Université de Paris VI
Laboratoire d'Oceanographie Dynamiqueet de
Climatologie (LODYC)
4 Place Jussieu
75252 Paris CEDEX 05

Univerité de Paris XII
Avenue Jean-Baptise Clément
93430 VILLETANEUSE

Mémoire de DEA d'intelligence Artificielle

Assimilation de données de traceur océanique : Une méthodologie Neuronale

Présenté par
Samir OUIS

sam_ouis@hotmail.com

Soutenu le 01 octobre 1999

Responsables de stage:

Université de Paris 13

Younes Bennani Maître de conférences, HDR

Paris 6- LODYC
Sylvie Thiria Professeur
Fouad Badran Maître de conférence, HDR
Laurent Memery Professeur

Année académique 1998-1999

Remerciements

Je tiens particulièrement à remercier Mme Sylvie Thiria, professeur de l'Université de Versailles, Mr Fouad Badran maître de conférence au CNAM, Younes Bennani maître de conférence à l'Université de Paris XIII et Laurent Memery chargé de recherche CNRS, mes responsables de stage, pour leur soutien, leur confiance, leurs apports sur le plan professionnel et scientifique, et surtout pour leur grande disponibilité.

Merci aussi à Joel Cheuoua, Ngan Tran, Carlos Mejia, Dominique Fraysinet et Christophe Menkes pour leur accueil, leurs conseils et leurs aides qui m'ont été précieuses.

Je tiens à remercier tous les membres du jury pour la caution qu'ils ont bien voulu apporter à ce travail.

Je remercie encore tout le personnel du LODYC au sein duquel j'ai pu m'intégrer et réaliser mon stage de DEA dans une ambiance des plus agréables.

Enfin, je remercie mes frères et soeurs ainsi que mes chers parents.

Encadrement

Environnement humain

Ce stage - mémoire a été réalisé sous la direction de Mme. S. THIRIA, M. F. BADRAN, M. Y. BENNANI et M. L. MEMERY au sein de l'équipe de recherche du LODYC (Laboratoire d'Océanographie Dynamique et de Climatologie). Ce mémoire est effectué dans le cadre du DEA "Intelligence Artificielle" à l'Université de Paris XIII au laboratoire LIPN et de l'Université de Paris VIII. Je remercie ces institutions qui m'ont permis d'effectuer ce travail de recherche.

Le LODYC est une Unité Mixte de Recherche (UMR 7617) dépendant du CNRS, département des Sciences de l'Univers, de l'Université Pierre et Marie Curie (Paris VI), et de l'ORSTOM. Il fait partie de l'Institut Pierre-Simon Laplace, fédération d'unités de laboratoires, et accueille du personnel de l'Université Versailles Saint-Quentin, de l'Université Paris VII et de Météo France. Le LODYC regroupe quelque 80 chercheurs, enseignants-chercheurs, ingénieurs, techniciens, administratifs et doctorants localisés à Jussieu.

La vocation première du Laboratoire est l'étude des processus dynamiques gouvernant la circulation océanique, la compréhension des mécanismes gouvernant l'évolution du système climatique terrestre où l'océan joue un rôle important.

Environnement technique

Sur le plan technique, j'ai travaillé sous un système UNIX relié au réseau de l'université de Paris VI. J'ai utilisé le logiciel SN28 qui est un outil de conception d'architecture et de simulation de fonctionnement neuronal. Ce logiciel s'appuie sur un langage Lisp Objet. On a utilisé le langage C++ pour la conception d'une modélisation d'un modèle de productivité marine pour valider notre modélisation, et le logiciel Matlab pour la constitution et le traitement des fichiers de données ainsi que pour la visualisation des résultats.

Table des matières

Chapitre 1

Introduction

Les capteurs satellitaires mesurant la couleur de l'océan dans une gamme étendue de longueur d'onde permettent d'obtenir des informations intéressantes sur le contenu en pigments du phytoplancton et la productivité marine. En deçà de la physique de la mesure qui s'est considérablement améliorée ces dernières années, l'inversion des signaux des capteurs multi canaux nécessite des algorithmes performants si l'on veut tirer le maximum d'informations.

En particulier, ces algorithmes doivent impérativement prendre en compte la non linéarité des phénomènes étudiés et les incertitudes de la mesure qui sont dues aux différents bruits de la mesure ou à la faiblesse de la modélisation (phénomènes non-observés).

Il est possible d'envisager l'assimilation des données de l'océan dans les modèles de productivité marine afin d'utiliser au mieux ces données dans le cadre de modèles opérationnels d'écosystèmes marins.

Notre but est de faire une étude prospective de recherche fondamentale qui va donc dans un premier temps simplifier à l'extrême les phénomènes physiques étudiés de manière à mettre au point une méthodologie. Ceci nous permettra de réduire la complexité des modèles de productivité marine ce qui simplifierait d'autant l'assimilation des données satellitales de couleur de l'océan.

Une première étude théorique a montré la faisabilité de remplacer la partie non-linéaire du modèle dynamique par un modèle neuronal.

Pour valider cette idée, nous avons implémenté une application qui nous a permis de voir la précision de cette nouvelle approche.

Dans un premier temps, nous présenterons le problème physique et nous expliciterons la modélisation proposée pour simplifier le modèle dynamique. Dans le troisième chapitre, nous détaillerons le problème de l'assimilation des données tous en précisant son utilisation dans le modèle adjoint que nous proposons.

Le quatrième chapitre nous permet de valider les réseaux de neurones qui ont été intégrés dans la modélisation à la place de la biologie.

Le cinquième chapitre représente la partie théorique de notre étude. Nous montrons comment la décomposition sous forme de graphe permet d'envisager une modélisation de l'adjoint d'un modèle complexe intégrant de la dynamique. Nous présenterons tous les algorithmes nécessaires à la mise en oeuvre de cet

adjoint.

Le sixième chapitre présente l'application du calcul de l'adjoint du modèle dynamique qui a été mis au point pour tester l'efficacité de la nouvelle modélisation.

Et nous finirons par une conclusion et le reste des travaux à réaliser.

Chapitre 2

Problème physique

Les processus cherchant à représenter les processus biogéochimiques dans l'océan ont un certain nombre de caractéristiques qui rendent les exercices de mise au point (paramétrisation) et/ou de validation très délicats. Les relations entre les différents niveaux de la chaîne trophique sont en effet fortement non linéaires. Ces relations sont en outre très souvent basées sur des formulations semi-empiriques, associées à des paramètres mal contraints et/ou très variables dans le temps et l'espace. D'un autre côté, au vu de la difficulté d'accès et de la complexité des mesures *in situ*, les données biogéochimiques dans l'océan sont très ponctuelles. Cependant, l'arrivée des données satellitales de couleur de la mer permet depuis quelques années d'avoir accès à une couverture synoptique globale de la distribution de la teneur en chlorophylle de la surface de l'océan. Tous ces points sont d'une part à l'origine d'une certaine spécificité des questions liées aux modèles biogéochimiques océaniques et d'autre part rendent nécessaire l'utilisation de techniques d'assimilation de données.

Contrairement aux applications classiques des techniques d'assimilation, cherchant à estimer l'état d'un système, ou à prévoir son état futur, une des questions fondamentales en biogéochimie de l'océan concerne l'estimation des paramètres (c-à-d modèle inverse). Ainsi, vu la complexité croissante des modèles, il est important de se demander quels sont les types de données qui peuvent contraindre effectivement la représentation d'un processus particulier. Par exemple, il n'est a priori pas évident de savoir jusqu'à quel échelon de la chaîne trophique les informations contenues dans les données satellitales de chlorophylle de surface peuvent être utiles pour estimer les paramètres du modèles. Maintenant nous allons traiter le problème simplifié, puis nous présenterons ses caractéristiques.

2.1 Equation de conservation

Les choix effectués sont basés sur le compromis suivant : simplifier au maximum le volet informatique du problème, tout en gardant les spécificités du couplage entre un modèle de circulation et un modèle biologique dans l'océan. Ces spécificités sont de deux types : le modèle biologique, contrôlant l'évolution des variables de base (concentrations en traceurs), dépend d'une manière non

linéaire de paramètres internes du modèle et des variables elles mêmes; le modèle dynamique (transport des traceurs) est linéaire par rapport aux variables de base et lie les points de l'espace entre eux à l'intérieure d'un domaine fini,[Levy,1996],[Loukos,1995]. Aussi, le modèle proposé est-il simple, et ne s'intéresse à l'évolution que d'un seul traceur (les nitrates). La dynamique s'applique sur une seule dimension (verticale) en ne prenant en compte que des phénomènes de mélange turbulent (diffusion). L'équation qui permet de présenter la dynamique est la suivante :

$$\frac{\partial N}{\partial t} - \frac{\partial}{\partial z}\left(Kz\,\frac{\partial N}{\partial z}\right) = Biologie \tag{2.1}$$

où N représente la concentration en nitrates, z la profondeur et Kz le coefficient de diffusion verticale turbulente.

Mais en fait, que signifie la biologie?

2.2 Biologie

Dans la couche de surface éclairée, les nitrates sont utilisés par la photosynthèse. Ce flux biologique s'écrit dans le modèle de la manière suivante :

$$P = \alpha \frac{N}{K_N + N}\,\frac{E}{K_E + E} \tag{2.2}$$

où P représente le passage de la matière inorganique dissoute (nitrates) à la matière organique (sous forme de phytoplancton), c-à-d qu'il représente le taux de photosynthèse ou la production primaire (puits en nitrates) .

α est un paramètre interne du modèle, ainsi que K_E et K_N.

Le terme $(\frac{N}{K_N+N})$ décrit la limitation de la photosynthèse par les nitrates : il varie entre 0 pour N faible à 1 pour N élevé. Le terme de la même forme contenant E (énergie lumineuse ou PAR) représente la limitation par la lumière (il varie en fonction du temps et de la profondeur). Si E_0 est l'énergie solaire visible journalière reçue à la surface de l'océan (énergie variable dans le temps, mais connue a priori), l'énergie à la profondeur z est donnée par l'équation suivante :

$$E(z) = a_r E_0 exp\left(\frac{-z}{\Lambda_r}\right) + (1 - a_r) E_0 exp\left(\frac{-z}{\Lambda_b}\right) \tag{2.3}$$

Où a_r représente la proportion de l'énergie solaire contenue dans les longueurs d'ondes absorbées très rapidement par l'océan (rouge), et les termes Λ_r et Λ_b sont les longueurs d'absorption de la lumière dans le rouge et dans les longueurs d'ondes plus courtes (bleu), qui se propagent plus profondément.

Le terme P représente donc le passage de la matière inorganique dissoute (nitrates) à la matière organique particulière (sous forme de phytoplancton, non représenté explicitement ici). Il y a un terme qui doit permettre de représenter

la transformation inverse, cette transformation, effectuée par l'intermédiaire des bactéries, est représentée ici d'une manière extrêmement simple par une loi de la forme suivante :

$$R = \rho \left(1 - \frac{E}{E + K_E}\right) \tag{2.4}$$

où R représente donc le passage inverse de P, autrement dit le passage de la matière organique en matière inorganique, c-à-d qu'il représente le taux de reminéralisation (source de nitrate), ρ est un paramètre interne du modèle.

On peut maintenant définir la quantité Biologie par :

$$Biologie = -P + R$$

On définit en fonction du temps et du niveau k une nouvelle quantité $N^*(k, n+1)$:

$$N^*(k, n+1) = N(k, n) + Biologie(k, n+1) \times \Delta t$$

i.e. que "Biologie" est calculé en fonction de $N(k, n)$ (et des paramètres internes du modèle). Pour passer de $N^*(k, n+1)$ à la valeur $N(k, n+1)$, il est nécessaire de faire intervenir la partie transport (mélange vertical) de l'équation de conservation des nitrates.

2.3 Dynamique

Si on reécrit l'équation (2.1) en précisant l'étape de temps n et la profondeur k, on aura :

$$\left(\frac{\partial N_k}{\partial t}\right) = \frac{N_k^n - N_k^{n-1}}{\Delta t} = \left(\frac{\partial}{\partial z}\left(K_z \frac{\partial N}{\partial z}\right)\right)_n + Biologie_n \tag{2.5}$$

On obtient :

$$N_k^n = N_k^{n-1} + \left(\frac{\partial}{\partial z}\left(K_z \frac{\partial N}{\partial z}\right)\right)_n \Delta t + Biologie_n \, \Delta t \tag{2.6}$$

Δt est le pas de temps.

La biologie est simple, car elle ne dépend que de N_k^{n-1} et bien évidemment des paramètres internes du modèle, tels que α et K_E. Ce n'est pas pareil pour la dynamique. Le schéma est implicite du fait qu'il nécessite l'utilisation des discrétisations au pas de temps n, c-à-d des concentrations que l'on calcule à toutes les profondeurs. Ceci demande alors l'inversion d'une matrice qui va être explicité en dessous. La discrétisation sur la verticale est la suivante :

Les d_k représentent l'épaisseur de la couche de N au niveau k, et les δ_k celle de la couche k_z au niveau k. Dans notre exemple, les valeurs de d et de δ ne

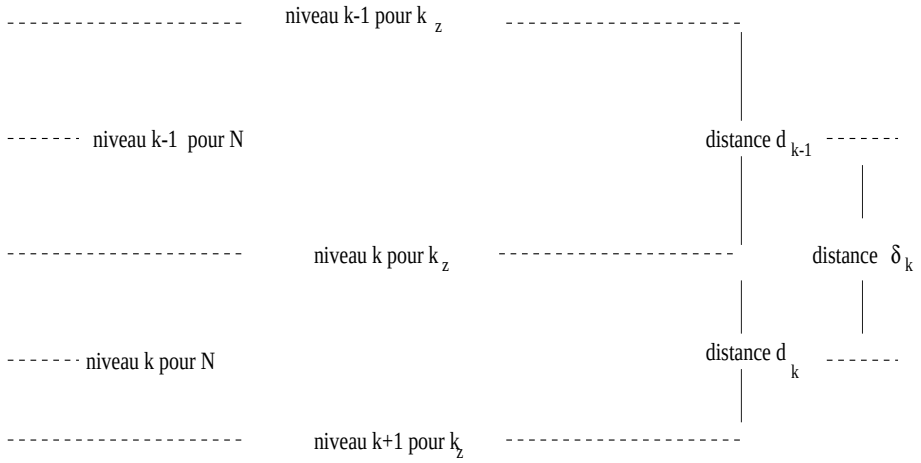

FIG. 2.1 –: *Schéma de discrétisation.*

dépendent pas de k et sont égales à 5 m.

On peut écrire la dynamique sous la forme suivante :

$$Dynamique_k^n = \left(\frac{\partial}{\partial z}\left(k_z\frac{\partial N}{\partial z}\right)\right)_k^n = \frac{\left(k_z\frac{\partial N}{\partial z}\right)_{k+1}^n - \left(k_z\frac{\partial N}{\partial z}\right)_k^n}{d_k} = \frac{\frac{k_{zk+1}^n\left(N_{k+1}^n - N_k^n\right)}{\delta_{k+1}} - \frac{k_{zk}^n\left(N_k^n - N_{k-1}^n\right)}{\delta_k}}{d_k}$$

Par suite, l'équation discrétisée s'écrit :

$$N_k^n = N_k^{n-1} + \left(\frac{k_{zk+1}^n\left(N_{k+1}^n - N_k^n\right)}{\delta_k} - \frac{k_{zk}^n\left(N_k^n - N_{k-1}^n\right)}{\delta_k}\right)\frac{1}{d_k}\Delta t + Biologie\left(N_k^{n-1}, E_k^{n-1}, Pa\right)\Delta t$$

où la biologie au niveau k dépend de la concentration en niveau k et au pas de temps précédent, de l'intensité lumineuse E et des paramètres internes Pa. Le schéma implicite de la diffusion implique donc une inversion de matrice, et l'équation précédente s'écrit alors :

$$-N_{k-1}^n\frac{k_{zk}^n}{\delta_k d_k}\Delta t + N_k^n\left(1 + \left(\frac{k_{zk}^n}{\delta_k d_k} + \frac{k_{zk+1}^n}{\delta_{k+1}d_k}\right)\Delta t\right) - N_{k+1}^n\frac{k_{zk+1}^n}{\delta_{k+1}d_k}\Delta t = N_k^{n-1} + Biologie\left(N_k^{n-1}, E_k^{n-1}, Pa\right)\Delta t \quad (2.7)$$

On peut écrire cette relation de la manière suivante :

$$A^n N^n = N^{n-1} + Bio^n$$
$$N^n = (A^n)^{-1} \left(N^{n-1} + Bio^n \right)$$

où N est le vecteur de dimension n_k (qui contient la concentration en nitrates pour chacun des niveaux considérés), ainsi que Bio. A est une matrice tridiagonale, représentant le rôle de la dynamique (diffusion verticale) dans l'équation de conservation des nitrates.

2.4 Intégration du modèle

Pour effectuer l'intégration temporelle à chaque pas de temps, on doit suivre les étapes suivantes :

1. Calculer la biologie Bio^n à partir des concentrations des nitrates au temps $n-1$ et des variables du modèle biologique,
2. Mettre à jour les champs de $N^{*n} = N^{n-1} + Bio^n$
3. Calculer les nouvelles concentrations au prochain pas de temps

$$N^n = (A^n)^{-1} N^{*n}$$

Les étapes 1 et 2 font intervenir des variables locales, i.e. des valeurs de concentrations et autre paramètres (lumière) à la profondeur k. Par contre, étant donné l'intervention de la matrice inverse $(A^n)^{-1}$, l'étape 3 relie tous les points sur la verticale entre eux. Ces étapes peuvent être représentées de la manière suivante :

Nous présentons maintenant la modélisation que nous avons effectué pour permettre par la suite une résolution neuronale de l'assimilation des données adoptées au problème que nous venons d'aborder.

Le travail consiste à bien faire apparaître pour chaque niveau et pas de temps les différentes transformations à prendre en compte pour réaliser le suivi du traceur nitrate au cours du temps.

Cette modélisation est représentée graphiquement (figure (2.2)) où chacune des boîtes représente une des fonctions que nous venons de présenter.

Sur cette figure, le temps se déroule selon les ordonnées et les profondeurs selon l'axe des abscisses. La représentation graphique des fonctions et des dépendances permettra d'envisager une résolution neuronale du problème complet d'assimilation. La fonction BIO sera modélisée par un réseau de neurones bien adapté à la modélisation des fonctions non linéaire et dont l'adjoint est facile à calculer pour les autres fonctions. Pour la dynamique, nous avons utilisé directement les équations que nous venons de présenter.

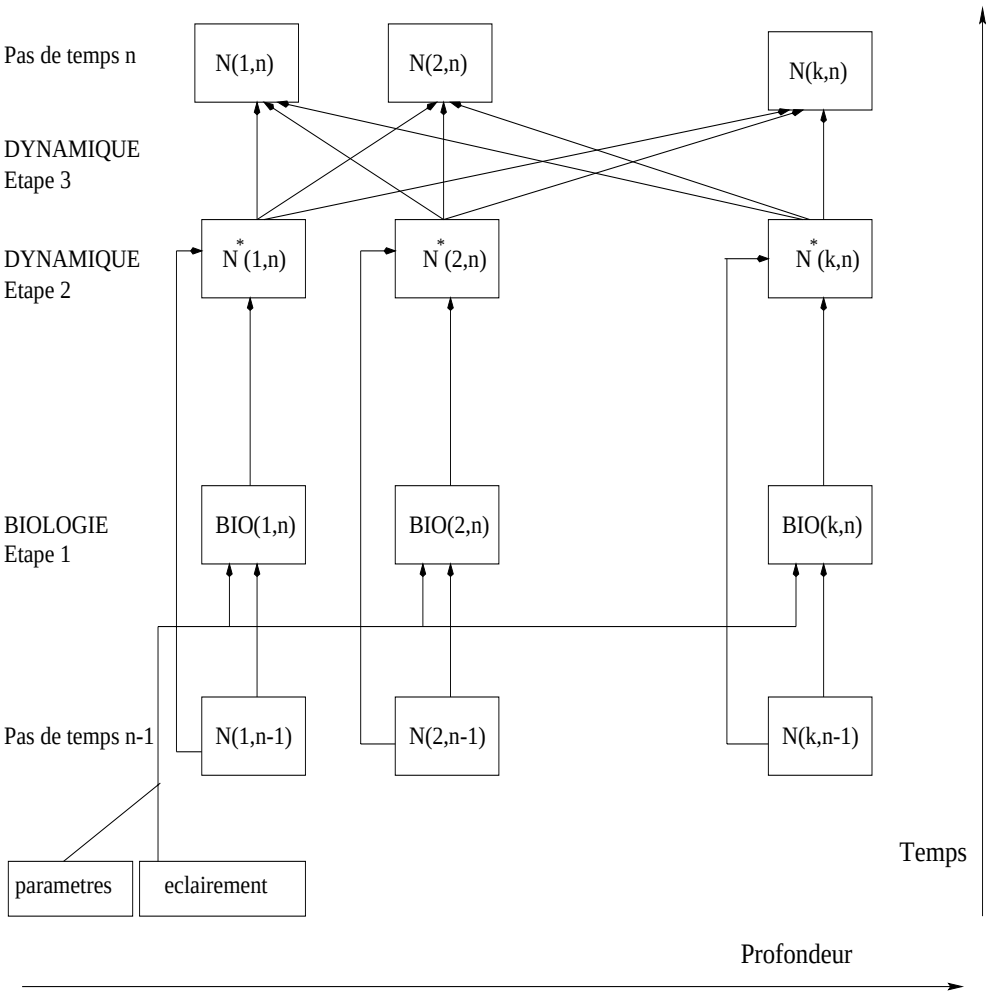

FIG. 2.2 –: *Modèle de couplage de la biologie avec la dynamique.*

Par contre il est nécessaire d'avoir des conditions aux limites à la surface et au fond pour la dynamique.

Pour la surface, le flux est nul, i.e. que la boîte de surface n'échange par mélange qu'avec la boîte du dessous. k est la $k^{\text{ème}}$ profondeur, dans la suite nous avons considéré 40 niveaux séparés par une distance de 5m entre chacun. Le pas de temps, Δt, utilisé dans la simulation est d'une heure.

Pour le fond, c'est plus compliqué, étant donné que la frontière est ouverte, donc il faut prendre en compte le niveau en dessous du 40^{eme} niveau.

Pour la surface, il n'y a pas de flux entre la boîte 1 et la boîte 0, d'où l'équation discrétisée s'écrit alors :

$$N_1^n \left(1 + \left(\frac{k_{z,2}^n}{\delta_2 d_1}\right)\delta t\right) - N_2^n \frac{k_{z,2}^n}{\delta_2 d_1}\Delta t = N_1^{n-1} + Biologie(N_1^{n-1}, E_1^{n-1}, P)\Delta t$$

Cette équation correspond à la première ligne d'une matrice tridiagonale, ceci est correct si on admet que les facteurs des concentrations en nitrates de la partie gauche de l'équation (2.7) correspondent à des valeurs de trois vecteurs constituant une matrice tridiagonale qui est la matrice A^n.

Pour le fond, c'est la même chose (correspond à la dernière ligne d'une matrice tridiagonale), mais il faut néanmoins prendre en compte les échanges avec la boîte du dessous (niveau 41). La relation précédente s'écrit alors :

$$\begin{aligned} &-N_{nz^{-1}}^n \frac{k_{z,nz}^n}{\delta_{nz} d_n z}\Delta t + N_{nz}^n \left(1 + \left(\frac{k_{z,nz}}{\delta_n z d_{nz}} + \frac{k_{z,nz+1}^n}{\delta_{nz+1} d_{nz}}\right)\Delta t\right) \\ &= N_n z^{n-1} + V_{nz+1}^n \frac{k_{z,nz+1}^n}{\delta_{nz+1} d_{nz}}\Delta t + Biologie\left(N_{nz}^{n-1}, E_{nz}^{n-1}, P\right) \end{aligned}$$

Le second terme de la partie droite correspond au "terme de forçage" ou de conditions au limites qui est la concentration en nitrate au niveau $n_z + 1$ (noté V). Cette concentration n'intervient que dans l'étape de la dynamique dans les conditions aux limites du fond, et joue le même rôle que les N^*, sauf que ça valeur est donnée et non calculée.

La prise en compte de ces deux conditions nous amène à modifier la modélisation qui est maintenant représentée par la figure (2.3). Bien entendu, nous n'avons représenté ici qu'un seul pas de temps et sans faire apparaître les fonctions de codages. Nous avons montré sur la figure (2.4) deux étapes de temps en affichant les fonctions de codage, ceci concerne la partie de propagation avant pour la prédiction des concentrations des nitrates alors qu'il faut calculer la fonction "coût" selon les observations présentes, ensuite il faut suivre le sens inverse des flèches sur la figure (2.4) pour calculer le gradient des deux paramètres α et K_E de telle sorte accumuler le gradient jusqu'a arriver au temps initial, ensuite il faut suivre la même démarche jusqu'à atteindre le seuil de la fonction "coût".

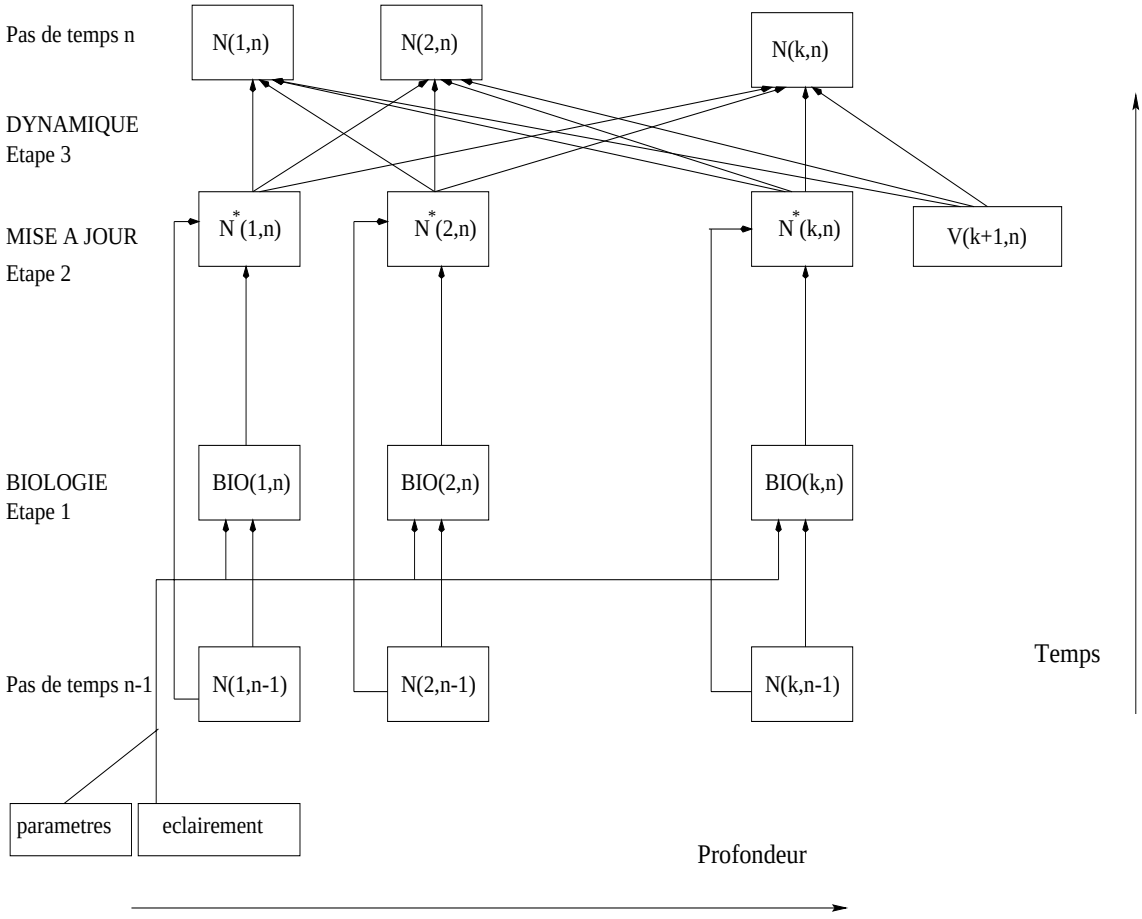

FIG. 2.3 –: *Modèle de couplage de la biologie avec la dynamique en prenant en compte le terme de forçage.*

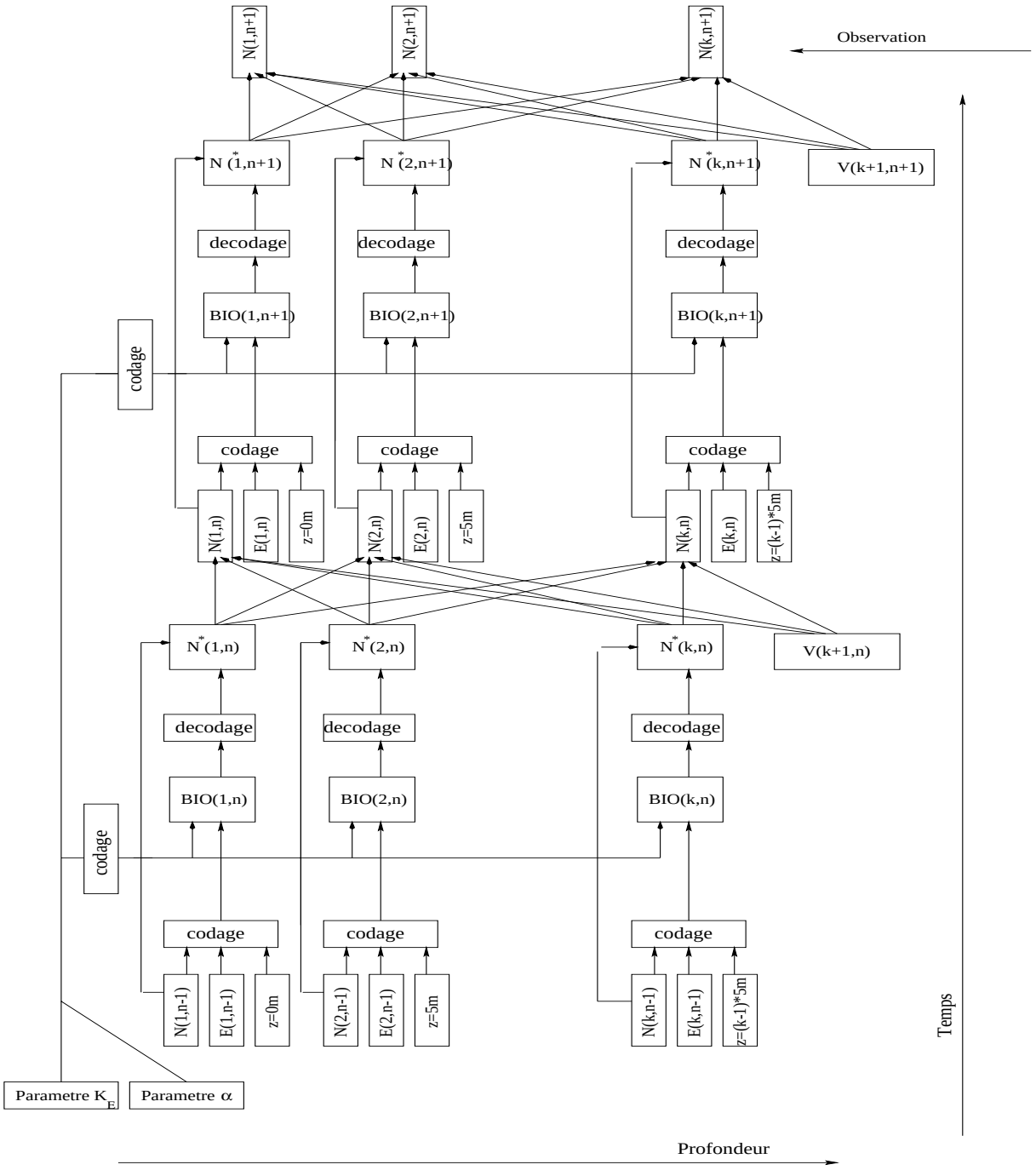

FIG. 2.4 –: *Modèle de couplage de la biologie avec la dynamique en prenant en compte le terme de forçage et le codage sur deux pas de temps.*

Chapitre 3

L'assimilation variationnelle

Les mouvements océaniques résultent de processus physiques qui concernent des échelles spatiales dont le spectre est très large.

Il en est de même pour les échelles temporelle. Les deux principaux outils d'étude de la circulation océanique sont d'une part les observations et de l'autre la théorie et les modèles numériques. La synthèse des deux approches doit tendre vers une représentation et une meilleure compréhension des phénomènes océaniques. Chaque moyen de mesure ne donne accès qu'à une composante de la circulation océanique, comme l'éclairage, avec un échantillonnage spatial et temporel donné, qui ne fournit pas une description complète de la circulation mais seulement "un point de vu".

Un modèle numérique d'océan est un outil de modélisation de la circulation générale ou d'étude de processus. Il est constitué par un ensemble d'équations différentielles d'évolution des différentes variables décrivant l'état de l'océan, Il donne une représentation spatiale et temporelle continue de la circulation, la connaissance des conditions initiales et des conditions aux frontières détermine entièrement la solution.

L'assimilation est une approche qui permet de conjuguer les connaissances théoriques contenues dans les modèles et les observations. Elle est utilisée pour différents objectifs. En océanographie, les principales applications sont :

1. l'estimation des paramètres d'un modèle physique,
2. l'estimation de l'état de l'océan à partir de données en utilisant des lois physiques comme information supplémentaire,
3. l'optimisation des positions des instruments d'un réseau d'observation.

Nous nous intéresserons ici au premier cas, dans le but de déterminer l'histoire d'un traceur de l'océan la plus en accord avec les observations sur toute la période où elles sont assimilées à l'aide d'un couplage de modèle biologique et dynamique, les paramètres estimés étant à tout instant influencés par les observations passées et futures existantes.

Le choix de la méthode d'assimilation dépend du but dans lequel on l'applique, ainsi les méthodes variationnelles sont intéressantes pour trouver une solution optimale à tout instant en accord avec les observations passées et

futures de l'intervalle d'assimilation alors que les méthodes séquentielles permettent d'accumuler les connaissances issues des observations passées pour avoir la meilleure estimation au dernier instant où les observations sont assimilées.

3.1 Inversion et assimilation

On désigne par problème inverse les problèmes de détermination des paramètres d'un système à partir d'observations de celui-ci, [Remy,1999]. Sa forme générale est la suivante :

Chercher les paramètres $\mathbf{x}$ décrivant le système observé tel que $\mathbf{y^{obs}} = \mathcal{H}(\mathbf{x}, \mathbf{p})$ (3.1)

où $\mathbf{y^{obs}}$ sont les observations, $\mathbf{p}$ est le vecteur des paramètres internes du modèle et $\mathcal{H}$ la fonction permettant de calculer l'équivalent des observations à partir des paramètres du système. On distingue le cas où l'on cherche une solution stationnaire, c'est à dire les paramètres décrivant le système observé à un instant donné. il s'agit alors d'inversion, du cas où l'on veut estimer l'évolution de ces paramètres, on parlera alors d'assimilation. Le but de l'inversion et de l'assimilation est d'estimer $\mathbf{x}$ (les paramètres du système couplé d'un modèle biologique et dynamique) à partir des observations tout en respectant des contraintes dynamiques.

3.1.1 Solution au problème inverse

Le but de l'inversion est d'estimer les paramètres d'un système à partir d'observations de ce dernier. Les modèles ne constituent qu'une représentation simplifiée du système physique observé, quant aux données elles contiennent des erreurs.

Le but de ce mémoire étant d'étudier l'assimilation de données. Nous allons dans ce paragraphe décrire une méthode de résolution de problème inverse, nous renvoyons à [Talagrand,1991] pour la résolution générale et les discussions sur le problème.

Solution des moindres carrés

L'approche aux moindres carrés, énoncée par Gauss en 1809, a pour but de minimiser la somme des erreurs quadratiques entre les observations et leur prédiction à partir du modèle. Cette approche est adaptée aux systèmes surcontraints car elle augmente le nombre des inconnues. On recherche le minimum de la fonction suivante, appelée fonction "coût" :

$$\mathcal{J} = (\mathbf{y^{obs}} - H(\mathbf{x}, \mathbf{p}))^* W^{-1} (\mathbf{y^{obs}} - H(\mathbf{x}, \mathbf{p})) \tag{3.2}$$

La solution est le vecteur $\mathbf{x^{est}}$ qui minimise les résidus au sens des moindres carrés. Dans cette équation W^{-1} représente la matrice de variance covariance des erreurs d'estimation, l'ajout de cette matrice de pondération des observations permet d'accorder plus d'importance aux données les plus précises.

3.1.2 Assimilation des observations

On peut prendre en compte la dépendance entre les variables du système à différents instants en ajoutant des équations d'évolution du système observé. Le modèle dynamique peut alors être vu comme un interpolateur spatial et temporel. La dynamique qu'il décrit induit des relations entre les différentes variables et l'information introduite localement est propagée en dehors des lieux et instants d'observations.

L'approche variationnelle nous permet d'estimer une solution globale qui vérifie au mieux l'ensemble des observations sur un intervalle de temps donné tout en suivant les équations du modèle. On peut chercher tout à la fin à estimer les variables d'entrée du modèle ($\mathbf{x}$) mais également des paramètres du modèle ($\mathbf{p}$).

Cette approche est comparable à celles des moindres carrés, mais l'espace dans lequel on cherche la solution $\mathbf{p}^{est}$ telle que $\mathcal{J}$ soit minimale, est restreint à l'ensemble des solutions du modèle dynamique d'évolution. Cette recherche se fait par ajustement des variables du vecteur de contrôle choisis parmi les paramètres du modèle, [Remy,1999].

Si $\mathbf{p}^{est}$ doit vérifier exactement les équations du modèle (c-à-d qu'il a une signification du point de vue modèle et éviter les cas impossibles), on dit qu'elles constituent une contrainte forte. La solution obtenue donne alors une description spatiale et temporelle complète, vérifiant au mieux les observations et est cohérente avec les équations d'évolution du modèle. Sinon il s'agit d'une contrainte faible. Lorsque le modèle constitue une contrainte forte, la solution minimisante est le champ lissé qui passe au plus près des observations tout en respectant la dynamique imposée par le modèle.

$$\mathcal{J} = \sum (\mathbf{y_{t_i}}^{obs} - H(\mathbf{x_{t_i}}, \mathbf{p_{t_i}}))^* W^{-1} (\mathbf{y_{t_i}}^{obs} - H(\mathbf{x_{t_i}}, \mathbf{p_{t_i}})) \tag{3.3}$$

$\mathbf{x_{t_i}}$ désigne l'etat du modèle à l'instant t_i, $\mathbf{p}$ désigne les paramètres à estimer du modèle, il intervient dans la fonction "coût" implicitement par l'intermédiaire des variables $\mathbf{x_i}$, $\mathcal{H}_i$ désigne la fonction d'observation tel que $\mathcal{H}_i(x_{t_i}) = y_{t_i}$ et $\mathcal{M}_i$ sont les équations du modèle telles que $\mathbf{x_{t_i}} = \mathcal{M}_i(\mathbf{x_{t_{i-1}}})$. La diagonale de W donne la pondération des observations les unes par rapport aux autres.

3.2 Présentation de la méthode variationnelle

Le principe des méthodes variationnelles est basé sur un calcul de sensibilité de la fonction coût, contenant l'écart aux observations, aux variations des paramètres du modèle. Pour un modèle de circulation océanique, la trajectoire est déterminée par les conditions initiales, les conditions aux frontière et des paramètres tels que les coefficients de diffusion. C'est en ajustant certaines variables choisies parmi celles-ci que l'on recherche une solution, ces variables sont appelées **paramètres de contrôle**. Leur valeur est déterminée de telle sorte que la somme quadratique des distances entre les observations réparties sur l'intervalle d'assimilation et la trajectoire du modèle soit minimale (figure 3.1).

La trajectoire optimale correspond au minimum de la fonctionnelle, appelée fonction "coût", mesurant la distance entre la trajectoire du modèle et les observations :

$$\mathcal{J} = \frac{1}{2}\sum_{i=0}^{T}(\mathbf{y_{t_i}} - \mathbf{y^{obs}_{t_i}})^*(\mathbf{y_{t_i}} - \mathbf{y^{obs}_{t_i}})$$

$\mathbf{y^{obs}}$ est un vecteur colonne contenant toutes les observations aux différents instants t_i de l'intervalle d'assimilation, $\mathbf{y}$ est leur équivalent exprimé en fonction des variables du modèle : $\mathbf{y_{t_i}} = \mathcal{H}_i(\mathbf{x_{t_i}}, \mathbf{p_{t_i}})$.

Cette minimisation doit être faite sous la contrainte de vérifier les équations du modèle.

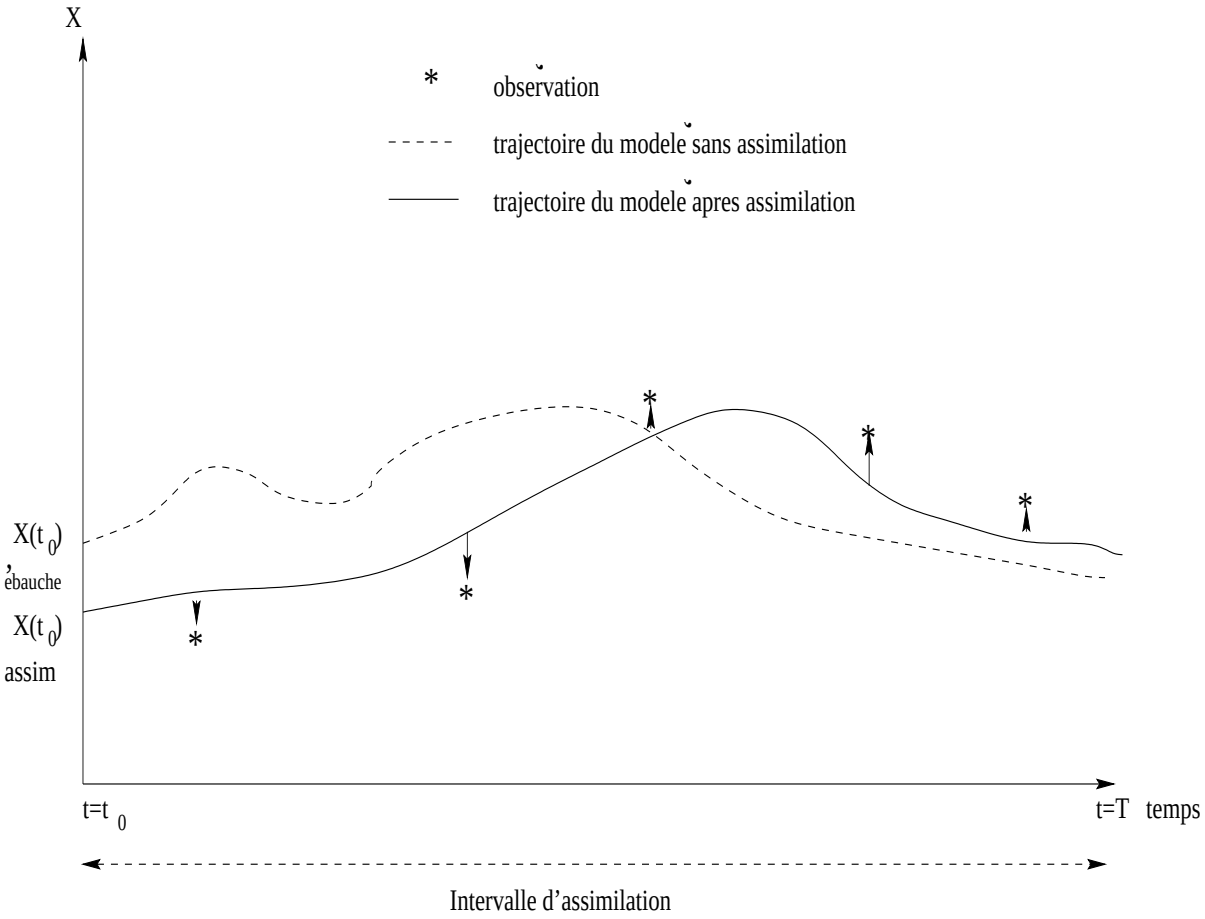

FIG. 3.1 –: *Schéma de principe de l'assimilation variationnelle : l'assimilation permet d'estimer une trajectoire du modèle qui passe au plus prés des observations (*) réparties sur tout l'intervalle du temps considéré, la minimisation de cet écart étant initialisée avec une solution a priori du modèle, l'ébauche.*

3.2.1 La technique adjointe

La fonction coût définie précédemment peut être écrite directement en fonction des variables du modèle $\mathbf{x}$ et pour des observations différentes de la manière suivante :

$$\mathcal{J} = \frac{1}{2}\sum_{i=0}^{T}(\mathcal{H}_i(\mathbf{x_{t_i}}, \mathbf{p}) - \mathbf{y_{t_i}}^{obs})^*(\mathcal{H}_i(\mathbf{x_{t_i}}, \mathbf{p}) - \mathbf{y_i^{obs}}) \tag{3.4}$$

Le gradient de la fonction coût s'écrit alors comme suit :

$$\left(\frac{\partial \mathcal{J}}{\partial \mathbf{p}}\right) = \sum_{i=0}^{T}\left(\frac{\partial \mathcal{H}_i(\mathbf{x_{t_i}}, \mathbf{p})}{\partial \mathbf{p}}\right)^* (\mathbf{y_{t_i}} - \mathbf{y_{t_i}^{obs}})$$

3.3 Mise en oeuvre de la méthode adjointe

La fonction que l'on veut minimiser (3.4) étant non-linéaire, la recherche du minimum est faite par un algorithme itératif qui détermine la direction de descente optimale par une méthode de gradient. La figure (3.2) montre les différentes étapes de la minimisation. On note $\mathbf{p}^{k=0}(t = 0)$ le point de départ de la minimisation, k désignant le numéro de l'itération. Nous prendrons des valeurs arbitraires significatives (i.e. moyenne théorique) comme point de départ de la minimisation, pour toutes les expériences que nous ferons.

1. Le modèle direct est intégré de $\mathbf{x}^k(t_0)$ à $\mathbf{x}^k(t_T)$: la somme des écarts entre les observations $\mathbf{y}_{t_i}^{obs}$ et leur équivalent dans le modèle $\mathcal{H}_i(\mathbf{x}_{t_i}^k)$ est calculée au fur et à mesure,

2. le modèle adjoint est intégré de l'instant $t = t_T$ à $t = t_0$, la résolution des équations adjointes permet de calculer la valeur du gradient de la fonction coût par rapport aux conditions initiales,

3. si cette valeur est suffisamment faible cela signifie que le minimum est atteint, la solution courante $\mathbf{p}^k(t_0)$ est la solution optimale. Sinon, un algorithme de descente détermine l'incrément $\partial\mathbf{p}$ à partir de la connaissance de $\mathcal{J}$ et $\nabla\mathcal{J}$: $\mathbf{p}^{k+1} = \mathbf{p}^k - \partial\mathbf{p}^k$.

Les deux principaux problèmes qui peuvent se poser pour l'utilisation de la méthode variationnelle sont l'existence de fonctions non différentiables dans le modèle et l'importance de non-linéarités qui rendent la convergence plus difficile.

Dans l'application qui nous intéresse le modèle biologique, contrôlant l'évolution des variables de base (concentrations en traceurs), dépend d'une manière non linéaire de paramètres internes du modèle et des variables elles mêmes, ceci va nous amener à modéliser le modèle biologique par un réseau neuronal.

On peut adapter l'étude théorique à notre problème en appliquant les modifications suivantes :

1. Le vecteur $\mathbf{x}_{t_i}$ représente les concentration en nitrate au temps t_i et le vecteur $\mathbf{y}_{t_i} = \mathbf{x}_{t_{i+1}}$ représente les concentrations des nitrates au temps t_{i+1} c-à-d celles obtenues à partir des concentrations au temps t_i,

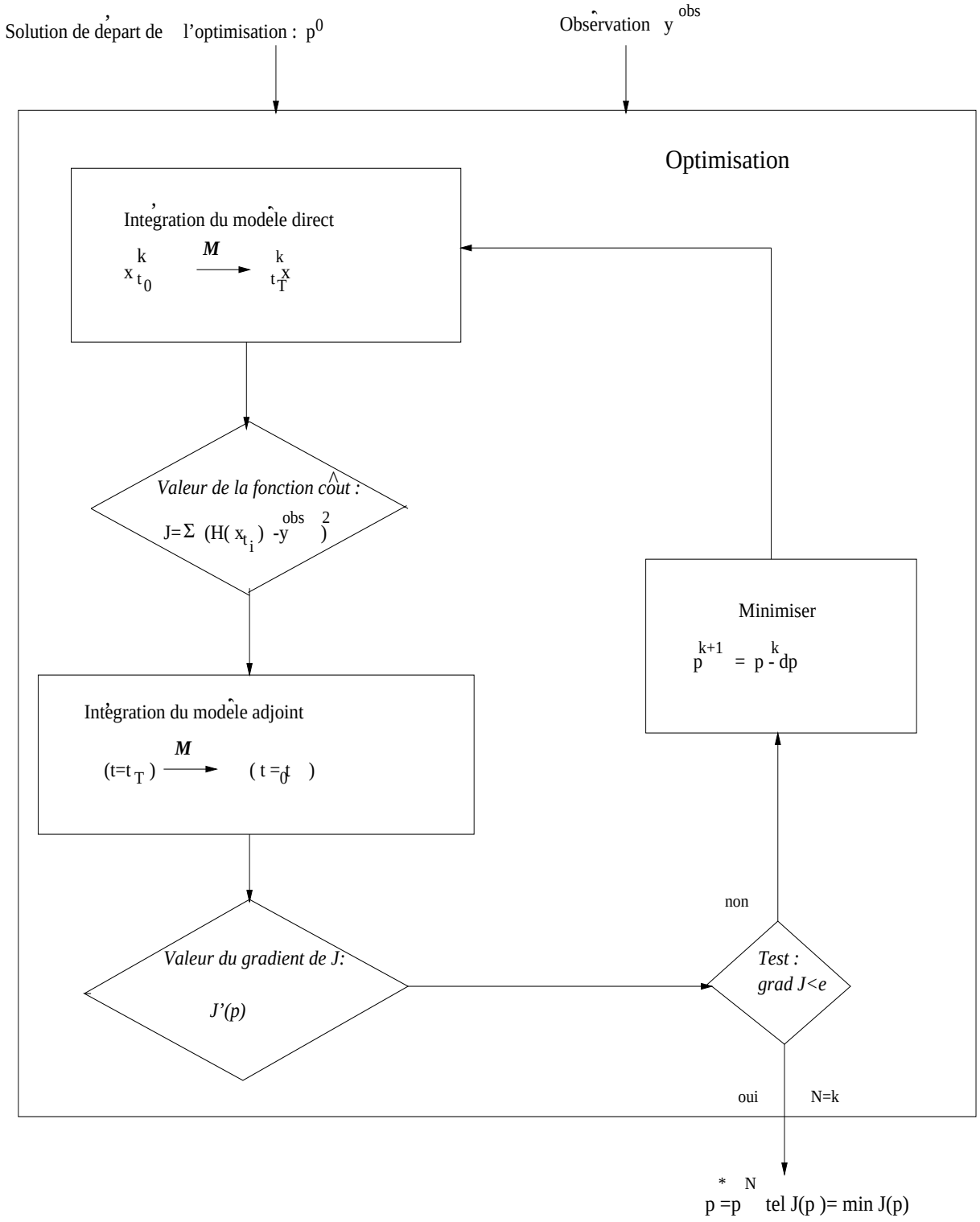

FIG. 3.2 –: *Organigramme de l'assimilation variationnelle.*

2. Les deux fonctions $\mathcal{H}_i$ et M_i sont identiques,

3. D'après le point précédent on déduit que les $\mathbf{x}$ sont calculés a partir de $\mathbf{p}$, donc on a
$$\frac{\partial \mathcal{H}_i(\mathbf{x},\mathbf{p})}{\partial \mathbf{p}} = \frac{\partial \mathcal{H}_i(\mathbf{x},\mathbf{p})}{\partial \mathbf{p}} + \frac{\partial \mathcal{H}_i(\mathbf{x},\mathbf{p})}{\partial \mathbf{x}} \times \frac{\partial \mathbf{x}}{\partial \mathbf{p}}$$

4. les paramètres qu'on veut estimer correspondent au deux paramètres évoqués dans le chapitre 2, α et K_e,

5. la fonction $\mathcal{H}$ correspond au modèle direct du couplage de la biologie avec la dynamique.

6. La matrice W des pondérations est prise égale à la matrice identité du fait qu'on considère que les observations des différents niveaux sont égales.

Chapitre 4

Modélisation Neuronale

4.1 Introduction

Dans ce paragraphe nous présentons la modélisation neuronale que nous avons effectué pour approximer la fonction "Bio" présentée au chapitre (2). Vu le manque de données expérimentales, nous avons procédé par simulation en utilisant les fonctions analytiques qui représentent le phénomène.

Il s'agit des équations présentées au chapitre (2). La modélisation a été effectuée en utilisant un perceptron multicouche. Le problème que nous avons eu à résoudre présente de nombreuses difficultés :

Les ordres de grandeurs qui interviennent (de l'ordre de 10^{-8} à 10^{-6}) nous ont obligé à utiliser différents codages et à séparer la modélisation en deux : un premier réseau permettant d'obtenir la biologie pour des profondeurs allant de 0 à $135m$, un second réseau modélisant la biologie de $140m$ à $195m$. Une telle division a fait apparaître des effets de bord que nous avons pu résoudre. Le premier paragraphe de ce chapitre présente des perceptrons multicouches que nous avons utilisé, le deuxième présente les deux réseaux "Bio" ainsi que les performances obtenues.

4.2 Introduction aux réseaux de neurones

Les premiers travaux sur les modèles neuronaux remontent aux années 1940 [McCulloch et Pitts, 1943]. Dans la dernière décennie, l'intérêt pour les réseaux de neurones a été rehaussé par l'évolution de l'informatique d'une part et d'autre part par la découverte de nouvelles techniques d'apprentissage levant les limitations des premiers modèles. Nous verrons, en particulier, l'algorithme de rétropropagation du gradient pour les perceptrons multicouches qui s'est construit en utilisant les méthodes de déscente du gradient [Rumelhart, 1986]. Un perceptron multicouche se caractérise par trois constituants de base: l'architecture du réseau, une règle d'activation et une règle d'apprentissage.

4.3 Architecture des réseaux multicouches

Un neurone se caractérise alors par trois concepts : son état, ses connexions et sa fonction de transition. Chaque connexion du neurone est affectée d'un poids de connexion. On note :

S : l'ensemble des états possible des neurones.

X_i : l'état d'un neurone i, ($X_i \in S$).

A_i : l'activité ou l'activation du neurone i.

W_{ij} : le poids de la connexion entre les neurones i et j.

l'actvité d'un neurone est calculée en fonction des états des neurones avec lesquels il a des connexions ainsi que des poids de ces dernières :

$$A_i = \sum_j W_{ij} X_j$$

l'état X_i du neurone i est une fonction de son activité A_i : $X_i = f(A_i)$ et f représente la fonction de transition.

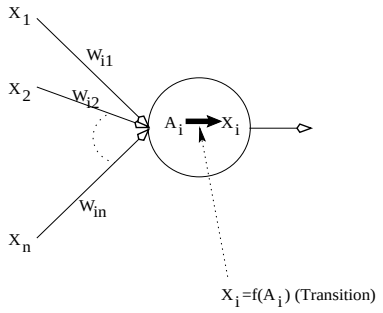

FIG. 4.1 –: *États du neurone*

4.3.1 Les fonctions de transition

L'introduction d'une fonction de transition dans les modèles neuronaux permet de rajouter un élément fondamental à ce type d'approche: la non-linéarité. Les fonctions de transition les plus utilisées dans le cadre des réseaux neuronaux sont la fonction identité, la fonction sigmoïde et la fonction gaussienne.

La fonction identité

L'état est alors calculé par : $X_i = A_i = \sum_j W_{ij} X_j$.

La fonction sigmoïde

C'est la plus utilisée car non seulement elle permet l'introduction des non-linéarites, mais c'est aussi une fonction continue, différentiable et bornée qui permet l'utilisation de valeurs continues pour les états. Elle a la forme :

$$f(x) = \frac{A(e^{Kx} - 1)}{e^{Kx} + 1} = A.tanh(\frac{K}{2}x)$$

La fonction exponentielle

L'état est calculé par $X_i = \exp(A_i) = \exp(\sum_j W_{ij} X_j)$.

4.3.2 Les couches

Une couche d'un réseau est un ensemble de plusieurs neurones. Les réseaux ou perceptrons multicouches comportent au moins trois couches de neurones : la couche d'entrée, les couches cachées et la couche de sortie. L'information est propagée vers l'avant.

4.4 Approximation de fonction

Intéressons nous maintenant à l'utilisation des perceptrons multicouches dans les problèmes de régression. On cherche des fonctions permettant d'exprimer la relation qui existe entre une variable $\vec{y}$ de dimension p et une variable d'entrée $\vec{x}$ de dimension n.

4.4.1 Apprentissage supervisé : rétro-propagation

On suppose par la suite que pour des réseaux multicouches, les couches d'entrée et les couches de sortie comportent respectivement n et p neurones, correspondant au nombre de variables qui interviennent dans le problème de modélisation mathématique. le modèle mathématique et le réseau multicouche sont alors représentés par la fonction :

$$F(\vec{x}, W) = \vec{y}$$

où W est la matrice des poids de connexion du réseau.

Les perceptrons multicouches permettent de réaliser des associations entre des signaux d'entrée et des signaux de sortie (qui représentent la variable à régresser). Les poids de connexion du réseau sont alors déterminés par un algorithme dit d'apprentissage qui les ajuste à partir d'un ensemble d'observations du phénomène étudié.

Erreur, fonction de coût :

Un des algorithmes permettant la détermination optimale de ces paramètres est l'algorithme de rétro-propagation du gradient. L'ensemble des observations (exemples d'apprentissage) utilisées pour la mise au point est appelé ensemble

d'apprentissage,[Rumelhart et al,1995]. L'algorithme de rétro-propagation agit par minimisation d'une fonction de coût. Il implique l'entraînement du réseau sur des exemples d'apprentissage pour lesquelles les valeurs correctes du phénomène $\mathcal{D} = \left\{ (\vec{x}, \vec{y})_i , i = 1, \ldots, N^{obs} \right\}$ sont connues. On peut alors évaluer l'erreur commise dans l'estimation (différences, au sens large, entre sortie estimée et sortie désirée connue) et procéder de manière itérative par un mécanisme de correction et d'ajustement de l'erreur en sortie. A chaque itération, le réseau ajuste les poids dans la direction qui réduit une erreur ou une fonction de coût. En un certain nombre d'itérations, les poids convergent graduellement vers leur valeur optimale.

Erreur en généralisation, erreur d'apprentissage :

L'intérêt d'un réseau de neurones réside aussi dans sa capacité à généraliser ses résultats à des exemples extérieurs à l'ensemble d'apprentissage. On parle d'***erreur en généralisation*** pour l'erreur effectuée sur tous les exemples, et d'***erreur d'apprentissage*** pour celle effectuée sur les exemples d'apprentissage. On ne connaît pas l'erreur en généralisation car on n'a pas toujours tous les exemples possibles. Ainsi, l'erreur que l'on minimise à l'aide de l'algorithme d'apprentissage est l'erreur d'apprentissage.

Erreur quadratique :

L'apprentissage revient ainsi à estimer les meilleurs paramètres du modèle au sens d'une fonction de coût. Il consiste à optimiser cette fonction de coût sur les exemples de la base d'apprentissage. La fonction de coût la plus fréquemment utilisée est l'erreur quadratique :

$$C(W) = \frac{1}{2} \sum_i \| \vec{y}_i - F(\vec{x}_i, W) \|^2$$

$\vec{y}_i$ est la réponse désirée fournie dans la base d'apprentissage $\mathcal{D}$

$\vec{x}_i$ est l'entrée associée dans la base d'apprentissage $\mathcal{D}$

$F(\vec{x}_i, W)$ est la sortie estimée par le réseau (W étant l'ensemble des poids)

Notons que cette expression est l'expression de l'erreur d'apprentissage qui n'est autre que la discrétisation de l'erreur en généralisation qui s'écrit :

$$C(W) = \frac{1}{2} \int \int \| \vec{y} - F(\vec{x}, W) \|^2 p(\vec{x}, \vec{y}) d\vec{x} d\vec{y}$$

$p(\vec{x}, \vec{y})$ étant la densité de distribution des observations $(\vec{x}, \vec{y})$

Il s'agit alors de minimiser une distance quadratique pour que les sorties estimées par le réseau se rapprochent le plus possible des valeurs attendues. La technique d'optimisation utilisée est une descente de gradient, c'est à dire que l'on va faire évoluer les poids W_{ij} dans la direction indiquée par le gradient de la fonction de coût C(W). Les poids seront au préalable initialisés à des valeurs aléatoires.

$$\Delta W_{ij} = -\varepsilon \frac{\partial C}{\partial W_{ij}}$$

Le gradient de l'erreur en sortie est donné par :

$$\frac{\partial C}{\partial F(\vec{x}_i, W)} f'(A_i)$$

(où f est la fonction de transition de la i^{eme} cellule de sortie et A_i son activation) est alors rétro-propagé dans toutes les couches du réseau.

Cet algorithme est facile à implémenter du fait de la simplicité du calcul des gradients des poids. Nous renvoyons à Bishop [Bishop,1995] pour la description complète des calculs des gradients.

4.5 Approximation de la biologie par un perceptron multicouche

Les études effectuées pour cette partie a eu lieu en deux étapes. La première avait pour but de simuler des données pour permettre l'apprentissage des réseaux de neurones, la difficulté de cette étape est due à l'inhomogénéité des valeurs à reproduire. Il fallait dans une analyse fine permettant de déterminer le seuil séparant deux ensembles homogènes, permettant d'obtenir une précision régulière pour la prévision de la Biologie.

La seconde étape avait pour but de coder les données et d'apprendre les réseaux de neurones selon une architecture bien choisie pour permettre ainsi une meilleure représentation du phénomène physique.

4.5.1 Validation croisée

Un des grands dangers des réseaux de neurones est le sur-apprentissage, qui peut arriver quand un réseau a trop de paramètres à estimer par rapport au nombre de données de l'ensemble d'apprentissage, ou quand on le laisse apprendre trop longtemps. Sur-apprentissage (overfitting) signifie que le réseau a "trop bien appris" et que finalement la fonction qu'il représente est un cas particulier. Ses performances, sur un ensemble de test possédant les mêmes caractéristiques statistiques que l'ensemble d'apprentissage, sont alors très mauvaises. Pour éviter cela, il faut savoir restreindre la puissance du réseau en limitant le nombre de neurones ou savoir arrêter l'apprentissage à temps [Murray et Smith, 1993]. Pour ces deux objectifs, on pratiquera une méthode intitulée *validation croisée*. Dans la première série d'expérience présentée, l'architecture

est fixée par la connaissance antérieure que l'on a du problème. On utilisera donc la validation croisée pour arrêter l'apprentissage à temps. Cette méthode consiste à tester, en permanence, les performances du réseau lors de l'apprentissage, sur l'ensemble d'apprentissage et sur un ensemble différent, ayant les mêmes caractéristiques, il est appelé *ensemble de validation*. On vérifie alors que le réseau généralise bien. On arrête l'apprentissage au maximum de performance sur l'ensemble de validation, donc juste avant le sur-apprentissage, ou alors bien avant, au moment où la distance RMS sur les deux ensembles paraît correcte. Bien entendu, les performances que nous présentons sont ensuite calculées à partir d'un ensemble de tests qui n'a participé ni à l'apprentissage ni à la validation.

4.5.2 Architecture utilisée

L'architecture utilisée durant cette étude (figure 4.2), pour faire correspondre le modèle biologique à partir des concentrations des nitrates et des paramètres internes, est choisie parmi plusieurs architectures essayées à l'aide de la même procédure de validation croisée, sachant qu'un perceptron multicouche avec une seule couche cachée peut estimer n'importe quelle fonction non-linéaire.

Nous avons finalement choisi un MLP à trois couches :

1. la couche d'entrée à 5 neurones qui permettent de faire figurer les variables de la fonction "Bio", il s'agit des concentrations des nitrates, l'énergie lumineuse, la profondeur, et des deux paramètres α et K_E qui vont être adaptés durant l'assimilation des données,

2. La deuxième couche (couche cachée) à 10 neurones,

3. et la dernière couche (couche de sortie) admet un seul neurone correspondant à la biologie,

Nous avons 71 paramètres à ajuster, étant donné que notre architecture affecte 5 neurones pour la couche d'entrée, 10 neurones pour la couche cachée et un seul neurone pour la couche de sortie, en ajoutant un neurone de seuil. Les fonctions de transitions sont toutes des sigmoïdes sauf pour les connexions entre la couche cachée et le neurone de sortie pour lesquelles on utilise la fonction linéaire $f(\Omega) = \Omega$.

L'apprentissage a été réalisé à l'aide du simulateur de neurones SN28 que nous décrirons ci-dessous.

4.5.3 Logiciel, ensemble d'apprentissage

Le simulateur SN28 est un outil de recherche et de développement d'application pour les réseaux de neurones, bien adapté à la simulation des réseaux de neurones multicouches et à leur apprentissage par rétro-propagation du gradient. Il présente une interface spécialisée en LISP et son noyau est en langage C.

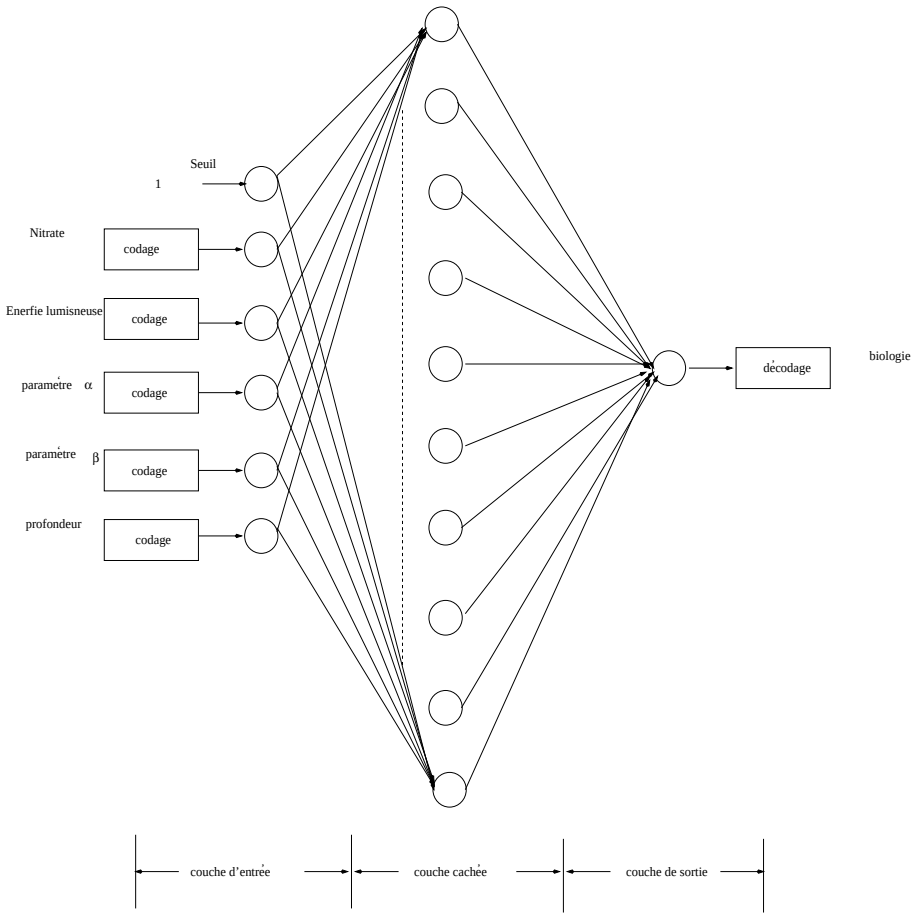

FIG. 4.2 –: *Architecture des deux réseaux de neurones.*

Vu qu'on a deux réseaux, on était contraint de créer deux bases d'apprentissage, une pour les profondeurs allant de 0 jusqu'à $140m$ et l'autre allant de 100 jusqu'à $220m$, pour éviter l'effet de bord, nous avons dépassé les frontières des deux tranches de profondeur. Nous avons donné la même importance à toutes les profondeurs. Afin d'obtenir une fonction bien régulière, nous avons échantillonné de manière régulière chaque profondeur et on a représenté chacune par 200 formes.

Nous avons, finalement obtenu pour le 1^{er} Réseau NN_1 allant de 0 à $140m$ un ensemble d'apprentissage de 5800 formes, et pour le second NN_2 allant de 100 à $220m$ un ensemble de 5000 formes. Les effets de bord on été atténués en générant pour chacun des réseaux des formes pour des profondeurs dépassant ces frontières.

4.5.4 Algorithme d'apprentissage et pré-codage

Pour la série d'apprentissage des deux réseaux, on utilise une méthode de gradient adaptatif ou stochastique [Widrow et Hoff, 1960]. Il s'agit d'une règle de gradient suivant la plus grande pente où les paramètres du réseau W_{ij}, sont modifiés non pas en tenant compte, à chaque itération, de l'erreur globale réalisée sur l'ensemble d'apprentissage, mais après chaque représentation de forme, on modifie la matrice de paramètres, $W_{ij}^{t+1} = W_{ij}^{t} + \Delta W ij^{t}$

De manière à améliorer l'apprentissage de la fonction "Bio" nous avons été amenés à centrer et réduire par variables les quantités manipulées. Les différentes variables sont alors compatibles, et interviennent d'une manière équivalente, et permettent de lutter contre les problèmes numériques durs aux ordres de grandeurs très différents à prendre en compte.

4.5.5 Apprentissage

On vérifie à l'aide de l'ensemble de validation la qualité de l'apprentissage. Comme nous n'avons jamais obtenu de sur-apprentissage, et nous avons arrêté l'apprentissage au bout de 50.000 itérations.

On observe sur la figure (4.3) la courbe de validation du réseau de neurones traitant les faibles profondeurs NN_1. ainsi que sur la figure (4.4), on observe la courbe de validation du réseau de neurones traitant les grandes profondeurs

Pour analyser les performances obtenus pour les deux réseaux NN_1 et NN_2, nous avons réalisé plusieurs expériences que nous présentons comme ceci.

4.6 Visualisation des performances des réseaux de neurones

4.6.1 Les profils

Une manière simple de visualiser les performances des réseaux de neurones est de tracer les profils , c-à-d dessiner les sorties du réseaux par rapport aux

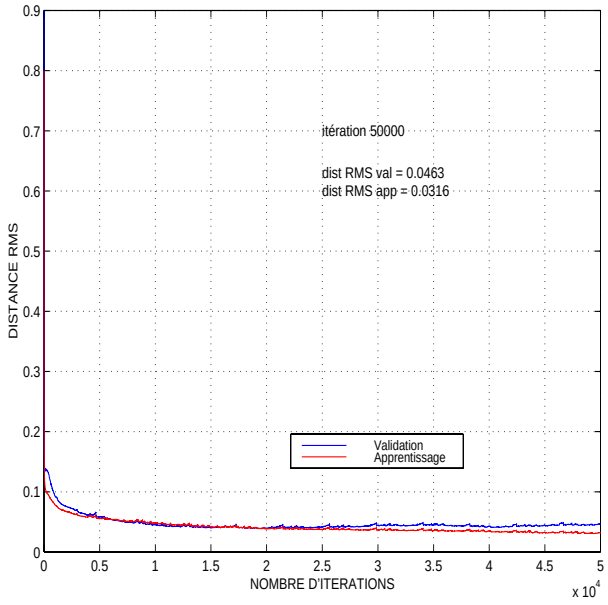

FIG. 4.3 –: *courbe de validation croisée lors d'un apprentissage du réseau de neurones pour les faibles profondeurs (NN1).*

différentes profondeurs et comparer la courbe faite avec la biologie calculée à partir du modèle équationnel. Cette même expérience peut être réalisée pour vérifier la validité des fonctions obtenues, en fonction d'autres paramètres.

La figure (4.5) concerne le premier réseau qui traite les faibles profondeurs NN_1, il représente 6 profils séparés par une période de 50 jours, et chacun de ces 6 profils admet deux courbes, une pour le réseau neuronal et l'autre pour la simulation.

La figure (4.6) concerne le deuxième réseau qui traite les grandes profondeurs NN_2, représente 6 profils séparés par une période de 50 jours, et chacun de ces 6 profils admet deux courbes, une pour le réseau neuronal et l'autre pour la simulation. On remarque que pour les deux réseaux de neurones NN_1 et NN_2, les ordres de grandeurs et les tendances sont bien reproduits et ceci quelque soit la période de l'année considérée.

4.6.2 Test de la biologie calculée avec celle simulée

Une autre manière de visualiser les performances du réseau de neurone, est de tracer un graphe de diffusion qui fait correspondre à chaque biologie calculée par le réseau de neurone, la quantité de biologie simulée, et on peut facilement faire un diagnostic en comparant le graphe par rapport à la diagonale. Quant à la figure (4.7) représente la correspondance entre la biologie calculée, à partir du réseau de neurones pour les faibles profondeurs et la biologie calculée à partir du système équationnel.

Alors que la figure (4.8) représente la correspondance entre la biologie calculée, à partir du réseau de neurones pour les grandes profondeurs et la biologie

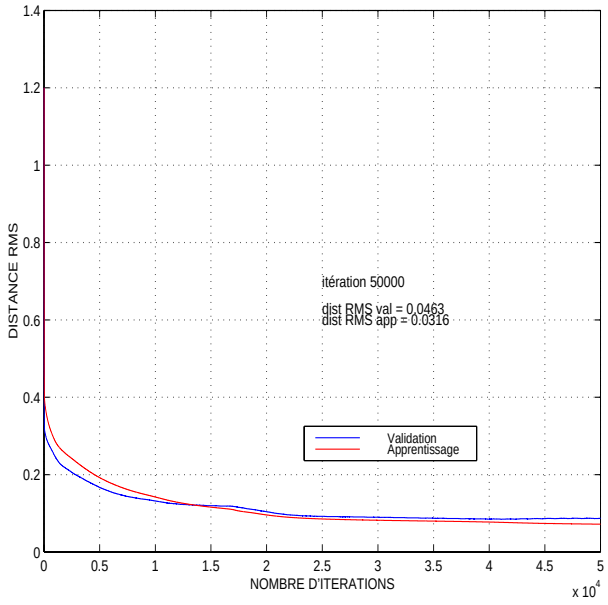

FIG. 4.4 –: *courbe de validation croisée lors d'un apprentissage du réseau de neurones pour les grandes profondeurs(NN2).*

calculée à partir du système équationnel.

De manière claire, les deux réseaux NN_1 et NN_2 représentent un comportement cohérent.

4.7 Conclusion

Ce chapitre a présenté les premiers résultats obtenus par les deux réseaux NN_1 et NN_2 mis au point à l'aide des équations biologiques représentant le phénomène que nous étudions. Les performances obtenues sont satisfaisantes, et les équations utilisées sont extrêmement non-linéaire, elles présentent donc de nombreux problèmes numériques dus aux ordres de grandeurs que nous manipulons. De nombreuses modifications peuvent être apportées pour améliorer l'approximation de la fonction "Bio", par exemple l'amélioration de l'ensemble d'apprentissage, de l'architecture, maîtrise des effets de bords...

Le but de ce mémoire étant de montrer le fonctionnement du système complet que nous avons utilisé, les réseaux NN_1 et NN_2 que nous venons de présenter et dont les performances sont suffisantes pour montrer la validité de l'assimilation des données neuronales.

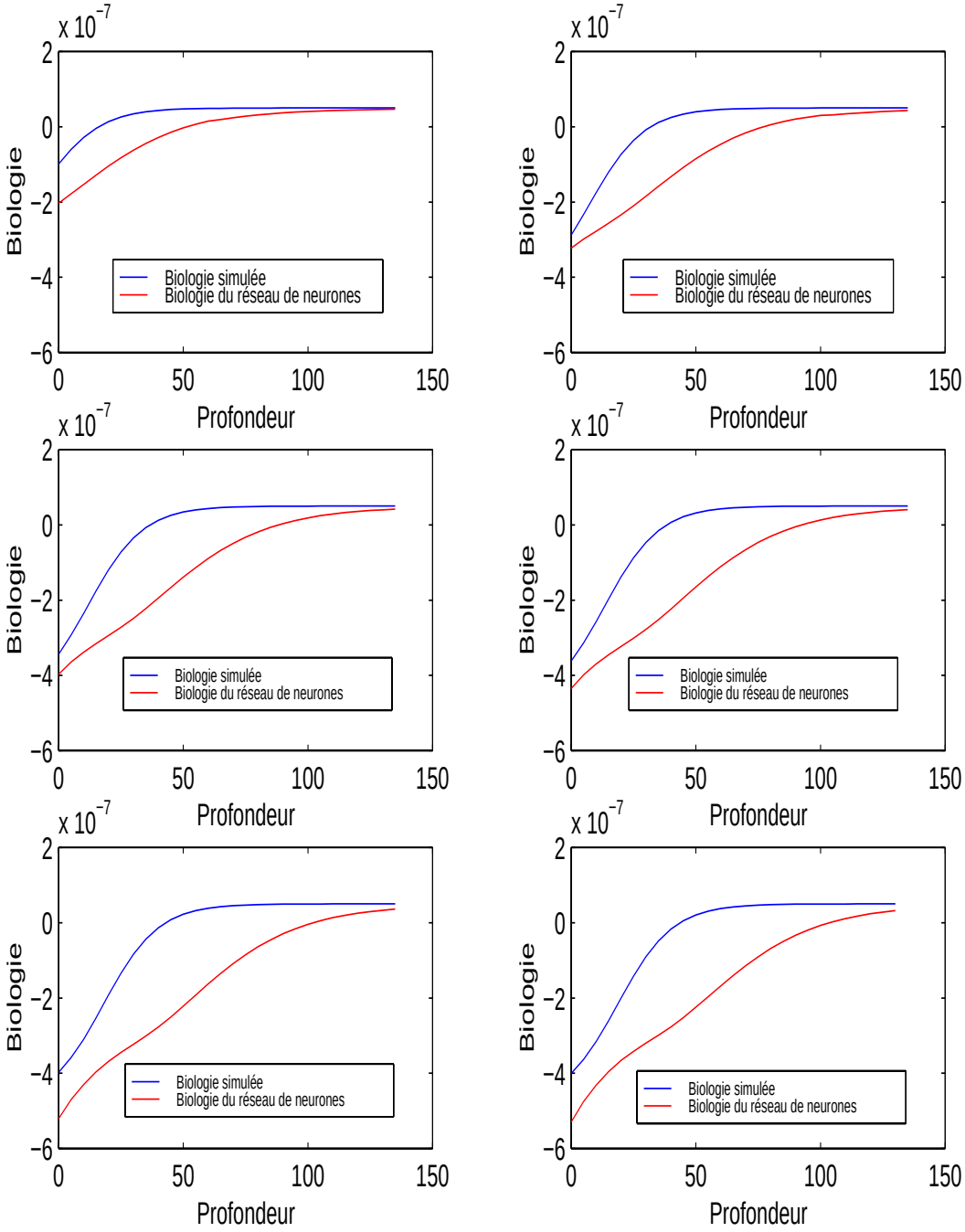

FIG. 4.5 –: *6 profils du réseau de neurones traitant les faibles profondeurs (NN2) chaque 50 jours .*

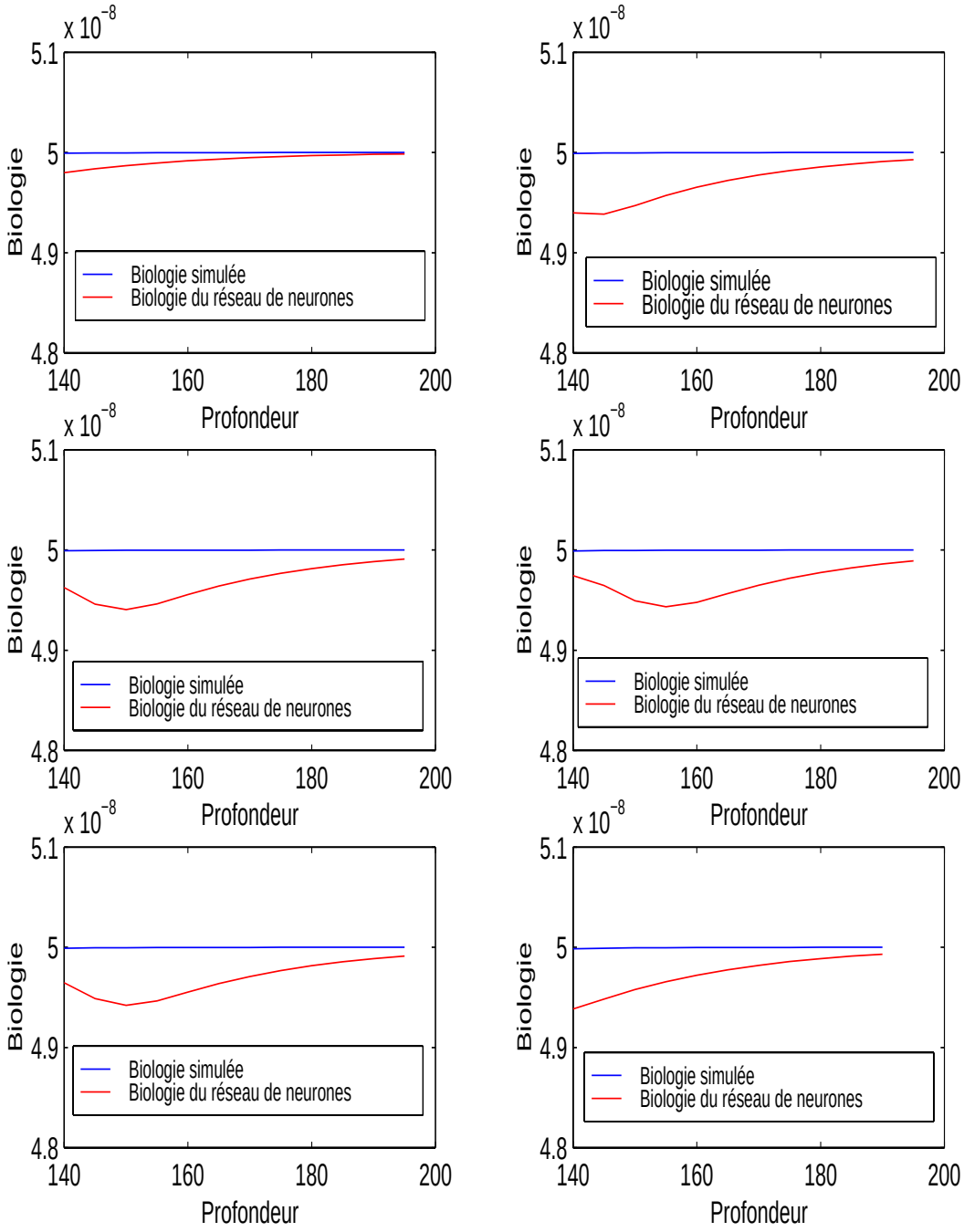

FIG. 4.6 –: *6 profils du réseau de neurones traitant les grandes profondeurs (NN2) chaque 50 jours .*

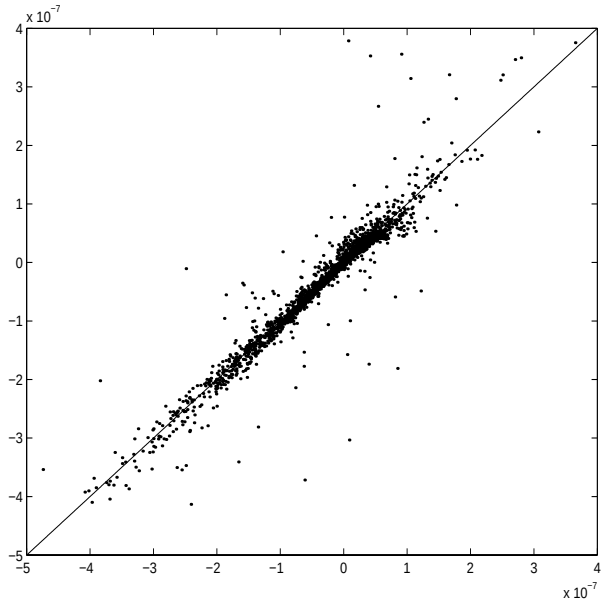

FIG. 4.7 –: *Comparaison entre la biologie calculée à partir du réseau de neurones des faibles profondeurs (NN1) et celle calculée à partir du système équationnel .*

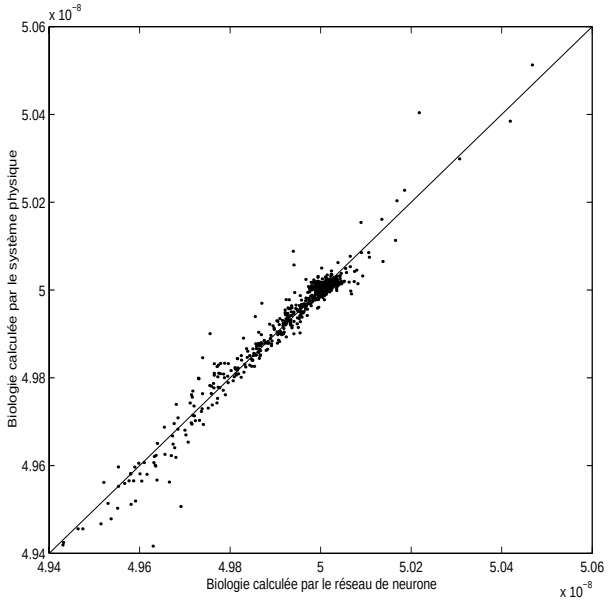

FIG. 4.8 –: *Comparaison entre la biologie calculée à partir du réseau de neurones des grandes profondeurs (NN2) et celle calculée à partir du système équationnel .*

Chapitre 5

Système Modulaire

5.1 Modélisation

De nombreux systèmes complexes d'apprentissage s'expriment aisément sous la forme de la coopération de modules plus simples. Ces modèles communiquent entre eux selon une structure qui peut être décrite par un graphe sans circuit ayant un module d'entrée et un module de sortie. Les variables d'entrée du module d'entrée constituent alors les variables d'entrée du système complexe et les variables de sortie du module de sortie constituent les variables de sortie de ce système complexe. Les différents modules constituant appartiennent à des modèles qui peuvent être de nature différentes; mais nous supposons que chaque module dépend d'un certain nombre de paramètres.

Ainsi, on note par W_i les paramètres du module numéro i et par W l'ensemble de tous les paramètres du système. L'apprentissage consiste à minimiser une fonction coût $\mathcal{J}$ qui s'exprime en fonction des sorties y du système, et que nous pouvons exprimer aussi par $\mathcal{J}(x, W)$ où x représente les variables d'entrée du système et W l'ensemble de ses paramètres. La structure modulaire du système complexe, décrite plus haut, amène à concevoir $\mathcal{J}(x.W)$ comme une composition de fonction.

Des recherches ont été menées afin de proposer une méthode unifiée pour la coopération de plusieurs algorithmes d'apprentissage, ces méthodes présentent de multiples intérêts :

- Elles permettent d'aborder l'aspect structurel des systèmes d'apprentissage;
- Elles permettent, en pratique, de concevoir des systèmes *modulaires* , ce qui peut avoir un impact considérable sur le développement et l'implémentation de systèmes d'apprentissage complexes.

Notre système modulaire peut être donc décrit comme un graphe sans circuit, dont les noeuds représentent des modules F_n, et les arcs représentent les connexions entre ces modules.

Les entrées de chaque module sont reliées à des sorties appartenant aux modules prédécesseurs sur le graphe (Fig 5.1). Chaque module est défini par une

relation dérivable par rapport aux entrées et aux paramètres, il peut être défini par exemple, par une fonction, par un réseau multicouches, par une fonction radiale de base et d'autres types de modèle.

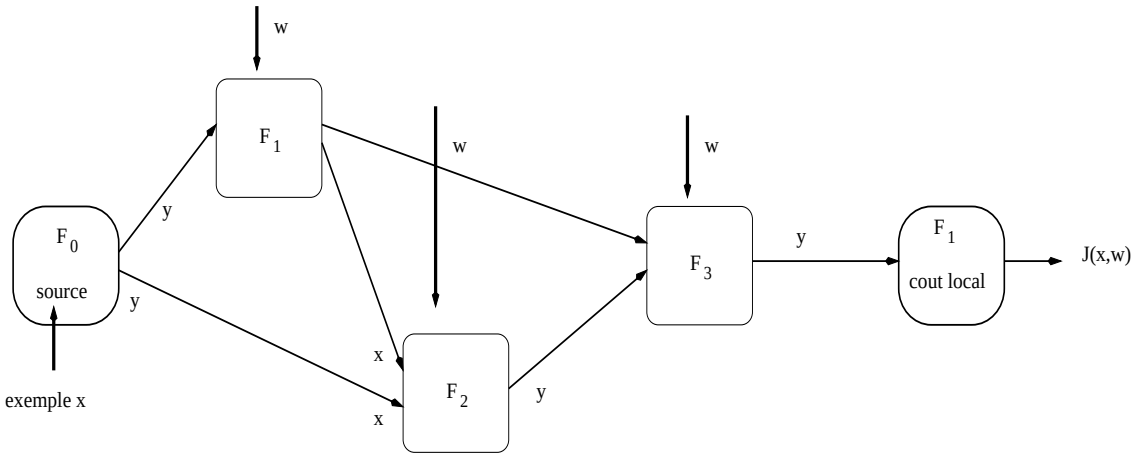

FIG. 5.1 –: *Un système modulaire*

Pour simplifier les calculs, nous considérons que le premier module (module d'entrée du système complexe) F_0 est la fonction identique, sa fonction consiste à alimenter le système global avec de nouvelles données x. De plus, le dernier module (module de sortie du système complexe) calcule simplement le coût local $\mathcal{J}(x, w)$.

La description mathématique des connexions entre modules requiert une certaine rigueur dans la façon d'indicer les différentes grandeurs considérées.

Le graphe décrivant l'interconnexion des différents modules étant sans circuit, il est alors possible de numéroter les modules suivant l'ordre topologique. Ainsi, relativement à cet ordre, l'existence d'un arc de F_p à F_q ($F_p \longrightarrow F_q$) implique nécessairement que $p < q$.

On note par N_{E_n} (resp N_{S_n}) le nombre de variables d'entrée (resp. sortie) du module F_n, la numérotation des variables d'entrée et de sortie des différents modules se fait de la manière suivante :

On commence par attribuer les numéros de 1 à NS_0 aux variables de sortie du module F_0, puis les numéros de $NS_0 + 1$ à $NS_0 + NS_1$ aux variables de sortie du module F_1 et ainsi de suite pour F_2,F_3,.... Le même procédé de numérotation s'applique aux variables d'entrée (sauf qu'il faut commencer par F_1 puisque $NE_0 = 0$) et il s'applique aussi pour la numérotation des paramètres.

On note par :

1. E l'ensemble des indices de toutes les variables d'entrée,
2. S l'ensemble des indices de toutes les variables de sortie précisant que le

dernier module ne comporte pas de sortie du fait qu'il calcul seulement la fonction coût,

3. W l'ensemble des indices de touts les paramètres,

4. M l'ensemble des indices des différents modules,

On définit alors deux applications notées X et Y :
$X : E \longrightarrow M$ telle que $\forall i \in E$, $X(i)$ désigne l'indice du module pour lequel i est l'indice d'une de ses variables d'entrée,

et $Y : S \longrightarrow M$ telle que $\forall j \in S$, $Y(j)$ désigne l'indice du module pour lequel j est l'indice d'une de ses variables de sortie.

Un module est défini par :

$$\forall j \in Y^{-1}(n),\ y_j = f_j((x_k)_{k \in X^{-1}(n)}, (w_i)_{i \in W^{-1}(n)}) \tag{5.1}$$

D'autre part, étant donné qu'une entrée d'un module provient d'une et d'une seule sortie d'un module prédécesseur, on peut alors exprimer la topologie de ses connexions entre les variables par une fonction ϕ, appliquer ϕ à l'indice d'une entrée donne l'indice de la sortie reliée à cette entrée.

$$\forall k\ x_k = y_{\phi(k)} \tag{5.2}$$

on a:

$$\forall k \in X^{-1}(n) \implies \phi(k) \in \cup_{p=0}^{n-1} Y^{-1}(p) \tag{5.3}$$

Dans la suite, notre objectif consiste à calculer les dérivées du coût local $\mathcal{J}(x, W)$. Or, il est possible de calculer simplement ces dérivées, à l'aide seulement des dérivées des fonctions f_l.

5.2 Apprentissage de système modulaire

Un formalisme Lagrangien permet alors de calculer aisément le gradient du coût local $\mathcal{J}(X, W)$ de façon peu dépendante de la forme des modules. Ce gradient peut être calculé efficacement au cours d'une simple passe arrière,[Bottou,1991].

Le lien entre ce calcul du gradient et l'algorithme de rétro-propagation du gradient, est également souligné.

5.2.1 Calcul des dérivées du coût local

Afin de calculer facilement les dérivés de $\mathcal{J}$ par rapport à tous les paramètres d'un système modulaire, il est utile d'introduire un formalisme Lagrangien.

Formalisme Lagrangien.

Afin de mieux présenter l'intérêt de ce formalisme, nous présentons d'abord un cas trivial (comme sur la figure (5.5)) avant de présenter l'étude générale.

Considérons deux fonctions, $f(x)$ de $\mathcal{R}^l$ vers $\mathcal{R}^M$ et $g(y)$ de $\mathcal{R}^M$ vers $\mathcal{R}$. Supposons maintenant, qu'on a un système modulaire composé de deux modules f et g,

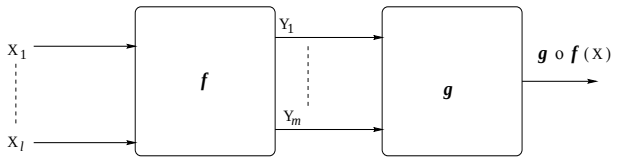

FIG. 5.2 –: *Système complexe composé de deux modules.*

On calcule d'habitude les dérivées de la composée $g \circ f$ à l'aide de la règle de la dérivation en chaîne. Une autre façon consiste à écrire le Lagrangien de g, avec la contrainte $y = f(x)$:

$$L(x, y, \beta) = g(y) - \sum_m \beta_m (y_m - f_m(x)) \tag{5.4}$$

Un calcul simple montre que choisir β en résolvant les équations

$$\forall m\,;\ \left.\frac{\partial L}{\partial y_m}\right|_{x,f(x),\beta(x)} = 0$$

implique

$$\left.\frac{\partial L}{\partial x_i}\right|_{x,f(x),\beta(x)} = \left.\frac{\partial g \circ f}{\partial x_i}\right|_x \tag{5.5}$$

En effet :

$$\begin{array}{rcl} \left.\frac{\partial L}{\partial y_m}\right|_{x,f(x),\beta(x)} & = & \frac{\partial g(y)}{\partial y_m} - \beta_m(x) = 0 \\ & = & \left.\frac{\partial g}{\partial y_m}\right|_{f(x)} - \beta_m(x) = 0 \end{array} \tag{5.6}$$

$$\text{d'où } \beta_m(x) = \left.\frac{\partial g}{\partial y_m}\right|_{f(x)} \tag{5.7}$$

Donc d'après l'équation (5.7) en remplaçant β_m par $\left.\frac{\partial g}{\partial y_m}\right|_{f(x)}$, on aura :

$$\left.\frac{\partial L}{\partial x_i}\right|_{x,f(x),beta(x)} = \sum_m \left.\frac{\partial g}{\partial y_m}\right|_{f(x)} \cdot \frac{\partial f_m}{\partial x_i} = \left.\frac{\partial g \circ f}{\partial x_i}\right|_x \tag{5.8}$$

Ceci démontre l'équation (5.5).

Nous venons de voir cette idée appliquée sur un cas trivial, nous allons maintenant étendre ce formalisme pour un système modulaire complexe.

Cas général.

En reprenant le système modulaire complexe, nous présenterons le formalisme lagrangien afin de définir une méthodologie permettant un calcul simple des dérivées.

Les équations (5.1) et (5.2) sont les contraintes de notre système. On peut alors écrire le Lagrangien comme suit :

$$L = \mathcal{J} - \sum_{k \in E} \beta_k \left(x_k - y_{\phi(k)}\right) - \sum_{j \in S} \alpha_j (y_j - f_j(\ (x_k)_{k \in X^{-1}(Y(j))},\ (w_i)_{i \in W^{-1}(Y(j))})) \tag{5.9}$$

Remarque. *La somme du second terme de la partie droite de l'équation précédente concerne tous les modules excepté le premier (celui qui génère notre système par les entrées), et la somme du troisième terme concerne tous les modules excepté le dernier (du fait qu'il calcule seulement la fonction coût).*

On détermine les quantités β et α en résolvant les équations :

$$\frac{\partial L}{\partial y_j} = \quad 0 = \quad -\alpha_j + \sum_{k \in \phi^{-1}(j)} \beta_k \tag{5.10}$$

$$\begin{aligned} \frac{\partial L}{\partial x_k} = \quad 0 = \quad & -\beta_k + \frac{\partial \mathcal{J}}{\partial x_k} \qquad si\ l'entrée\ k\ appartient\ au\ dernier\ module \\ = \quad & -\beta_k + \sum_{j \in Y^{-1}(X(k))} \alpha_j \frac{\partial f_j}{\partial x_k} \quad sinon \end{aligned} \tag{5.11}$$

D'après l'équation (5.11) qui est explicitée par la figure 5.3, on voit bien qu'on joint à chaque variable x_k une variable conjuguée β_k , même remarque est faite pour l'équation (5.10) qui est explicitée par la figure 5.4, par contre on joint à chaque variable y_j une variable conjuguée α_j.

A partir des équations (5.10) et (5.11), on peut décrire une méthode itérative permettant de calculer tous les α_j et β_k. La première partie de l'équation (5.11) permet d'initialiser les $\beta_k = \frac{\partial \mathcal{J}}{\partial x_k}$ pour les entrées du dernier module. L'équation (5.10) permet de calculer α_j connaissant tous les β_k pour $k \in \phi^{-1}(j)$.

L'équation (5.11) permet de calculer les β_k connaissant tous les α_j pour $j \in Y^{-1}(X(k))$. Ce dernier calcul est possible car on suppose que les $\frac{\partial f_j}{\partial x_k}$ sont

connus. Ces 2 formules permettent de calculer alternativement les quantités α_j et β_k par rétro-propagation.

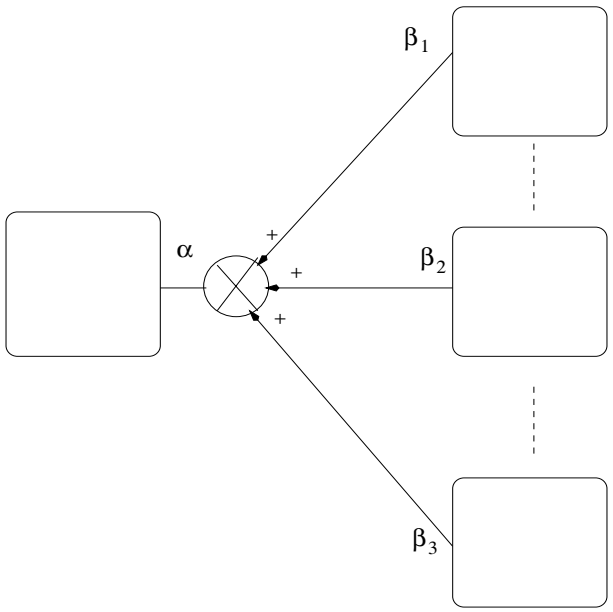

FIG. 5.3 –: *La rétro-propagation entre modules*

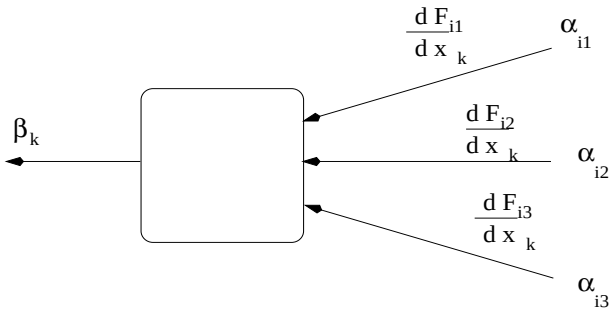

FIG. 5.4 –: *La rétro-propagation à l'interieur des modules*

$$\forall j \in S \quad \alpha_j = \frac{\partial \mathcal{J}}{\partial y_j} \tag{5.12}$$

, et

$$\forall k \in E \quad \beta_k = \frac{\partial \mathcal{J}}{\partial x_k} \tag{5.13}$$

Démonstration. Pour démontrer que $\beta_k = \frac{\partial \mathcal{J}}{\partial x_k}$ et que $\alpha_j = \frac{\partial \mathcal{J}}{\partial y_j}$, on doit d'abord démontrer la vérité des propositions suivantes :

1. Si $\forall k \in \phi^{-1}(j) \quad \beta_k = \frac{\partial \mathcal{J}}{\partial x_k}$ alors $\alpha_j = \frac{\partial \mathcal{J}}{\partial y_j}$.

2. Si $\forall j \in Y^{-1}(X(k)) \quad \alpha_j = \frac{\partial \mathcal{J}}{\partial y_j}$ alors $\beta_k = \frac{\partial \mathcal{J}}{\partial x_k}$

Pour démontrer la proposition 1, on commence par supposer que :

$$\forall k \in \phi^{-1}(j) \quad b_k = \frac{\partial \mathcal{J}}{\partial x_k}$$

et il faut alors démontrer que :

$$\alpha_j = \frac{\partial \mathcal{J}}{\partial y_j}$$

. en effet :

$$\frac{\partial \mathcal{J}}{\partial y_j} = \sum_{k \in \phi^{-1}(j)} \frac{\partial x_k}{\partial y_j} \cdot \frac{\partial \mathcal{J}}{\partial x_k} \tag{5.14}$$

mais d'après l'hypothèse 1) on a $\frac{\partial \mathcal{J}}{\partial x_k} = \beta_k$ et d'après la définition de la fonction ϕ, on a :

$$\forall k \in \phi^{-1}(j),\ x_k = y_j \Longrightarrow \frac{\partial x_k}{\partial y_j} = 1$$

donc

$$(5.14) \Longrightarrow \frac{\partial \mathcal{J}}{\partial y_j} = \sum_{x \in \phi^{-1}(j)} \beta_k = \alpha_j \quad \textbf{C.Q.F.D.}$$

Pour démontrer la proposition 2, on commence par supposer que :

$$\forall j \in Y^{-1}(X(k)) \quad \alpha_j = \frac{\partial \mathcal{J}}{\partial y_j}$$

et il faut alors démontrer que :

$$\beta_k = \frac{\partial \mathcal{J}}{\partial x_k}$$

en effet :

$$\frac{\partial \mathcal{J}}{\partial x_k} = \sum_{j \in Y^{-1}(X(k))} \frac{\partial y_j}{\partial x_k} \cdot \frac{\partial \mathcal{J}}{\partial y_j} \tag{5.15}$$

Mais par l'hypothèse 2), on a $\frac{\partial \mathcal{J}}{\partial y_j} = \alpha_j$ et d'après les contraintes, on a $y_j = f_j$.

donc

$$(5.15) \Longrightarrow \frac{\partial \mathcal{J}}{\partial x_k} = \sum_{j \in Y^{-1}(X(k))} \alpha_j \cdot \frac{\partial f_j}{\partial x_k} \quad \textbf{C.Q.F.D.}$$

Ainsi tous les multiplicateurs de Lagrange peuvent être calculés alternativement au cours d'une unique récurrence arrière, ceci peut être facilement vu par la figure (5.5), en initialisant le vecteur β_4 par $\frac{\partial \mathcal{J}}{\partial x_k}$ où x_k est l'ensemble des entrées du dernier module. Ensuite, on peut par l'équation (5.10) calculer le

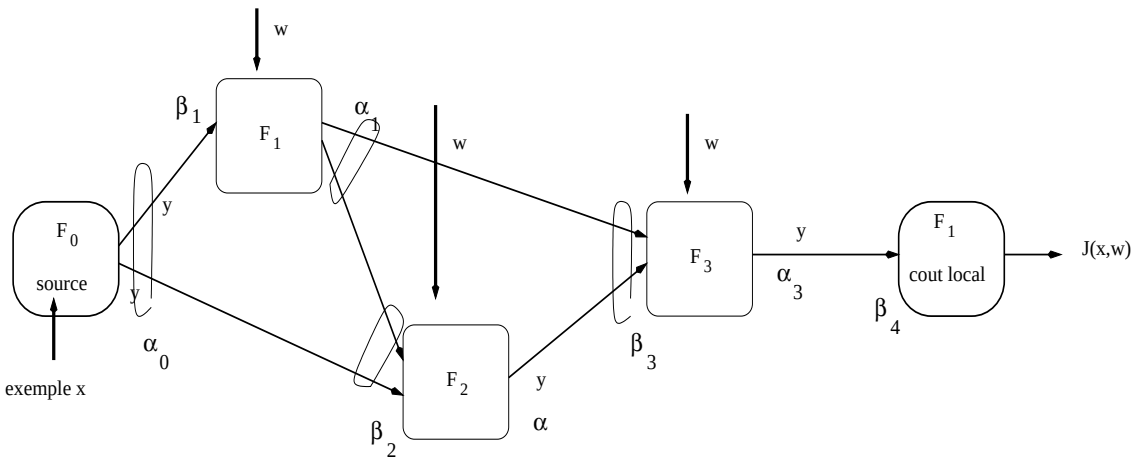

FIG. 5.5 –: *Le chaînage entre les variables conjuguées.*

vecteur α_3 connaissant tous les β_k tel que $k \in \phi^{-1}$, ainsi de suite, il est clair qu'en une seule passe, on peut calculer toutes les variables conjuguées.

Aprés avoir calculé tous les α_j et β_k, il est alors facile d'obtenir les dérivées Δ_i de $\mathcal{J}$ par rapport au paramètre w_i, en effet :

$$\Delta_i = \frac{\partial \mathcal{J}}{\partial w_i} = \sum_{j \in Y^{-1}(W(i))} \frac{\partial y_j}{\partial w_i} \cdot \frac{\partial \mathcal{J}}{\partial y_j} \tag{5.16}$$

car w_i intervient dans $\mathcal{J}$ par l'intermédiaire de y_j telque $j \in Y^{-1}(W(i))$ ce qui peut être vu par la figure 5.6, mais d'autre part on a:

$$y_j = f_j \Longrightarrow \quad \frac{\partial y_j}{\partial w_i} = \frac{\partial f_j((x_k)_{k \in X^{-1}(n)})}{\partial w_i} \tag{5.17}$$

De ce qui précède et de l'équation (5.12), on déduit que :

$$\Delta_i = \frac{\partial \mathcal{J}}{\partial w_i} = \sum_{j \in Y^{-1} W(i)} \alpha_j \cdot \frac{\partial f_i((x_k)_{k \in X^{-1}(Y(j))}}{\partial w_i} \tag{5.18}$$

En résumé,

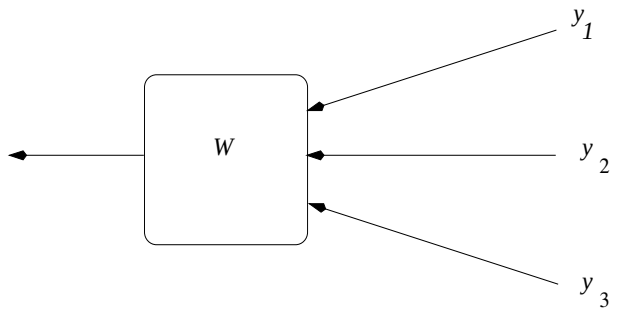

FIG. 5.6 –: *Le lien entre les paramètres W et les variables de sortie y*

$$\begin{aligned}
\beta_k &= \frac{\partial \mathcal{J}}{\partial x_k} \quad \text{si l'entrée } k \text{ appartient au dernier module.} \\
&= \sum_{j \in Y^{-1}X(k)} \alpha_j \cdot \frac{\partial f_j}{\partial x_k} \quad \text{dans les autres cas.} \\
\alpha_j &= \sum_{k \in \phi^{-1}(j)} \beta_k. \\
\Delta_i &= \frac{\partial \mathcal{J}}{\partial w_i} = \sum_{j \in Y^{-1}W(i)} \alpha_j \cdot \frac{\partial f_j}{\partial w_i} \qquad (5.19)
\end{aligned}$$

Remarque 1. *Si on s'intéresse à la dérivée du coût $\mathcal{J}$ par rapport aux entrées du système modulaire complexe, $\frac{\partial \mathcal{J}}{\partial x_k}$, alors dans ce cas on s'intéressera au calcul de $\alpha_0, \alpha_1, \ldots, \alpha_{N_{S_0}}$, car le premier module est l'application identique, donc $\alpha_i = \beta_i \quad pour \quad i = 1, \cdots, N_{S_0}$.*

Remarque 2. *On peut voir le réseau multicouche comme un système modulaire complexe, si chaque module est représenté par un neurone (somme pondérée de ces entrées suivie de l'application de sa fonction de transfert). En réécrivant les formules précédentes, on retrouve les formules classiques de l'algorithme de la rétro-propagation.*

L'apprentissage consiste à minimiser $\mathcal{J}$ par rapport à W qui peut se faire par une méthode de gradient, ce qui nécessite le calcul de Δ_i,[Bottou et Gallinari,1992],[Jordan et Rumelhart,1995].

Les formules précédentes décrivent de façon générale les échanges d'information entre modules nécessaires pour effectuer l'apprentissage global de notre système. Il suffit donc de savoir effectuer trois opérations sur chaque module:

– Calculer ses sorties y_i, connaissant ses entrées x_k et ses paramètres w_i.

– Calculer les conjugés β_k de ses entrées, connaissant les conjugés α_j de ses sorties, ses entrées x_k et ses paramètres w_i.

– Calculer les dérivées de ces paramètres Δ_i, connaissant les conjugués α_j de ses sorties, ses entrées x_k et ses paramètres w_i.

Stratégie d'apprentissage

Concernant l'apprentissage, il peut se faire selon des stratégies différentes :

1. On peut faire l'apprentissage de tous les modules globalement, c-à-d qu'on modifie tous les poids des modules en calcul tous les Δ_i,
2. On peut apprendre quelques modules en figeant les autres, ce qu'on appelle "apprentissage partiel", puis alterner en faisant un apprentissage partiel sur d'autres modules et ainsi de suite.
3. On peut combiner les deux stratégies précédentes.

Remarque. *Pour la suite, on suppose que les paramètres des modules sont figés, et on souhaite calculer* $\frac{\partial \mathcal{J}}{\partial x_i}$ *pour* $i = 1, \ldots, NS_0$ *où* x *représente les entrées du système complexe ce qui équivaut à calculer* $\alpha_j = \frac{\partial \mathcal{J}}{\partial y_j}$ *pour* $j = 1, \ldots, NS_0$ *du fait que le premier module est une fonction identique.*

Chapitre 6

Assimilation de données de productivité marine

6.1 Introduction

Nous avons présenté aux chapitres précédents : les aspects physiques du problème de la productivité marine, la problématique de l'assimilation de données variationnelles, les réseaux de neurones et les systèmes modulaires. Nous présentons dans ce chapitre une méthodologie d'assimilation de données variationnelles basée sur les réseaux de neurones et les systèmes modulaires que nous appliquerons au modèle biologique de la productivité marine.

La figure (2.3) illustre le modèle de couplage de la biologie avec la dynamique. Dans le cadre d'une évolution temporelle de T unités de temps, le graphe décrit par la figure (2.3) sera répété T fois. Le graphe (2.4) représente la dynamique pour $T = 2$. L'évolution temporelle du modèle permet de prédire les concentrations en nitrates, en fonction de la profondeur et du temps, il dépendde deux paramètres K_E et α pour lesquels il faut avoir de "bonnes" estimations.

Ainsi, dans le graphe généré, les blocs représentent des fonctions et les arcs leurs échanges de données entre ces fonctions, il s'agit donc d'un système modulaire. D'autre part, on dispose d'observations de concentrations des nitrates à des instants et des profondeurs données. Les concentrations des nitrates prédites par le modèle aux points d'observations peuvent être très différentes des concentrations observées. Il s'agit donc d'adapter les paramètres α et K_E du modèle de façon à minimiser l'écart entre les observations prédites et les observations observées. Ainsi si on note par $\mathcal{J}$ l'erreur quadratique entre les concentrations prédites et observées, minimiser $\mathcal{J}$ par une méthode de gradient, nécessite le calcul de $\frac{\partial \mathcal{J}}{\partial \alpha}$ et $\frac{\partial \mathcal{J}}{\partial K_E}$. Au chapitre 5, nous avons présenté les formules permettant de faire ce calcul dans le cadre d'un système modulaire.

L'application de ces formules, nécessite au niveau de chaque bloc, la connaissance de la dérivée de sa sortie par rapport à ses entrées. Ces dérivées sont simples dans le cas des blocs du type "N" et "N^*" de la figure (2.3), car il s'agit dans ce cas de fonctions linéaires. Mais par contre, les blocs "BIO" représentent

des fonctions non linéaires définies par des équations différentielles.

Afin de disposer des dérivées pour ces blocs, nous avons été amené à les modéliser par les réseaux multicouches que nous avons présenté au chapitre 4. En effet, la modélisation d'une fonction non linéaire par réseau de neurone multicouche permet de calculer par rétro-propagation et d'une manière simple sa dérivée par rapport à ses variables d'entrées.

6.2 Application

L'application que nous avons traité dans le cadre de ce stage a été présenté au chapitre 2, comme nous l'avons signalé, il s'agit d'un modèle simplifié. Dans le cadre de ce stage, nous nous sommes restreints à 2 pas de temps. Nous avons appliqué le modèle décrit au chapitre 2 à partir du jeu de deux paramètres α et K_E et nous avons obtenu des concentrations en nitrates à $T = 2$. Nous avons considéré que ces concentrations calculées par le modèle sont des mesures observées, puis nous avons perturbé les valeurs des deux paramètres (α et K_E) et nous avons appliqué la méthode d'assimilation de données par système modulaire neuronal pour calculer le gradient et adapter les paramètres α et K_E. Le but de cette manipulation est de vérifier que l'algorithme permet de retrouver les valeurs exactes des paramètres.

Nous avons généré 4 types de données en prenant les jeux de paramètres :

1. $\alpha = 10 \times 10^{-7}$, $K_E = 5$
2. $\alpha = 5 \times 10^{-7}$, $K_E = 10$
3. $\alpha = 10 \times 10^{-7}$, $K_E = 10$
4. $\alpha = 5 \times 10^{-7}$, $K_E = 5$

Le choix des paramètres peut être motivé par les considérations suivantes : Les paramètres ont été choisi de la manière suivante :

- $\alpha = 5 \times 10^{-7}$ correspond à la valeur moyenne de ce paramètre,
- $\alpha = 10 \times 10^{-7}$ correspond à une valeur extrême puisque l'écart type de α est de 3.75×10^{-7},
- $K_E = 5$ correspond à la valeur moyenne du paramètre K_E,
- $K_E = 10$ correspond à une valeur extrême puisque l'écart type de α est de 3.75,

Afin de bien visualiser les résultats, nous avons présenté pour chaque jeu de données les 3 courbes montrant l'évolution en fonction des itérations de l'erreur quadratique et des 2 paramètres α et K_E estimés.

La figure (6.1) illustre l'évolution des paramètres pour le premier type de données, on voit bien que α converge vers la valeur de 10×10^{-7}, de même pour

le paramètre K_E qui tend vers la valeur 5, et en parallèle on voit la minimisation de l'erreur quadratique, dans ce cas la valeur initiale de α a été prise égale à 5×10^{-7}, et la valeur initiale de K_E à 10.

La figure (6.2) illustre l'évolution des paramètres pour le deuxième type de données, on voit bien que α converge vers la valeur de 5×10^{-7}, de même pour le paramètre K_E qui tend vers la valeur 10, et en parallèle on voit la minimisation de l'erreur quadratique, dans ce cas la valeur initiale de α a été prise égale à 0, et la valeur initiale de K_E à 15.

La figure (6.3) illustre l'évolution des paramètres pour le troisième type de données, on voit bien que α converge vers la valeur de 10×10^{-7}, de même pour le paramètre K_E qui tend vers la valeur 10, et en parallèle on voit la minimisation de l'erreur quadratique, dans ce cas la valeur initiale de α a été prise égale à 20×10^{-7}, et la valeur initiale de K_E à 1.

La figure (6.4) illustre l'évolution des paramètres pour le quatrième type de données, on voit bien que α converge vers la valeur de 5×10^{-7}, de même pour le paramètre K_E qui tend vers la valeur 5, et en parallèle on voit la minimisation de l'erreur quadratique, dans ce cas la valeur initiale de α a été prise égale à 0, et la valeur initiale de K_E à 15.

6.3 Logiciel, méthodologie d'implémentation

L'application a été implémenté sous le langage C++ qui nous a permis de profiter d'avantage de la programmation orientée objet et de sa gestion des ressources. Le code de cette méthodologie est partagé suivant des modules et des classes, la taille de ce code est de 2500 lignes. Nous détaillerons quelques structures de données dans l'annexe (1).

Nous avons considéré la figure (2.3) comme étant un graphe où chaque boîte correspond à un noeud et les arcs correspondent à la relation qui permet aux boîtes de communiquer entre elles.

Ainsi, dans l'implémentation nous avons pris en compte la structure du graphe. Chaque noeud étant un objet, on lui affecte les fonctions spécifiques, cette implémentation nous permet d'avoir un programme générique.

D'après cette démarche, on a représenté en C++ chaque noeud par un objet qui a deux fonctions, une pour procéder à la propagation avant, et l'autre pour la rétro-propagation, mais vu que les noeuds ont des fonctionnalités différentes, on a profité de l'héritage pour adapter ces objets à nos fonctions du modèle.

Si on itère la modélisation plusieurs fois, un graphe sera généré en mémoire, dans le cadre du problème d'assimilation de données, nous générons à chaque fois la partie du graphe correspondant à une étape de temps, ceci nous permettra de faire des simulations à n'importe quel pas de temps, et n'importe quel nombre de données observées.

Notre méthodologie d'implémentation consiste à décrire le graphe, c'est à dire les noeuds et les relations entre ces dernièrs, ensuite par la notion d'héritage à spécifier et à modifier chaque fonction selon nos besoins, et enfin, à décrire les fonctions des propagations et de rétro-propagations. Les données observées sont décrites dans un fichier que le programme peut consulter.

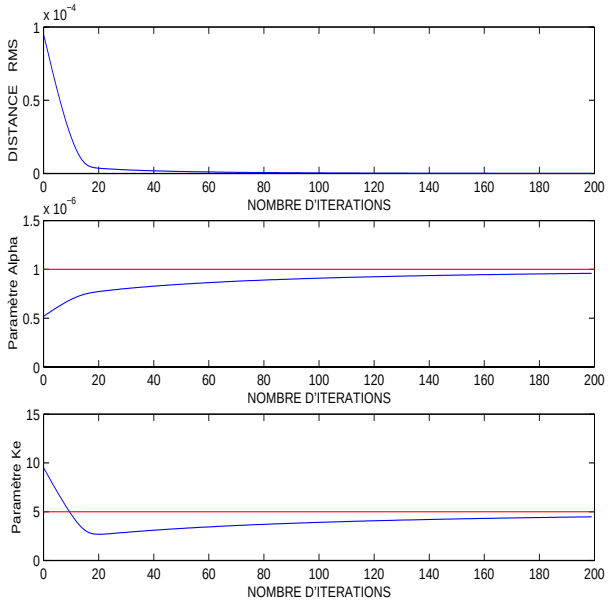

FIG. 6.1 –: *L'affichage de la distance RMS, la convergence des deux paramètres pour le premier jeu de données avec les initialisations* $\alpha = 5 \times 10^{-7}$ *et* $K_E = 10$.

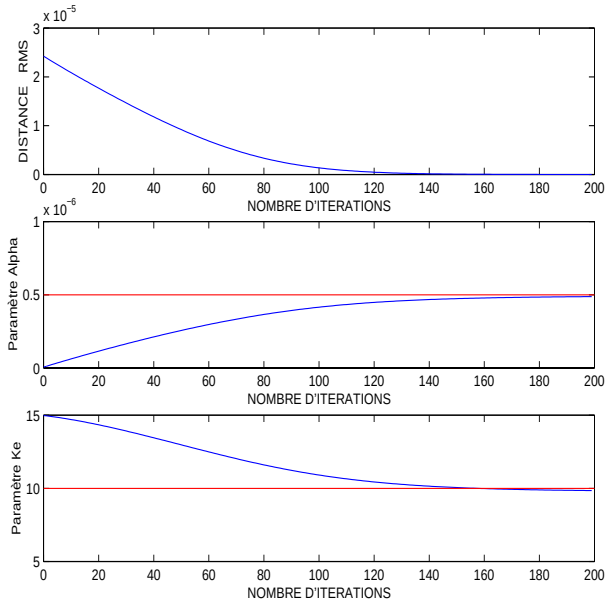

FIG. 6.2 –: *L'affichage de la distance RMS, la convergence des deux paramètres pour le deuxième jeu de données avec les initialisations* $\alpha = 0$ *et* $K_E = 15$.

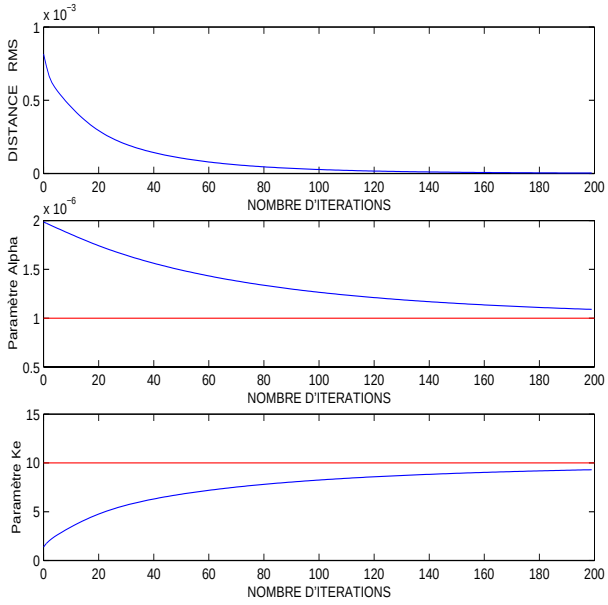

FIG. 6.3 –: *L'affichage de la distance RMS, la convergence des deux paramètres pour le troisième jeu de données avec les initialisations* $\alpha = 20\times10^{-7}$ *et* $K_E = 1$.

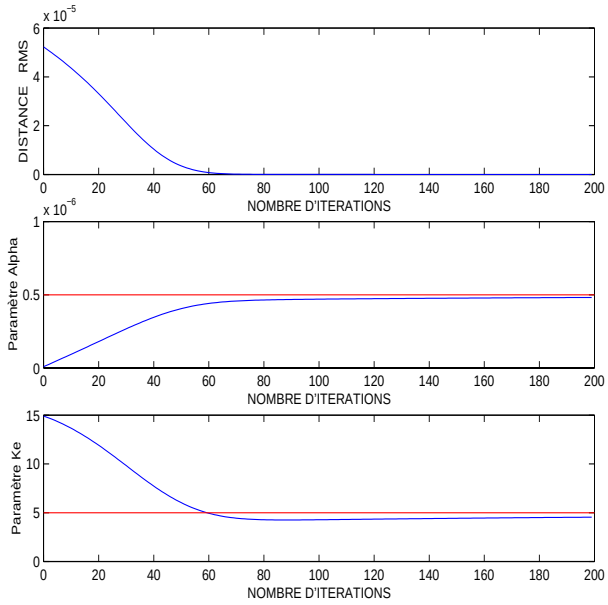

FIG. 6.4 –: *L'affichage de la distance RMS, la convergence des deux paramètres pour le quatrième jeu de données avec les initialisations* $\alpha = 0$ *et* $K_E = 15$.

Chapitre 7

Conclusion et Perspectives

Les recherches sur la couleur de l'océan sont abordées au LODYC depuis 1996. Les premiers résultats ont été obtenus sur l'inversion des radiances marines. Depuis, un grand nombre d'actions prospectives ont été engagée. Des recherches dans ce domaine ont montré l'utilité des réseaux de neurones.

Le caractère intrinsèquement non linéaire des relations trophiques pose un problème méthodologique , concernant la recherche du minimum d'une fonction fortement non linéaire (écart entre les résultats du modèle et les données). Jusqu'à maintenant, peu d'études d'assimilation en biogéochimie océanique ont été menées et elles se sont toutes appuyées sur des méthodes itératives après linéarisation du modèle, en utilisant par exemple le modèle linéaire tangent. Cette procédure ne garantie pas la convergence de la solution, et a des difficultés à appréhender des systèmes fortement non linéaires.

Dans ce stage, au lieu de procéder à une linéarisation d'une fonction non linéaire, nous avons choisi de la modéliser par un réseau de neurone multicouche.

Ainsi la modélisation des modèles biologiques de productivité marine par un réseau multicouche permet de prendre en compte les relations non linéaires entre les données.

Au cours de ce stage, nous avons présenté une méthodologie générale d'assimilation de données variationnelles basée sur la modélisation d'un système complexe modulaire. La générécité de cette méthode et la simplicité des formules que l'on peut déduire de la structure du graphe sous-jacente, permettent de calculer d'une manière simple le gradient d'une fonction définie sur les variables de notre système par rapport à ses variables d'entrée. Nous avons montré sur une application simple d'assimilation de données en biogéochimie océanique que la modélisation par système modulaire couplée avec une représentation de la non linéarité par un réseau de neurone multicouche permet d'avoir des résultats qui s'avèrent encourageants.

Ce stage donne une première présentation de l'assimilation de données par des méthodes neuronales. Il est encore loin de prouver l'efficacité de cette méthode dans le cadre d'application réelle. En effet, pour cela, il faut traiter le cas où le temps T d'évolution du système est grand. Ainsi que la prise en considération les contraintes spatiales (3D). En effet, dans ce stage nous avons traité uniquement les contraintes verticales, c'est à dire que chaque point

dans l'océan dépend uniquement du point qui le précède et de celui qui le succède (profondeur inférieure et profondeur supérieure) alors qu'en réalité, chaque point de l'océan est relié à un domaine 3D.

Annexe A

Quelques classes de notre programme

```
/*************************************************************************
 * classe  noeud est la classe de base des noeuds de notre graphe
 *_______________________________________________________________________
 *************************************************************************/

   class noeud {

public :

int temps,num, nbrpred ;
Val_etat etat ;
float mean,sdev;
noeud  * pred[50] ;

noeud(int temps1, int nump,int nbr)
{
  //cout<< "creation de noeud\n";
temps=temps1; //designe le temps de l'\'evolution d'un noeud
num=nump; //pour le tritopologique
nbrpred=nbr; //nombre de ces predecesseurs
}

 ~noeud()
{
}

void forward()
{
}
```

```
void backward()
{
}
};

/* Nous allons maintenant pr\'esenter une classe qui va \^etre utilis\'ee dans
   les noeuds de type Biologie pour permettre la propagation avant et
   arri\‘ere dans ce types de noeuds. */

/*****************************************************************************
 * class MonReseau
 *---------------------------------------------------------------------------
 *****************************************************************************/

class MonReseau: public Reseau <float , 1>{

  public :
int nbrBloc,etatfile;
FILE *file;
int numres;
    MonReseau (int numRes):Reseau(){
numres=numRes; //numero du Reseau de neurone.

this->Input->NbCellules=nb_ent-1;   //nombre des neurones de la couche d'entr\'ee
this->Hid[0]->NbCellules=nb_hide-1; //nombre des neurones de la couche cach\'ee
this->Output->NbCellules=1;         //nombre des neurones de la couche de sortie

this->ChargePoids();

this->Output->Cell[0]->transtype="lin";
//d\'esigne la fonction d'activation de la couche de sortie
//alors que pour la couche cach\'ee c'est une  sigmoide
  }

~MonReseau ()
  {
  }

  void ChargePoids() {
    cout << "Chargement des poids module unique" << "\n";

    if (numres==1){
    OuvrirFichierPoids("ficpds_11.dat","ficpdsbiais_11.dat");
    ChargerPoidsCouche(Input,Hid[0]);
    FermerFichierPoids();
    OuvrirFichierPoids("ficpds_output1.dat","ficpdsbiais_output1.dat");
    ChargerPoidsCouche(Hid[0],Output);
```

```
    FermerFichierPoids();
    }
else
{
    OuvrirFichierPoids("ficpds_12.dat","ficpdsbiais_12.dat");
    ChargerPoidsCouche(Input,Hid[0]);
    FermerFichierPoids();
    OuvrirFichierPoids("ficpds_output2.dat","ficpdsbiais_output2.dat");
    ChargerPoidsCouche(Hid[0],Output);
    FermerFichierPoids();
    }
  }

void forward(noeud * n);//elle fait la progapagation avant d'un r\'eseau
              // et elle comporte le codage, (voir figure 2.4)

void backward (noeud  * );//elle fait la r\'etro-progapagation  d'un r\'eseau
              // et elle comporte le d\'ecodage, (voir figure 2.4)

void SaveConnect(noeud *n); //sauvegarde les valeurs d'activations
                            //pour faire la r\'etro-propagation

void LoadConnect(noeud *n);//charge les valeurs d'activations
                           //pour faire la r\'etro-propagation

};

/*  Ci-dessous, un mod\'ele d'une classe h\'erit\'ee de notre classe de base Noeud pour
    l'adapter aux noeuds Biologie. Ces noeuds   sont cr\'ees par l'expression
      Bio[i]= new noeudBio (temps,tritopologique,nombre_de_pr\'ed\'ecesseurs,
    R\'eseau_neuronal).  */

/*************************************************************************
 * classe NoeudBio
 *-----------------------------------------------------------------------
 *************************************************************************/
class noeudBio:public noeud
{
  public :
MonReseau * fonc_arc ;

  noeudBio(int tmps, int num,int nbr,MonReseau * res):noeud(tmps, num,nbr)
  {
fonc_arc=res;
  }

~noeudBio()
```

```
  {
  }

forward()
  {
fonc_arc->forward((noeud *) this);
 }

backward()
  {
fonc_arc->backward((noeud *) this);
  }

};
```

Bibliographie

[Bottou,1991] *Bottou L. : Une Approche théorique de l'Apprentissage Connexioniste; Application à la reconnaissance de la parole, Thèse de Doctorat de l'Université de Paris Sud, Centre d'Orsay, 1991.*

[Bottou et Gallinari,1992] *Bottou L., Gallinari P. : A framework for the cooperation of Learning Algorithms,pp. 781-788,1992.*

[Bishop,1995] *Bishop C.M.: Neural Networks for Pattern Recognition, Oxford University Press, 482 p., 1995.*

[Bishop,1994] *Bishop C.M.: Mixture Density Networks. Neural computing research group report NCRG/4288, Aston University, 1994.*

[Carrier et Pearson,1976] *Carrier G.F., Pearson C.E. : Partial Differential Equations, Edition Academic Press,1976.*

[Charpentier,1998] *Charpentier I. : Génération de codes adjoints : Traitement de la trajectoire du modèle direct, INRIA Num3405, 1998.*

[Cheuoua,1999] *Cheuoua J. : Réseaux de neurones et télédetection : Restitution des mesures de vecteurs vents à la surface de la mer, Mémoire d'ingénieur du CNAM de l'Institut Informatique d'Entreprise, 1999.*

[Goutte et Ledoux,1996] *Goutte C., Ledoux C. : Synthèse de techniques de commande connexionniste,1996.*

[Gross, 1997] *Gross L. : Interprétation des mesures satellitaires de la couleur de l'océan par réseaux de neurones, Mémoire de DEA de l'Université de Saint-Quentin-en-Yvelines, 1997.*

[Hsieh et Tang,1998] *Hsieh W.W., Tang B. :Applying Neural Network Models to Prediction and Data Analysis in Meteorology and Oceanography, Dans Bulletin of the American Meteorological Society,Vol 79 N°9, pp. 1855-1870,September 1998.*

[Jordan et Rumelhart,1995] *Jodran M.I., Rumelhart D.E : Forward Models : Supervised Learning with a Distal Teacher, Dans Backpropagation : Theory, Architectures and Applications, pp. 1-34,1995.*

[Levy,1996] *Levy M. : Modélisation des processus Biogéochimiques en méditerranée Nord-Occidentale, Cycle Saisonnier et Variabilité Mésoéchelle, Thèse de Doctorat de l'université Jussieu,1996.*

[Lopez,1998] *Lopez P. : Cours de Graphes, LAAS-CNRS,1998.*

[Loukos,1995] *Loukos H. : Simulation du Cycle Océanique du Carbone dans l'Atlantique Equatorial : Validation de l'Année 1983 (FOCAL),Thèse de Doctorat de l'université Jussieu,1995.*

[Manuel,1997] *Le manuel de SN28 : A connectionist Simulator,1997.*

[Mejia,1992] *Mejia C. E. : Architecture Neuronales pour l'Approximation des Fonction de Transfert : application à la télédétection, Thèse de Doctorat de l'Université de Paris Sud, Centre d'Orsay, 1992.*

[Murray et Smith,1993] *Murray, Smith : Neural networks for statistical modeling, VNR, chap8,1993.*

[Press et al,1995] *Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B. : Numerical Recipes in C : The art of scientific Computing, Second Edition, Published by the Press Syndicate of the University of Cambridge,1995.*

[Rumelhart et al,1996] *Rumelhart D.E., Hinton G.E., Williams R.J. : Learning Internal Representations by Error Propagation, Dan Parallel distributed processing : explorations in the microstructures of cognition. CH.8, MIT Press, Vol. 1, 1986.*

[Rumelhart et al,1995] *Rumelhart D.E., Durbin R., Golden R., Chauvin Y. : Backpropagation : The Basic Theory, Dans Backpropagation : Theory, Architectures and Applications, pp. 1-34,1995.*

[Schwartz,1987] *Schwartz L. : Méthodes mathématiques pour les sciences physiques, Edition Hermann,1987.*

[Talagrand,1991] *Talagrand O. : The Use of Adjoint Equat ions in Numerical Modelling of the Atmospheric Circulation, Published in Proceeding of Workshop on Automatic Differentiation of Algorithms : Theory, Implementation and Application (Breckenridge, Colorado, 7-9, 1991.*

[Thiria et al,1997] *Thiria S., Lechevallier Y., Gascuel O., Canu S. : Statistique et méthodes neuronales, édition Dunod,1997.*

[Remy,1999] *Remy E. : Assimilation variationnelle de données tomographiques simulée dans des modèles de circulation océanique, Thèse de Doctorat de l'université Jussieu ,Juillet 1999.*