** : Pour éviter une croissance non contrôlée des pondérations et pour favoriser la formation de clusters cohérents, il est souvent nécessaire d'introduire des mécanismes d'inhibition et de clipping. Par exemple, l'ajout d'un terme inhibiteur tel que

$$-\gamma \sum_{k \neq j} \omega_{i,k}\,(t)$$

  dans la mise à jour permet de pénaliser la distribution excessive de liens par une entité. Ce module peut être implémenté soit comme une étape intégrée dans le module de mise à jour, soit comme une étape postérieure appliquant un filtrage sur la matrice $\boldsymbol{\omega}$. Le recours à une méthode de clipping, où l'on borne $\omega_{i,j}(t)$ à une valeur maximale $\omega_{\max}$, peut également être intégré ici pour éviter des valeurs aberrantes. Cette modularisation garantit qu'en cas de besoin, le comportement de régulation pourra être ajusté sans modifier les autres composants du système.

Ces modules interagissent au sein d'un **pipeline** orchestré par un **engine** central, dont le rôle est de gérer la séquence d'exécution, la synchronisation (via par exemple un double-buffering pour garantir la cohérence) et la persistance des données entre les itérations. Le code principal de ce

pipeline appelle successivement le module de calcul de synergie, puis le module de mise à jour des pondérations, et enfin le module de régulation, avant de stocker le résultat ou de déclencher une phase de visualisation.

B. Gestion Dynamique et Paramétrage

Une des forces de l'architecture modulaire est la capacité à gérer dynamiquement les paramètres η, τ et γ. Un **Param Manager** est ainsi mis en place, centralisant la configuration de ces valeurs. Ce gestionnaire peut, par exemple, surveiller la convergence en calculant à chaque itération des indicateurs tels que la norme $\| \boldsymbol{\omega}(t+1) - \boldsymbol{\omega}(t) \|$ ou la variation de l'énergie potentielle $\mathcal{J}(\boldsymbol{\omega})$. Si une oscillation excessive est détectée, le Param Manager peut alors automatiquement réduire η (ou augmenter τ) pour stabiliser la dynamique. La flexibilité apportée par ce module permet non seulement d'ajuster les paramètres de manière adaptative en fonction des données entrantes, mais également de tester divers schémas de mise à jour sans réécriture globale du code.

C. Persistance et Interface d'Intégration

La persistance des données dans un SCN est cruciale, surtout lorsqu'on déploie le système sur des volumes de données conséquents ou sur des plateformes distribuées. Une fois que la boucle de mise à jour est exécutée, la matrice $\boldsymbol{\omega}$ doit être sauvegardée pour permettre une reprise ultérieure ou une analyse post-hoc. Ce processus s'effectue via des modules de sauvegarde qui peuvent écrire la matrice dans un format binaire optimisé (par exemple, en utilisant le format HDF5) ou en format sparse si la majorité des liens tend à être nulle. Par ailleurs, une API d'intégration permet de brancher le SCN au sein d'un framework plus large – par exemple, pour interfacer avec des simulateurs robotiques ou des systèmes de monitoring en temps réel. Ce type d'interface se traduit par des endpoints REST ou par l'injection de flux de données dans le SCN, facilitant ainsi son déploiement dans des environnements industriels.

D. Orchestration de la Dynamique par un Engine Central

Le cœur de l'implémentation repose sur une boucle maîtresse ou un **scheduler** qui orchestre les appels successifs aux modules. Ce scheduler, implémenté par exemple en Python via des bibliothèques de multi-threading (comme *multiprocessing* ou *concurrent.futures*), effectue les opérations suivantes à chaque itération t :

- Il appelle le module de **calcul de synergie** pour obtenir la matrice $S(t)$ ou pour la mettre à jour en fonction des nouvelles données.
- Il déclenche le module de **mise à jour de** ω en passant les paramètres courants et la matrice $S(t)$, puis recueille la nouvelle matrice $\omega(t+1)$ via un système de double-buffering afin d'assurer une lecture cohérente.
- Il active ensuite le module de **régulation** (inhibition et clipping) pour appliquer des ajustements supplémentaires sur la matrice $\omega(t+1)$.
- Enfin, il enregistre l'état actuel de $\omega(t+1)$ pour permettre des analyses ultérieures (par exemple, via des heatmaps ou des calculs d'indicateurs de modularité).

Cet ensemble d'opérations, en étant modulé et orchestré par un engine central, garantit que la **dynamique globale** du SCN reste stable, tout en permettant une adaptation rapide aux changements de paramètres ou à l'insertion de nouveaux modules.

E. Conclusion

L'architecture logicielle modulaire décrite ici transforme la théorie du SCN en un système robuste, extensible et maintenable. En séparant clairement le **calcul de synergie**, la **mise à jour de** ω et la **régulation** (via inhibition et clipping), le framework permet une substitution rapide de chaque composant en fonction des besoins spécifiques de l'application. La gestion dynamique des paramètres, la persistance des matrices et l'intégration d'un engine central pour orchestrer la boucle d'auto-organisation font de ce système un outil flexible, capable d'être intégré dans des environnements multi-agents, distribués ou industriels. Le **Chapitre 5** détaillera davantage ces aspects, en fournissant des spécifications d'API, des exemples de modules en C++ et en Python, ainsi que des études de cas sur l'intégration du SCN dans un framework plus large. Cette modularisation permet ainsi de préserver à la fois la **souplesse** théorique du SCN et la **stabilité** pratique nécessaire pour un déploiement réel, facilitant ainsi l'expérimentation et l'évolution continue du système.

4.9. Conclusion et Transition

Tout au long de ce **chapitre 4**, nous avons examiné en détail comment un **SCN** (Synergistic Connection Network) s'**auto-organise** et fait émerger des **clusters** cohérents via la mise à jour adaptative des pondérations $\omega_{i,j}$. Les divers mécanismes (inhibition, saturation, injection de bruit, parallélisation…) ont montré la **richesse** et la **flexibilité** de cette dynamique. Il est maintenant temps de faire la **synthèse** (4.9.1) et de préparer le passage vers les **chapitres** suivants (4.9.2 et 4.9.3) où l'on abordera l'architecture globale (Chap. 5) et des développements plus avancés (Chaps. 6, 7, 8).

4.9.1. Synthèse

Dans l'ensemble des développements qui précèdent, le **cœur** de la dynamique d'un **Synergistic Connection Network** (SCN) repose sur la mise à jour locale des pondérations $\omega_{i,j}(t)$ selon des règles inspirées du schéma $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big]$. Une telle relation, parfois agrémentée d'un terme d'**inhibition** ou d'un **recuit**, définit un processus d'**auto-organisation** qui tend à renforcer les liens $\omega_{i,j}$ pour lesquels la **synergie** $S(i,j)$ est jugée élevée, tout en affaiblissant ceux qui se révèlent peu utiles ou inconsistants. Il est apparu, au fil des sections, que cette mécanique engendre la **formation** de **clusters** cohésifs, c'est-à-dire des ensembles d'entités fortement connectées par des pondérations $\omega_{i,j}$ élevées, alors que les connexions inter-groupes demeurent faibles ou presque nulles.

La règle additive simple, $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big]$, se situe au fondement du processus. Dans un régime stationnaire, elle conduit souvent à un équilibre $\omega_{i,j}^{*} \approx S(i,j)/\tau$ pour chaque paire (i,j), à condition que les entités n'évoluent pas et que les paramètres restent constants. Les **variantes** multiplicatives ou inhibitrices enrichissent ce tableau, en introduisant soit des effets de proportionnalité (cas multiplicatif) soit des phénomènes de compétition plus marqués (cas d'une inhibition latérale ou globale). Il a également été mis en évidence que la présence de **recuit** ou l'injection de bruit pouvaient aider à franchir des barrières ou à sortir de minima locaux, favorisant un **cluster** plus pertinent à l'échelle globale.

Le **contrôle** des oscillations se révèle un aspect capital pour la stabilité. Un choix cohérent de η (taux d'apprentissage), τ (décroissance) et γ (inhibition) circonscrit la zone d'attraction évitant les emballements, tout en permettant une différenciation nette des liaisons fortes et faibles. Il subsiste néanmoins une possibilité de **multi-stabilité** lorsque plusieurs configurations de clusters satisfont la matrice de synergie **S**. Le système peut alors basculer vers l'un ou l'autre attracteur suivant la condition initiale ou les perturbations reçues au fil des itérations.

À l'issue de cette analyse, il apparaît clairement que la dynamique d'un SCN ne saurait se réduire à une suite d'équations isolées. Dans un contexte pratique ou à grande échelle, elle doit s'inscrire dans une **architecture logicielle** modulaire, où l'on distingue le **module** affecté au **calcul** de la synergie, le **module** chargé de la **mise à jour** ω, le **module** gérant l'**inhibition** (ou la sparsification), et enfin les composantes d'**observation** ou de **clustering** (détection de communautés, visualisation). Cette séparation en blocs distincts, qui sera explicitée plus avant dans le chapitre 5, facilite la **maintenance**, l'**évolution** et la **persistance** des données, tout en permettant

d'orchestrer de manière ordonnée la boucle itérative : mise à jour parallèle, double-buffer, logs, etc.

En conclusion, la **mise à jour** $\omega_{i,j}(t+1)$, complétée par les mécanismes d'**inhibition**, de **recuit** ou de **sparcification**, constitue le **noyau** d'un **SCN** permettant d'aboutir à des **clusters** stables ou à un régime oscillatoire si les paramètres le favorisent. La prochaine étape, telle que le décrit le **chapitre 5**, est de **concrétiser** cette dynamique dans un cadre logiciel structuré, garantissant la **cohérence** des opérations, la **scalabilité** face à un grand nombre d'entités et l'**interopérabilité** avec d'autres modules ou frameworks. Cette articulation entre la **théorie** (chapitre 4) et l'**architecture** (chapitre 5) rend possible l'**implémentation** efficace d'un SCN, depuis les principes de base jusqu'au déploiement à grande échelle.

4.9.2. Ouverture

Les développements du présent chapitre 4 ont présenté la **dynamique** d'un **Synergistic Connection Network** (*SCN*), en détaillant les principales règles de **mise à jour** $\omega_{i,j}(t+1)$ (qu'elles soient additives ou multiplicatives), l'**inhibition** sous diverses formes, les mécanismes de **recuit** ou de **sparsification**, ainsi que l'**émergence** de **clusters** et les questions de **parallélisation**. Ces éléments constituent les fondations d'un système d'**auto-organisation** robuste. Néanmoins, ils ne sont que la première étape vers un **DSL** (*Deep Synergy Learning*) pleinement intégré. Les chapitres ultérieurs, à commencer par le **Chapitre 5**, franchiront un palier supplémentaire en décrivant la manière dont cette logique s'inscrit dans une **architecture** modulaire et scalable, apte à gérer de gros volumes de données, à s'interfacer avec d'autres modules, et à évoluer vers des scénarios avancés.

A. Chapitre 5 : Architecture Logicielle d'un SCN

Le chapitre suivant s'attachera à montrer **comment** transformer la simple "boucle de mise à jour" $\omega_{i,j}(t+1)$ et ses règles (inhibition, recuit, etc.) en un **environnement logiciel** :

> **Organisation en modules**. Le cœur du système (mise à jour de ω, calcul de synergie **S**, stratégies d'inhibition ou de sparsification) sera découpé en **briques** clairement définies, chacune remplissant un rôle précis.
>
> **Persistance** de la matrice ω. Il s'agira notamment de gérer le stockage (sur disque, en RAM distribuée, ou sous format sparse) pour des matrices éventuellement très volumineuses, ainsi que de prévoir la reprise du SCN là où il s'était arrêté (enregistrant $\omega(t)$, les paramètres, etc.).
>
> **API et intégration**. On décrira les interfaces (fonctions, classes, endpoints) permettant de brancher un SCN dans un framework existant (multi-agent, robotique, deep learning, etc.), de recevoir les flux de données exogènes (évolution de la synergie **S**), ou d'exporter les clusters émergents vers d'autres composants.

Cette approche ***mature*** du SCN assure la **maintenance** du code, la possibilité de tester facilement de nouveaux modules (par exemple un module d'inhibition différent), et une **adaptation** aisée à des cas pratiques (robotique, streaming de données, etc.).

B. Chapitres 6, 7, et 8 : Extensions et Scénarios Avancés

La base théorique et algorithmique mise en place dans le chapitre 4 (dynamique, clusters, multi-stabilité) et concrétisée architecturalement dans le chapitre 5 trouvera des prolongements au sein des chapitres 6, 7 et 8 :

> **Chapitre 6** : on traitera de l'**apprentissage multi-échelle**, ou de la structure **hiérarchique** d'un SCN, où l'on combine différents niveaux de granularité (micro-clusters, macro-clusters) et où la **fractalité** ou l'organisation emboîtée jouent un rôle.
>
> **Chapitre 7** : on explorera les **algorithmes d'optimisation** plus complexes (au-delà de la simple règle additive) et les techniques d'**adaptation dynamique**, qu'il s'agisse d'ajuster automatiquement η ou τ au fil du temps, ou de concevoir des approches globales (recuit multi-nœuds, stratégies d'exploration).
>
> **Chapitre 8** : on se tournera vers la **fusion multimodale**, c'est-à-dire des SCN manipulant des entités décrites par plusieurs modalités (texte, image, son, capteurs divers). Cette extension mettra en évidence la **flexibilité** du DSL pour unifier différents canaux de synergie au sein d'une même architecture.

Conclusion

Les **idées clés** de ce chapitre 4 (règle de mise à jour, inhibition, stabilisation, clusters, parallélisation) ne demeurent pas à l'état de simples algorithmes théoriques. Elles se prolongent immédiatement en :

> **Chapitre 5**, qui décrit **comment** disposer les **modules** (mise à jour ω, calcul **S**, persistance, APIs) dans une **infrastructure** modulaire et extensible,
>
> **Chapitres 6, 7, 8**, qui approfondissent des **dimensions** plus avancées : multi-échelle, optimisation et adaptation dynamique, ou fusion multimodale.

Ainsi, la **dynamique** d'un SCN (avec ses phénomènes de **clusters** émergents et ses mécanismes de contrôle) s'enrichit dans les chapitres ultérieurs d'une **architecture** logicielle robuste et de **scénarios** pratiques plus complexes, faisant de l'auto-organisation synergique un outil **générique** et **puissant** pour l'**apprentissage** et la **coordination** distribuée.

4.9.3. Place du Chapitre 4

Ce chapitre a présenté la **dynamique** propre à un **Synergistic Connection Network** (SCN), en montrant de quelle manière la **synergie** $S(i,j)$, introduite au chapitre 2 et liée aux représentations d'entités décrites au chapitre 3, conduit à la **mise à jour** adaptative de $\omega_{i,j}(t)$. Ce quatrième volet joue ainsi un **rôle de charnière** entre les **fondements** conceptuels (chapitres 2 et 3) et les **perspectives** d'implémentation et d'extension (chapitres 5 à 8).

A. Passage des Concepts à la Dynamique Opérationnelle

Les deux premiers chapitres ont posé les **fondations** d'un SCN : la notion d'entités, de synergie $S(i,j)$, et de co-occurrences ou de similarités logiques ou sub-symboliques. Sans la **dynamique**

décrite ici, ces idées se limiteraient à une théorie ou une cartographie d'affinités statiques. Le chapitre 4 a explicité comment, itération après itération, les pondérations $\omega_{i,j}$ s'**auto-adaptent** en fonction de $\omega_{i,j}(t)$ et de $S(i,j)$. Les notions de **clusters** émergents, de **multi-stabilité**, de **rôle** des paramètres η, τ, γ et de **convergence** (ou oscillations) illustrent la mise en mouvement concrète des entités, faisant passer le SCN d'une théorie de la synergie à un **réseau vivant**.

Cette transition se manifeste par la **formule** de mise à jour de base

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big],$$

éventuellement agrémentée d'un terme d'**inhibition** compétitive, d'un **recuit**, ou d'autres variantes non linéaires. Ces mécanismes répondent directement à la volonté de concrétiser la synergie en des liaisons qui se renforcent ou s'affaiblissent au fil du temps, selon leur utilité mutuelle.

B. Mise en Relation avec le Chapitre 5

Le chapitre 5 abordera la **conception** d'une **architecture** logicielle modulaire, destinée à rendre cette dynamique exploitable à grande échelle ou dans des systèmes complexes. Les principes détaillés ici — tels que l'inhibition locale ou globale, les règles additives ou multiplicatives, les stratégies de **sparcification**, la détection de clusters (section 4.6), la parallélisation ou l'asynchronisme (section 4.8) — s'intègrent dans un "**engine**" de mise à jour, que le chapitre 5 décrira selon des modules de calcul de **S**, de mise à jour ω, de persistance ou de connexion avec des frameworks externes.

L'**insertion** de ces éléments dans une architecture modulaire permet de **séparer** clairement la fonction "calculer la synergie" (chap. 3) de la fonction "faire évoluer la matrice ω" (chap. 4). Le chapitre 5 montrera que cette discipline, impliquant parfois un double-buffer ou un mécanisme synchrone, évite bien des conflits ou pertes de cohérence. C'est grâce à de telles dispositions qu'il devient possible de **maintenir** ou de **faire évoluer** un SCN au fil du temps, éventuellement dans un contexte distribué ou multimodal.

C. Reliance aux Chapitres Avancés (6, 7, 8)

Les phénomènes d'**auto-organisation** décrits en ce chapitre sont aussi la **base** sur laquelle s'appuient les travaux ultérieurs :

– Le **multi-niveau** ou **multi-échelle** (chapitre 6), où l'on construit un SCN à plusieurs paliers hiérarchiques ;

– Les **algorithmes** d'optimisation et d'adaptation (chapitre 7), poussant plus loin la gestion de la multi-stabilité ou l'autoconfiguration de η, τ ;

– La **fusion multimodale** (chapitre 8), montrant que la mise à jour $\omega_{i,j}$ peut agréger simultanément plusieurs types de synergie (similitudes de vecteurs visuels, textes, sons), prolongeant la simple règle additive de base vers des combinaisons plus riches.

Ces développements avancés réutilisent la **dynamique** présentée dans le chapitre 4, parfois en l'agrémentant de mécanismes supplémentaires ou de variantes de la formule de mise à jour.

Conclusion

Le **chapitre 4** assure la **charnière** essentielle entre les **concepts** (chapitres 2 et 3) et la **réalisation** logicielle (chapitre 5) ainsi que les **extensions** (chapitres 6 à 8). Il a permis de voir comment la **synergie** se transforme, via des équations locales de mise à jour, en un **processus** d'auto-organisation conduisant à la **naissance de clusters**. Les règles d'**inhibition**, de **recuit** ou de **sparcification** assurent un contrôle précis du comportement de la dynamique, permettant d'éviter les effets négatifs (divergence, oscillations incontrôlées) et de stabiliser le système. Les résultats en attestent : de petits **clusters** émergent dans des cas simples, et des communautés plus complexes apparaissent dans des situations multimodales ou multi-agent.

Le chapitre suivant explicitera de manière détaillée la **structure modulaire** d'un SCN, ouvrant la voie à la mise en œuvre d'un environnement professionnel, évolutif et apte à intégrer des flux de données massifs ou permanents. Cette transition du **concret** algorithmique (chapitre 4) à l'**ingénierie** (chapitre 5) parachève la logique du **Deep Synergy Learning**, associant auto-organisation théorique et déploiement logiciel robuste.