Chapitre 3

LES RESEAUX DE NEURONES RECURRENTS ET DYNAMIQUES

3.1 Introduction

La quête d'un modèle neuronal qui soit à la fois versatile et dont l'apprentissage requière un effort de calcul raisonnable demeure un enjeu primordial dans la recherche connexioniste. Dans cet esprit, nous présentons une classe de réseaux récurrents et dynamiques appelée "Dynamical Recurrent Neural Networks" (DRNN) [Aussem 95b, Aussem 95c, Aussem 95d] qui fédère un grand nombre de réseaux de neurones introduits pour le traitement temporel [Lang 88, Leighton 91, Sastry 94, Tsoi 94, Wan 93b, Williams 89] et fondés sur le modèle de neurone artificiel à fonction de transfert sigmoïdale de McCulloch et Pitts. L'architecture du modèle DRNN est des plus générales puisque des connexions arbitrairement retardées et bouclées sont permises. Le vecteur d'état du système est obtenue à l'issue d'une phase de relaxation vers un état d'équilibre stable dont nous discuterons l'existence. L'attraction majeure de ce réseau de neurones est sa mémoire adaptable lui permettant d'implémenter directement un système dynamique. Les interactions retardées mutuelles entre les neurones formels permettent de conserver une mémoire sous forme d'informations codées de la séquence présentée en entrée. En vertu de la récursion temporelle et spatiale, le comportement du modèle est décrit par un système d'équations aux différences dont les variables internes sont les activations des neurones formels cachés. Ce comportement dynamique contraste avec les réseaux non-bouclés gouvernés par des équations algébriques entre le vecteur d'entrée et de sortie.

Le modèle défini, il reste à mettre au point une procédure d'adaptation des paramètres internes. A cette fin, deux algorithmes récursifs de rétro-propagation sont développés puis examinés minutieusement sous l'angle de la complexité. Le cadre formel, sur lequel repose le développement de ces procédures, emprunte à l'analyse mathématique des systèmes dynamiques tel qu'il a été formulé par Werbos en 1974 [Werbos 74]. Le lecteur intéressé trouvera une reformulation concise de l'expansion des dérivées partielles dans les systèmes dynamiques ainsi que leurs applications dans [Piche 94].

Ces algorithmes portent le nom de propagation en "chaînage avant" et propagation en "chaînage arrière" en référence au sens de propagation des signaux d'erreur dans le réseau adjoint [Baldi 95, Srinivasan 94]. Nous démontrons que la propagation du gradient en "chaînage arrière" exhibe un comportement exponentiel et des conditions suffisantes de convergence sont établies. La décroissance exponentielle du gradient d'erreur rétro-propagé permet une expansion limitée du réseau adjoint dans le temps, sans pour autant nuire au calcul du gradient et altérer la trajectoire idéale du système; ce résultat est le cœur de cette thèse. La majeure partie de ce chapitre est consacrée au développement d'un algorithme itératif procédant en "chaînage arrière" appelé Temporal Recurrent Back-Propagation (TRBP) lequel réduit considérablement le volume des ressources nécessaires à l'implémentation d'une procédure en "chaînage avant". Outre sa faible complexité, TRBP est simple à mettre en œuvre car la propagation dans l'adjoint est symétrique à la propagation dans le réseau de neurones.

Nous nous gardons bien toutefois d'ériger le DRNN muni de TRBP en modèle universel. La rapide décroissance du gradient dans l'adjoint confirme la difficulté de la descente du gradient à capturer des contingences temporelles de longues durées [Bengio 94]. En établissant des conditions suffisantes pour assurer la stabilité de la procédure d'ajustement des poids, nous apporterons des éléments nouveaux qui mettent à jour la faiblesse des algorithmes de descente du gradient en général.

3.2 Le DRNN

3.2.1 La représentation interne de la mémoire

En vue de remédier aux limitations des réseaux non-bouclés dans le contexte de la prédiction, nous montrerons au Chapitre 4 que la récurrence doit être explicitement introduite au sein du réseau. Les tâches temporelles exigent le stockage et la mise à jour d'informations contextuelles extraites de la séquence des valeurs d'entrée. Or, la plage temporelle qui recelle l'informations pertinente peut s'étendre sur des intervalles de longueur indéterminée, si bien que la technique de la fenêtre temporelle devient ineffective. En conséquence, le réseau doit être muni d'une mémoire interne remontant aussi loin que possible dans la passé.

Dans ce dessein, une modification essentielle est apportée au réseau de neurones du type de Rumelhart [Rumelhart 86] en remplaçant le poids synaptique par un filtre linéaire à réponse impulsionnelle finie (FIR). A une impulsion de durée finie correspond une réponse de durée finie. Soit $x_i(n)$ l'activation d'un neurone i à l'instant n; $x_i(n)$ s'exprime en fonction des entrées du neurone par

$$x_i(n) = f(\sum_j \sum_{d=0}^{D} w_{ji}^d x_j(n-d)) = f(\sum_j \mathbf{w}_{ji} * \mathbf{x}_j(n)) \tag{3.1}$$

où $\mathbf{w}_{ji} = [w_{ji}^0, w_{ji}^1, \ldots, w_{ji}^D]^T$ et $\mathbf{x}_j(n) = [x_j(n), x_j(n-1), \ldots, x_j(n-D)]^T$. Nous avons déjà entrevu au § 2.7 la sous-classe des réseaux FIR non-bouclés. Cette modification, somme toute élémentaire, permet aux interactions neuronales de représenter un vaste choix de mémoires formelles. Les coefficients du filtres doivent être ajustés de façon à sélectionner le type de mémoire le mieux adapté au problème donné.

Cette approche est motivée par l'idée que la mémoire puisse permettre au réseau de trouver une représentation interne qui permette de capturer les régularités essentielles sur lesquelles repose l'évolution d'un processus. De surcroît, la fenêtre temporelle, ou "time-lagged vector", n'est plus nécessaire; il suffit d'injecter la dernière valeur de la séquence en entrée du réseau. Le réseau, conduit par une procédure d'apprentissage efficace, adaptera sa mémoire interne pour ne retenir que l'information pertinente à la reconstruction d'un vecteur d'état interne. Williams et Zipser ont illustré la richesse d'une telle architecture en utilisant la sous-classe $w_{ij}^d = 0, \forall d \neq 1$. Outre la reconnaissance d'une séquence de symboles sur une période potentiellement illimitée, ce modèle est capable d'émuler un automate à états finis [Williams 89].

La méthode des connexions FIR a été introduite en 1993 par Wan dans le cadre des réseaux non-bouclés, et nous nous proposons de l'étendre aux réseaux bouclés. La prédiction apparaît, de toute évidence, comme le domaine de prédilection de cette classe de réseaux de neurones bouclés puisque la version non-bouclée s'est illustrée lors d'un concours international, comme le meilleur modèle prédictif à court terme [Gershenfeld 93]. En dépit de ses capacités fort attrayantes, le réseau FIR non-bouclé n'échappe pas aux limitations des modèles NARMA dont nous discuterons abondamment au Chapitre 4. Il ne se distingue des réseaux statiques non-bouclés que par un représentation compacte et un algorithme d'apprentissage opérant en temps réel. Conceptuellement, cette architecture ne diffère en rien du *time-delay neural network* de Lang [Lang 88] ou du *autoregressive network* de Leighton [Leighton 91].

Finalement, le modèle que nous présentons est un réseau FIR complètement bouclé mais qu'on ne s'y trompe pas, toute la difficulté réside dans l'apprentissage et non dans la définition d'une architecture. Car à la différence des réseaux dont nous avons discuté jusqu'à présent, la rétro-propagation classique du gradient est exclue du fait de la récursion *temporelle* et *spatiale*. C'est la raison pour laquelle le pas n'a pas été franchi jusqu'à présent. L'objet de cette étude est de lever, d'une manière formelle et rigoureuse, les obstacles rencontrés dans le calcul exacte du gradient d'erreur de ce modèle dynamique et récurrent.

3.2.2 Une dynamique à relaxation

Dans cette section, les ingrédients nécessaires à la spécification du réseau de neurones dynamique et récurrent (DRNN) sont présentés. Considérons un réseau totalement bouclé de N neurones à réponse continue dans $[0, 1]$. L'élément matriciel w_{ij}^d représente le poids de la connexion retardée de d unités de temps, reliant le neurone i au neurone j. D représente le délai maximal, i.e. l'ordre du filtre. On appelle activation neuronale la réponse d'un neurone, et $\mathbf{v}_k$ ($\mathbf{v}_k \in R^{[N\times 1]}$) est le vecteur d'activité des neurones à l'instant k. Comme à l'ordinaire, une sous-classe de ces neurones est dédiée à la réception et à la propagation du vecteur d'entrée à travers le réseau. Ces neurones portent le nom de neurones d'entrée et possèdent une activation fixée par les composantes du vecteur d'entrée $\mathbf{x}_k$ ($\mathbf{x}_k \in R^{[N\times 1]}$).

$w_k^d(i,j)$	poids de la connexion retardée de d entre le neurone i et le neurone j à l'instant k
$\mathbf{w}_k^d(j)$	$= [w_k^d(1,j), \ldots, w_k^d(N,j)]^T$
$\mathbf{W}_k^d$	$= [\mathbf{w}_k^d(1), \mathbf{w}_k^d(2), \ldots, \mathbf{w}_k^d(N)]$
$\mathbf{s}_k$	$= \sum_{d=0}^{D} \mathbf{W}_k^{d^T} \mathbf{v}_{k-d}$
$\mathbf{v}_k$	$= g(\mathbf{s}_k)$ le vecteur d'activation des neurones
$\mathbf{x}_k$	vecteur d'entrée du réseau à l'instant k

TAB. 3.1 - : Notations du DRNN.

Hormis les signaux externes, les neurones d'entrée ne reçoivent pas de signaux émanant d'autres neurones, i.e. $w_{ij}^d = 0$ si j ne désigne pas un neurone d'entrée. D'une façon similaire, certains neurones, dits de sortie, possèdent une activation cible, ou valeur désirée, $\mathbf{d}_k$ ($\mathbf{d}_k \in R^{[N\times 1]}$). Les neurones n'ayant pas de relation avec le monde extérieur sont les neurones cachés. Enfin, le rôle de biais incombe par définition au neurone d'indice 0. Ce dernier est perçu comme une neurone d'entrée supplémentaire dont l'activation est fixée à 1.0. Pour des raisons de commodité, nous ne distinguerons pas, dans les formules suivantes, les classes de neurones que nous venons d'introduire. Les notations sont exposées en Table 3.1.

Enfin, le vecteur de poids des connexions retardées de d unités de temps, et destinées au neurone j à l'itération k, est spécifié par $\mathbf{w}_k^d(j) = [w_k^d(1,j), \ldots, w_k^d(N,j)]^T$. De la même façon, la matrice $N \times N$ définie par $\mathbf{W}_k^d = [\mathbf{w}_k^d(1), \mathbf{w}_k^d(2), \ldots, \mathbf{w}_k^d(N)]$, contient tous les poids des connexions retardées d'un délai $d \le D$ à l'itération k. Les vecteurs initiaux $\mathbf{W}_0^d, \forall d$ sont généralement produient grâce à un générateur de nombres aléatoires. Cependant, nous verrons par la suite que les $\mathbf{W}_0^d$ doivent satisfaire des conditions préliminaires pour être certain de la stabilité de la procédure d'adaptation des poids.

Nous déterminons les paramètres du système, à chaque étape k, en relaxant le système d'équations différentielles

$$\begin{aligned} \mathbf{T}\frac{d\mathbf{v}_k(t)}{dt} &= -\mathbf{v}_k(t) + g(\mathbf{s}_k(t)) + \mathbf{x}_k \\ \mathbf{s}_k(t) &= \sum_{d=1}^{D} \mathbf{W}_k^{d^T} \mathbf{v}_{k-d} + \mathbf{W}_k^{0^T} \mathbf{v}_k(t) \end{aligned} \tag{3.2}$$

Ces équations différentielles, particulièrement adaptées à une implantation VLSI, gouvernent la dynamique du réseau jusqu'à sa convergence vers un point fixe dont nous discuterons l'existence et la stabilité par la suite. Le vecteur $\mathbf{T} = [\tau_1, \ldots, \tau_N]$ est convenablement choisi pour amener dans un temps raisonnable la système vers un état d'équilibre. $\mathbf{s}_k$ désigne le vecteur des entrées internes des neurones à l'itération k et $g()$ est une sigmoïde à valeur positive

$$\begin{aligned} g(\mathbf{s}_k) &= [g(s_k^1), \ldots, g(s_k^n)]^T \\ g(s) &= \frac{1}{1+e^{-\beta s}} \end{aligned} \tag{3.3}$$

Le paramètre β, fixé arbitrairement à 1/2, commande la pente de la sigmoïde.

Il n'est pas garanti que le système Eq. 3.2 converge vers un point d'équilibre stable; un comportement oscillatoire ou une dynamique chaotique ne sont pas exclus [Hertz 91]. Faute de point fixe, une nouvelle matrice de poids doit être réinitialisée. Aussi, l'existence d'au moins un point fixe est d'ordinaire posée en postulat; seules des conditions nécessaires et suffisantes de stabilité seront établies. Par conséquent, si le système (3.2) admet un point d'équilibre, stable ou instable, il vérifie la relation

$$\mathbf{v}_k = g(\sum_{d=0}^{D} \mathbf{W}_k^{d^T} \mathbf{v}_{k-d}) + \mathbf{x}_k \tag{3.4}$$

Il est important d'observer que chaque neurone est tenu de garder en mémoire une trace de ses D dernières activations, ce qui conduit à un volume de stockage de l'ordre de DN valeurs.

A l'inverse des réseaux *synchrones* [Wan 93b, Williams 89] pour lesquels $w_{ij}^d \neq 0$ pour $d > 0$ uniquement, le DRNN possède des connexions non retardées, et de ce fait, nécessite une phase de relaxation. Le DRNN appartient donc à une classe plus générale de réseaux connexionistes.

3.3 La descente du gradient dans le DRNN

Dans les réseaux de neurones non-bouclés, l'ajustement des poids est réalisé sans détour grâce à la propagation directe du gradient d'erreur dans le sens inverse des signaux, c.a.d. des neurones en aval vers les neurones en amont. La rétro-propagation du gradient standard [Le Cun 85, Parker 85, Rumelhart 86, Werbos 74] n'est pas applicable au DRNN puisque ce dernier admet des connexions bouclées et retardées. Clairement, la variation de la valeur d'un paramètre à l'instant présent, se répercute sur l'évolution ultérieure du système. L'influence de l'état courant du réseau sur sa trajectoire future est une caractéristique désirable, faute de quoi le réseau ne peut conserver une mémoire du passé. En contre partie, il faut remonter dans la passé pour déceler l'origine de l'erreur courante. En somme, il reste à pourvoir le DRNN d'une procédure d'apprentissage qui, par un ajustement approprié des poids, adapte la mémoire du réseau.

Comme nous l'avons vue en § 2.7.1, la rétro-propagation dans le temps (BPTT) [Rumelhart 86] implémente, en essence, la chaîne d'expansion des dérivées partielles dans la direction opposée aux flux des signaux, *spatialement* mais aussi *temporellement*. On obtient un grand réseau statique non-bouclé multi-couches dont la taille croît d'une couche par itération. L'inconvénient majeur de cette approche réside dans le choix souvent arbitraire du nombre de duplications du réseau. Ce choix peut alternativement se faire à la lumière des caractéristiques temporelles de la série lorsque celles-ci sont connues et d'étendue limitée. Quand bien même ce nombre de duplications est élevé, certaines séries requièrent une fenêtre temporelle de taille potentiellement illimitée. C'est un cas fréquent en analyse syntaxique où les séquences du durée indéterminée sont interprétées en fonction des symboles aux extrémités $(a, \ldots, b)$. La troncature systématique du gradient d'erreur exacte et la grande capacité de mémorisation requise pénalisent l'algorithme BPTT, de plus en plus délaissé au profit de procédures plus attractives dont TRBP, exposé dans ce chapitre.

La quête d'une procédure permettant l'approximation précise du gradient, et ne nécessitant pas la mobilisation d'une quantité excessive de ressources informatiques, est au centre de cette étude. Aussi, nous allons recourir à deux stratégies de descente du gradient fondées sur les mêmes concepts mathématiques, mais dont les déclinaisons conduisent à des implémentations radicalement différentes. Les concepts mathématiques reposent sur le cadre mathématique formulé par Werbos en 1974 [Werbos 74], et traitent de l'expansion des dérivées partielles pour obtenir le gradient d'erreur dans un système dynamique opérant en temps discret. Le lecteur désireux d'acquérir une compréhension plus approfondie de ces principes pourra se référer à [Werbos 74, Piche 94]. La dynamique du DRNN, qui repose sur la relaxation du réseau à chaque itération, ne rentre cependant pas dans le cadre des systèmes dynamiques traités par Werbos. Force nous est d'adapter les chaînes d'expansion des dérivées partielles afin d'élargir leur champs d'application au DRNN.

3.3.1 L'expansion des dérivées partielles

Pour déterminer les dérivées partielles qui surgissent dans le calcul du gradient, il est judicieux de représenter le DRNN sous une forme standard, couramment adoptée en théorie du contrôle des systèmes dynamiques [Narendra 91, Srinivasan 94]. L'entrée du réseau est composée de deux composantes : l'entrée externe $\mathbf{x}_k$ ainsi que D versions retardées de ce vecteur $\mathbf{x}_{k-1}, \ldots, \mathbf{x}_{k-D}$ d'une part, et D vecteurs d'état retardés du système $\mathbf{v}_{k-1}, \ldots, \mathbf{v}_{k-D}$ d'autre part.

En transposant la mémoire interne du réseau à l'extérieur comme une entrée supplémentaire, on cherche à exprimer le lois qui gouvernent l'évolution du DRNN par

$$\mathbf{v}_k = f(\mathbf{x}_k, \ldots, \mathbf{x}_{k-D}, \mathbf{v}_{k-1}, \mathbf{v}_{k-2}, \ldots \mathbf{v}_{k-D}, \mathbf{W}_k^0, \ldots, \mathbf{W}_k^D) \tag{3.5}$$

où la fonction $f()$ ne contient aucun délai [Narendra 91, Piche 94]. Mais comme la fonction $f()$ n'est pas définie analytiquement dans le cadre du DRNN, nous sommes contraint d'avoir recours à Eq. 3.4 qui est récursive par rapport aux termes $\mathbf{v}_k$. La représentation est visualisée en Fig. 3.1. Il faut donc étendre le formalisme de (3.5). Dans cet esprit, considérons un ensemble de n variables $z_1, \ldots, z_n$ dont les valeurs sont déterminées par un ensemble de n équations *ordonnées*

$$z_i = f_i(z_1, \ldots, z_{i-1}, z_i) \tag{3.6}$$

dans lesquelles chaque variable z_i n'est pas seulement fonction de $z_1, \ldots, z_{i-1}$ mais aussi d'elle-même.

Pour permettre le calcul des dérivées partielles, il faut spécifier les variables assimilées à des constantes, des autres. Typiquement, lorsque rien n'est spécifié, nous supposerons

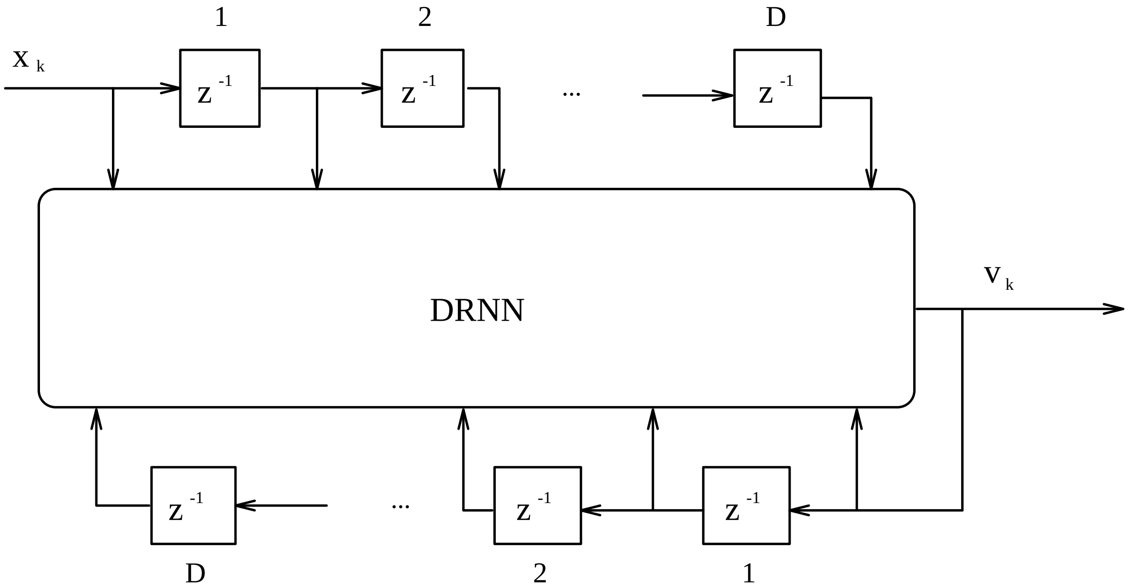

FIG. 3.1 - : Représentation du DRNN comme un système dynamique.

que ces variables sont maintenues constantes, excepté celles qui apparaissent au dénominateur des dérivées partielles. Une dérivée partielle ordonnée est une dérivée partielle pour laquelle les variables constantes sont déterminées grâce à un ensemble d'équations ordonnées. Selon la notation mathématique consacrée, on a

$$\frac{\partial^+ z_j}{\partial z_i} = \left[\frac{\partial z_j}{\partial z_i}\right]_{\{z_1,\ldots,z_{i-1}\}}. \tag{3.7}$$

Supposons que $I - \partial f_j(z)/\partial z_j$ soit inversible, $\partial z_j/\partial z_i$ pour $(i < j)$ est alors obtenue par

$$\frac{\partial z_j}{\partial z_i} = \frac{\partial f_j(z)}{\partial z_i} + \frac{\partial f_j(z)}{\partial z_j}\frac{\partial z_j}{\partial z_i} = (I - \frac{\partial f_j(z)}{\partial z_j})^{-1}\frac{\partial f_j(z)}{\partial z_i} \tag{3.8}$$

De cette définition, il vient les relation suivantes

$$\frac{\partial^+ z_{i+1}}{\partial z_i} = \frac{\partial z_{i+1}}{\partial z_i} \tag{3.9}$$

et

$$\frac{\partial^+ z_j}{\partial z_i} = 0 \quad \text{pour} \quad j < i \tag{3.10}$$

Lorsque $j > i+1$, les dérivées ordonnées s'obtiennent par les lois d'expansion suivantes

$$\frac{\partial^+ z_j}{\partial z_i} \quad = \quad \frac{\partial z_j}{\partial z_i} + \sum_{k=i+1}^{j-1} \frac{\partial^+ z_j}{\partial z_k}\frac{\partial z_k}{\partial z_i}$$

$$= \sum_{k=i+1}^{j} \frac{\partial^+ z_j}{\partial z_k} (I - \frac{\partial f_k(z)}{\partial z_k})^{-1} \frac{\partial f_k(z)}{\partial z_i} \quad (3.11)$$

et

$$\begin{aligned} \frac{\partial^+ z_j}{\partial z_i} &= \frac{\partial z_j}{\partial z_i} + \sum_{k=i+1}^{j-1} \frac{\partial z_j}{\partial z_k} \frac{\partial^+ z_k}{\partial z_i} \\ &= \sum_{k=i}^{j-1} (I - \frac{\partial f_j(z)}{\partial z_j})^{-1} \frac{\partial f_j(z)}{\partial z_k} \frac{\partial^+ z_k}{\partial z_i} \end{aligned} \quad (3.12)$$

Avant de procéder au calcul du gradient, un commentaire sur les notations mathématiques employées. Nous supposerons implicitement par la suite que la dérivée partielle $\partial \mathbf{a}/\partial \mathbf{b}$, où $\mathbf{a} \in R^{[A\times 1]}$ et $\mathbf{b} \in R^{[B\times 1]}$, est la matrice jacobienne de dimension $R^{[A\times B]}$. Avec (3.11) et (3.12), nous sommes équipés avec le bagage mathématique nécessaire pour mener le développement des algorithmes.

3.3.2 Des algorithmes à propagation avant et arrière

L'objet de l'apprentissage est d'ajuster la matrice de poids $\{w_{ij}^d\}$ de façon à faire évoluer le réseau, sous l'action des entrées $\{\mathbf{x}_k\}$, d'un état $\mathbf{v}_0$ vers une suite de points fixes dont les composantes de sortie s'approchent des valeurs désirées $\{\mathbf{d}_k\}$. Comme à l'accoutumée, on cherche à minimiser l'erreur quadratique

$$E = \frac{1}{2} \sum_{k=0}^{k_f} \mathbf{e}_k^T \mathbf{e}_k \quad (3.13)$$

où $\mathbf{e}_k = \mathbf{d}_k - \mathbf{v}_k$ mesure le vecteur d'erreur le long de la trajectoire, ou époque, de l'itération 0 à k_f. Rappelons que $e_k(i) = d_k(i) = 0$ si le neurone i n'est pas un neurone de sortie. Supposons que l'adaptation s'effectue au terme de la trajectoire, le DRNN est gouverné par un ensemble d'équations ordonnées

$$\left\{ \begin{array}{lcl} \boldsymbol{\mathcal{W}}_0 &=& \boldsymbol{\mathcal{W}} \\ \mathbf{v}_0 &=& f(\mathbf{x}_0, \ldots, \mathbf{x}_{-D}, \mathbf{v}_0, \ldots \mathbf{v}_{-D}, \boldsymbol{\mathcal{W}}_0) \\ & \vdots & \\ \boldsymbol{\mathcal{W}}_{k_f} &=& \boldsymbol{\mathcal{W}} \\ \mathbf{v}_{k_f} &=& f(\mathbf{x}_{k_f}, \ldots, \mathbf{x}_{k_f-D}, \mathbf{v}_{k_f}, \ldots \mathbf{v}_{k_f-D}, \boldsymbol{\mathcal{W}}_{k_f}) \end{array} \right. \quad (3.14)$$

où $\boldsymbol{\mathcal{W}}_k$ est un vecteur de taille $(D+1)N^2$, constitué de la concaténation des vecteurs $\mathbf{w}_k^d(j)$ pour $j = 1, \ldots, N$ et $d = 0, \ldots, D$. L'important est que le vecteur $\boldsymbol{\mathcal{W}}_k$ contienne tous les paramètres adaptatifs du système à chaque itération k, et non sa forme particulière. Un nouveau vecteur de poids est généré à l'itération k_f selon

$$\Delta \mathcal{W} = -\eta \frac{\partial^+ E}{\partial \mathcal{W}} \tag{3.15}$$

où η, le pas d'apprentissage, est une constante positive choisie convenablement. Puisque l'erreur est calculée au terme de l'époque, E est la dernière variable du système ordonné. L'objectif que nous poursuivons est de déterminer analytiquement $\partial^+ E/\partial \mathcal{W}$. L'application des lois d'expansion (3.11) et (3.12) pour le calcul de $\partial^+ E/\partial \mathcal{W}$ conduit à deux algorithmes distincts que nous allons décrire exhaustivement. Ces schémas d'adaptation portent le nom d'algorithme *récursif* et algorithme à *balayage arrière* en théorie des systèmes dynamiques [Piche 94]. Dans un esprit de clarté, nous adoptons toutefois une terminologie plus intuitive de propagation *récurrente avant* et *récurrente arrière*, qui reflète la direction du flot des dérivées partielles par rapport au flot des signaux.

3.4 La propagation récurrente avant

L'application répétée de la seconde loi d'expansion (3.12) conduit à l'algorithme dit de propagation récurrente avant. Grâce au système ordonné (3.14), nous pouvons développer $\partial^+ E/\partial w$ où w est un scalaire de la matrice $\mathcal{W}$

$$\frac{\partial^+ E}{\partial w} = \frac{\partial E}{\partial w} + \sum_{k=0}^{k_f} \left(\frac{\partial E}{\partial \mathbf{v}_k} \frac{\partial^+ \mathbf{v}_k}{\partial w} + \frac{\partial E}{\partial \mathbf{W}_k} \frac{\partial^+ \mathbf{W}_k}{\partial w} \right) \tag{3.16}$$

Le terme $\partial E/\partial w$ et les composantes du vecteur $\partial E/\partial \mathbf{W}_k$ sont nuls car E n'est pas une fonction directe de w et $\mathbf{W}_k$. Nous pouvons donc réécrire (3.16) de la façon suivante

$$\frac{\partial^+ E}{\partial w} = \sum_{k=0}^{k_f} \frac{\partial E}{\partial \mathbf{v}_k} \frac{\partial^+ \mathbf{v}_k}{\partial w} = \sum_{k=0}^{k_f} \mathbf{e}_k \frac{\partial^+ \mathbf{v}_k}{\partial w} \tag{3.17}$$

Le terme $\partial^+ \mathbf{v}_k/\partial w$ est obtenu en utilisant la seconde loi d'expansion

$$\frac{\partial^+ \mathbf{v}_k}{\partial w} = \frac{\partial \mathbf{v}_k}{\partial w} + \sum_{j=0}^{k-1} \left(\frac{\partial \mathbf{v}_k}{\partial \mathbf{W}_j} \frac{\partial^+ \mathbf{W}_j}{\partial w} + \frac{\partial \mathbf{v}_k}{\partial \mathbf{v}_j} \frac{\partial^+ \mathbf{v}_j}{\partial w} \right) \tag{3.18}$$

Les composantes du vecteur $\partial \mathbf{v}_k/\partial w$ sont nulles. Le terme $\partial \mathbf{v}_k/\mathbf{W}_j$ est non nul seulement pour $k = j$. Il vient que le premier terme de la somme s'écrit $\partial \mathbf{v}_k/\partial w_k$. Par ailleurs, le terme $\partial \mathbf{v}_k/\mathbf{v}_j$ est non nul quand $k - j \leq D$. Appliquons ces observations à (3.18), il résulte

$$\frac{\partial^+ \mathbf{v}_k}{\partial w} = \frac{\partial \mathbf{v}_k}{\partial w_k} + \sum_{j=1}^{D} \frac{\partial \mathbf{v}_k}{\partial \mathbf{v}_{k-j}} \frac{\partial^+ \mathbf{v}_{k-j}}{\partial w} \tag{3.19}$$

$\partial \mathbf{v}_k/\partial \mathbf{v}_j$ pour $k - D \leq j < k$ est obtenu en différenciant (3.2). Couplé à (3.8), il vient

$$\frac{\partial \mathbf{v}_k}{\partial \mathbf{v}_{k-j}} = [\mathbf{I} - \mathbf{G'}_k \mathbf{W}_k^{0\,T}]^{-1} \mathbf{G'}_k \mathbf{W}_k^{j\,T} \tag{3.20}$$

où $\mathbf{G'}_k$ est donnée par

$$\mathbf{G'}_k = 2\beta \mathbf{G}_k(1 - \mathbf{G}_k) \tag{3.21}$$

avec $\mathbf{G}_k$ définie comme la matrice diagonale $N \times N$ construite à partir de $g(s_k(j))$ pour $j = 1, \ldots, N$. Rappelons que $g'(s) = 2\beta g(s)(1 - g(s))$ et que $\beta = 1/2$.

Finalement, la substitution de (3.20) dans (3.19) donne

$$\frac{\partial^+ \mathbf{v}_k}{\partial w} = \frac{\partial \mathbf{v}_k}{\partial w_k} + \sum_{j=1}^{D} [\mathbf{I} - \mathbf{G'}_k \mathbf{W}_k^{0\,T}]^{-1} \mathbf{G'}_k \mathbf{W}_k^{j\,T} \frac{\partial^+ \mathbf{v}_{k-j}}{\partial w} \tag{3.22}$$

en dérivant par rapport au vecteur $\boldsymbol{\mathcal{W}}$, il vient

$$\frac{\partial^+ \mathbf{v}_k}{\partial \boldsymbol{\mathcal{W}}} = [\mathbf{I} - \mathbf{G'}_k \mathbf{W}_k^{0\,T}]^{-1} [\frac{\partial f_k}{\partial \boldsymbol{\mathcal{W}}_k} + \sum_{j=1}^{D} \mathbf{G'}_k \mathbf{W}_k^{j\,T} \frac{\partial^+ \mathbf{v}_{k-j}}{\partial \boldsymbol{\mathcal{W}}}] \tag{3.23}$$

où f_k désigne la fonction $f()$ lorsqu'elle est appliquée à l'instant k. Cette équation est itérée récursivement pour calculer les gradients associés aux neurones pour l'époque entière avec les conditions initiales suivantes

$$\frac{\partial^+ \mathbf{v}_{-1}}{\partial \boldsymbol{\mathcal{W}}}, \ldots, \frac{\partial^+ \mathbf{v}_{-D}}{\partial \boldsymbol{\mathcal{W}}} = \mathbf{0} \tag{3.24}$$

La dérivée $\partial \mathbf{v}_k / \partial \boldsymbol{\mathcal{W}}$ peut être obtenue aisément en appliquant (3.8) à (3.4). Le calcul de $\partial E / \partial \boldsymbol{\mathcal{W}}$ est effectué en itérant l'équation aux différences (3.23) et en injectant le résultat dans (3.17). La procédure est répétée jusqu'à ce que le système mémorise correctement les associations temporelles. Cet algorithme à propagation avant étend l'algorithme real-time-recurrent learning algorithm (RTRL) de Williams et Zipser [Williams 89] au DRNN. On remarque que les dérivées apparaissant dans (3.23) sont les sorties du réseau original *linéarisé* avec les spécifications suivantes :

- une inversion matricielle est requise due à la récursion dans le réseau original.

- les entrées $\mathbf{x}_k$ sont replacées par le produit de $[\mathbf{I} - \mathbf{G'}_k \mathbf{W}_k^{0\,T}]^{-1}$ avec les dérivées partielles $\partial f_k / \partial \boldsymbol{\mathcal{W}}_k$.

- la fonction de transfert des neurones n'est plus sigmoïdale mais linéaire.

Cette procédure est fortement grevée par la masse de calcul et de mémoire requise. En premier lieu, chaque dérivée doit être stockée ce qui entraîne une capacité de mémorisation de l'ordre de $O(DN^3)$. Ensuite une quantité draconienne d'opérations, de l'ordre de $O(D^2N^4)$ est nécessaire à chaque itération car l'adaptation les DN^3 dérivées requiert chacune $O(DN)$ opérations. Enfin, une inversion matricielle est effectuée ce qui résulte dans la perte de localité en ce sens que chaque neurone doit avoir connaissance de la matrice de poids.

Des heuristiques pour réduire la complexité

Trouver un équilibre judicieux entre complexité et efficacité d'une procédure d'apprentissage a toujours été un enjeu central au sein de la communauté connexioniste. La difficulté de mettre au point un algorithme pour le DRNN qui nécessite des ressources acceptables provient de la forme de Eq. 3.23. Soit une mémorisation globale de la table des dérivées partielles est nécessaire, ou bien des simplifications restrictives sont faites ce qui en retour altère l'estimation des gradients d'erreur. Essentiellement, cela revient à substituer un gain de temps par un gain en performance. Nous ne sommes jamais libérés de ce dilemme complexité/efficacité[1]. En gardant ce compromis à l'esprit, des stratégies évidentes peuvent être envisagées dans l'intention de réduire substantiellement la complexité, parmi lesquelles on compte les suivantes :

- remplacer $\partial \mathbf{v}_{k-j}/\partial w, j > 0$ par zéro revient à ignorer l'influence de l'état courant à l'instant k sur les futures erreurs $\mathbf{e}_{k+n}, n > 0$. Aussi, le réseau n'effectue pas de "crédit assignment" dans le temps mais utilise les états passés comme des entrées courantes, ce qui en retour limite sa capacité à résoudre des problèmes temporels épineux. Cette simplification a été largement utilisée au tout début de la rétropropagation [Elman 90, Jordan 86].

- une approche plus originale consiste à spéculer sur la vitesse de variation des dérivées partielles $\partial \mathbf{v}_k/\partial \boldsymbol{\mathcal{W}}$ le long d'une trajectoire de longueur D de façon à approximer $\partial \mathbf{v}_{k-n}/\partial \boldsymbol{\mathcal{W}}$ par

$$\frac{\partial \hat{\mathbf{v}}_{k-n}}{\partial \boldsymbol{\mathcal{W}}} = \frac{\partial \mathbf{v}_k}{\partial \boldsymbol{\mathcal{W}}}, \quad \text{pour} \quad n \leq D \tag{3.25}$$

  Cette assertion repose sur l'observation que les dérivées successives sont fréquemment positivement corrélées. La substitution de (3.25) dans (3.23) permet de se débarrasser du stockage excessif puisqu'on a

$$\frac{\partial \mathbf{v}_k}{\partial \boldsymbol{\mathcal{W}}} = [\mathbf{I} - \sum_{d=0}^{D} \mathbf{G'}_k \mathbf{W}_k^{d^T}]^{-1} \frac{\partial f_k}{\partial \boldsymbol{\mathcal{W}}_k} \tag{3.26}$$

Nous ne discutons pas de cela plus en détail car une procédure plus attractive peut être proposée pourvu que nous inversions la direction des signaux d'erreur.

[1]Certains se sont ingéniés à mêler BBTT et RTRL en un seul algorithme de manière à établir un équilibre paramètrable entre les deux [Schmidhuber 92]. La complexité se réduit alors à $O(N^3)$ au lieu de $O(N^4)$. L'idée consiste à décomposer les exemples en blocs de taille $O(N)$ et à appliquer BPTT dans les blocs et RTRL à la jonction.

3.5 La propagation récurrente arrière

Comme nous l'avons montré auparavant, l'application répetée de la seconde loi d'expansion des dérivées partielles conduit à un algorithme récursif à propagation avant. Inversement, le calcul de $\partial^+ E/\partial w$ en utilisant la première loi (3.12) conduit à un algorithme à propagation arrière qui, à défaut d'inversion matricielle, nécessite la relaxation d'un réseau adjoint. L'analyse exhaustive de cet algorithme nous éclairera sur la mise au point d'une procédure attractive, tant sur le point de la complexité que de l'implémentation, appelée la rétro-propagation temporelle récurrente.

L'expansion de $\partial^+ E/\partial w$ grâce à (3.11) s'écrit

$$\frac{\partial^+ E}{\partial w} = \frac{\partial E}{\partial w} + \sum_{k=0}^{k_f} \left(\frac{\partial^+ E}{\partial \mathbf{v}_k} \frac{\partial \mathbf{v}_k}{\partial w} + \frac{\partial^+ E}{\partial \mathbf{W}_k} \frac{\partial \mathbf{W}_k}{\partial w} \right) \tag{3.27}$$

Le terme $\partial E/\partial w$ et les composantes du vecteur $\partial \mathbf{v}_k/\partial w$ sont nulles car l'erreur E et $\mathbf{v}_k$ ne sont pas des fonctions directes de w et $\mathbf{W}_k$. En conséquence, (3.27) devient

$$\frac{\partial^+ E}{\partial w} = \sum_{k=0}^{k_f} \frac{\partial^+ E}{\partial \mathbf{W}_k} \frac{\partial \mathbf{W}_k}{\partial w} = \sum_{k=0}^{k_f} \frac{\partial^+ E}{\partial w_k} \tag{3.28}$$

Une fois encore, l'expansion de $\partial^+ E/\partial w_k$ conduit à

$$\frac{\partial^+ E}{\partial w_k} = \frac{\partial E}{\partial w_k} + \sum_{j=k}^{k_f} \frac{\partial^+ E}{\partial \mathbf{v}_j} \frac{\partial \mathbf{v}_j}{\partial w_k} + \sum_{j=k+1}^{k_f} \frac{\partial^+ E}{\partial \mathbf{W}_j} \frac{\partial \mathbf{W}_j}{\partial w_k} \tag{3.29}$$

Le terme $\partial E/\partial w_k$ et les composantes du vecteur $\partial \mathbf{W}_j/\partial w_k$ sont nulles. Le terme $\partial \mathbf{v}_j/\partial w_k$ est non nul dès lors que $k = j$. Aussi, $\partial^+ E/\partial w_k$ se réduit à

$$\frac{\partial^+ E}{\partial w_k} = \frac{\partial^+ E}{\partial \mathbf{v}_k} \frac{\partial \mathbf{v}_k}{\partial w_k} \tag{3.30}$$

La substitution de (3.30) dans (3.29) donne

$$\frac{\partial^+ E}{\partial w} = \sum_{k=0}^{k_f} \frac{\partial^+ E}{\partial \mathbf{v}_k} \frac{\partial \mathbf{v}_k}{\partial w_k} \tag{3.31}$$

Le terme $\partial^+ E/\partial \mathbf{v}_k$ s'obtient par la loi d'expansion

$$\frac{\partial^+ E}{\partial \mathbf{v}_k} = \frac{\partial E}{\partial \mathbf{v}_k} + \sum_{j=k+1}^{k_f} \left(\frac{\partial^+ E}{\partial \mathbf{v}_j} \frac{\partial \mathbf{v}_j}{\partial \mathbf{v}_k} + \frac{\partial^+ E}{\partial \mathbf{W}_j} \frac{\partial \mathbf{W}_j}{\partial \mathbf{v}_k} \right) \tag{3.32}$$

Les composantes de la matrice $\partial \mathbf{W}_j/\partial \mathbf{v}_k$ sont nulles. Les termes $\partial \mathbf{v}_j/\partial \mathbf{v}_k$ sont non nuls dès lors que $j \leq k + D$. $\partial^+ E/\mathbf{v}_k$ s'écrit

$$\frac{\partial^+ E}{\partial \mathbf{v}_k} = \mathbf{e}_k^T + \sum_{j=k+1}^{k_f} \frac{\partial^+ E}{\partial \mathbf{v}_j} (I - \frac{\partial f_j}{\partial \mathbf{v}_j})^{-1} \frac{\partial f_j}{\partial \mathbf{v}_k} \tag{3.33}$$

Les termes $\partial f_j/\partial \mathbf{v}_k$ sont non nuls pour $k \leq j \leq k+D$. En substituant (3.20) dans (3.33), on obtient finalement

$$\frac{\partial^+ E}{\partial \mathbf{v}_k} = \mathbf{e}_k^T + \sum_{j=1}^{D} \frac{\partial^+ E}{\partial \mathbf{v}_{k+j}} [\mathbf{I} - \mathbf{G'}_{k+j} \mathbf{W}^0_{k+j}{}^T]^{-1} \mathbf{G'}_{k+j} \mathbf{W}_{k+j}{}^T \tag{3.34}$$

(3.34) est une équation aux différence inversée dans le temps qui se résout grâce aux conditions aux limites suivantes

$$\frac{\partial^+ E}{\partial \mathbf{v}_{k_f}} = \mathbf{e}^T_{k_f} \tag{3.35}$$

$$\frac{\partial^+ E}{\partial \mathbf{v}_{k_f+1}}, \ldots, \frac{\partial^+ E}{\partial \mathbf{v}_{k_f+D}} = \mathbf{0} \tag{3.36}$$

Définissons

$$\mathbf{y}_k^T = \frac{\partial^+ E}{\partial \mathbf{v}_k} [\mathbf{I} - \mathbf{G'}_k \mathbf{W}_k^{0\,T}]^{-1} \tag{3.37}$$

La substitution (3.37) dans (3.34) permet de calculer $\mathbf{y}_k$ en itérant l'équation

$$\mathbf{y}_k^T = [\mathbf{e}_k^T + \sum_{j=1}^{D} \mathbf{y}_{k+j}^T \mathbf{G'}_{k+j} \mathbf{W}^j_{k+j}{}^T][\mathbf{I} - \mathbf{G'}_k \mathbf{W}_k^{0\,T}]^{-1} \tag{3.38}$$

La transposition de (3.38) donne

$$\mathbf{y}_k = [\mathbf{I} - \mathbf{W}_k^0 \mathbf{G'}_k]^{-1} [\mathbf{e}_k + \sum_{j=1}^{D} \mathbf{W}^j_{k+j} \mathbf{G'}_{k+j} \mathbf{y}_{k+j}] \tag{3.39}$$

Rappelons que

$$\begin{aligned} \mathbf{W}_k^d &= [\mathbf{w}_k^d(1), \mathbf{w}_k^d(2), \ldots, \mathbf{w}_k^d(N)] \\ \mathbf{w}_k^d(j) &= [w_k^d(1,j), \ldots, w_k^d(N,j)]^T \end{aligned}$$

Définissons la matrice $N \times N$

$$\mathbf{V}_{k-d}(j) = [\mathbf{0}_N, \ldots, \mathbf{0}_N, \mathbf{v}_{k-d}, \mathbf{0}_N, \ldots, \mathbf{0}_N]^T \tag{3.40}$$

où $\mathbf{v}_{k-d}$ est placé à la j-ième ligne de la matrice et $\mathbf{0}_N$ désigne un vecteur nul de taille N. La différentiation de (3.4) donne

$$\frac{\partial \mathbf{v}_k}{\partial \mathbf{w}_k^d(j)} = [\mathbf{I} - \mathbf{G'}_k \mathbf{W}_k^{0\,T}]^{-1} \mathbf{G'}_k \mathbf{V}_{k-d}(j) \tag{3.41}$$

D'après (3.31), (3.39) et (3.41), il vient

$$\begin{aligned}\Delta \mathbf{w}^d(j) &= \eta \sum_{k=d}^{k_f} \frac{\partial^+ E}{\partial \mathbf{v}_k} [\mathbf{I} - \mathbf{G'}_k \mathbf{W}_k^{0^T}]^{-1} \mathbf{G'}_k \mathbf{V}_{k-d}(j) \\ &= \eta \sum_{k=d}^{k_f} g'_k(j) \mathbf{y}_k^T \mathbf{V}_{k-d}(j) \\ &= \eta \sum_{k=d}^{k_f} g'_k(j) y_k(j) \mathbf{v}_{k-d}^T \end{aligned} \tag{3.42}$$

(3.43)

Finalement, on obtient une généralisation vectorielle de la règle delta pour le DRNN

$$\Delta \mathbf{W}^d = \eta \sum_{k=d}^{k_f} \boldsymbol{\delta}_k \mathbf{v}_{k-d}^T \tag{3.44}$$

où le vecteur $\boldsymbol{\delta}_k$ est donné par

$$\boldsymbol{\delta}_k = \mathbf{G'}_k \mathbf{y}_k \tag{3.45}$$

La procédure à laquelle nous arrivons englobe la rétro-propagation temporelle pour les réseaux FIR non-bouclés [Wan 93b], la rétro-propagation récurrente pour des réseaux statiques récurrents [Almeida 87, Pineda 87], et la rétro-propagation dans le temps pour des réseaux à délais [Rumelhart 86]. On peut concevoir cet algorithme comme la rétro-propagation récurrente appliquée à un réseau *adjoint* [Baldi 95, Srinivasan 94] ou *transposé* [Hertz 91]. Un modèle DRNN simple et son adjoint sont décrits Fig. 3.2. L'adjoint est construit suivant les règles suivantes :

- inverser les flots de tous les signaux de telle façon que les sorties se substituent aux entrées.
- redéfinir les neurones comme des opérateurs affines en utilisant la dérivée du point fixe (3.4).
- inverser le temps de manière à ce que l' itération k devienne l'itération $k_f - k$ pour chaque variable, où k_f est l'itération courante.

Comme nous venons de le préciser, l'algorithme à propagation récurrente arrière nécessite l'artifice du dépliage de l'adjoint dans le temps. La réplication des neurones et des connexions est effectuée jusqu'à ce que tous les délais aient disparus. La méthode produit un réseau *time*-feedforward et *space*-recurrent, c'est-à-dire une architecture admettant des connexions bouclées mais non retardées uniquement. En conséquence, la règle ∂ généralisée (3.44) requiert le stockage des entrées internes $\mathbf{s}_k$ et des activations $\mathbf{v}_k$ au fur et à mesure des itérations. Le calcul du gradient est mené récursivement dans le temps en utilisant les valeurs stockées. Toutefois, pour que le réseau déplié soit strictement équivalent au réseau original, la démultiplication du réseau adjoint doit être répétée *ad infinitum*, c'est-à-dire jusqu'à la première itération ce qui est rédhibitoire du point de vue de l'implémentation. La procédure a semble-t-il hérité des faiblesses de BPTT.

Or nous ne pouvons grever la procédure d'apprentissage d'opérations lourdes et coûteuses. Nous allons voir comment contourner ce problème épineux tout en maintenant le coût de l'algorithme à un niveau minimal.

3.5.1 La rétro-propagation temporelle récurrente

Etant donné Eq. 3.39, $\mathbf{y}_k$ peut être obtenu en itérant une équation linéaire. Cela nécessite cependant l'inversion de la matrice $\mathbf{I} - \mathbf{W}_k^0\mathbf{G'}_k$ à chaque itération k. Pour se débarrasser de cette contrainte, Eq. 3.39 peut être résolue en relaxant le réseau adjoint de manière analogue à (3.2), par

$$\mathbf{T}\frac{d\mathbf{y}_k(t)}{dt} = [\mathbf{W}_k^0\mathbf{G'}_k - \mathbf{I}]\mathbf{y}_k(t) + \mathbf{e}_k(t) + \sum_{n=0}^{D} \mathbf{W}_{k+n}^n\mathbf{G'}_{k+n}\mathbf{y}_{k+n} \tag{3.46}$$

Les opérations non locales sont bannies de la procédure (i.e. l'inversion matricielle) afin de respecter la topologie du réseau. Le point fixe est donné par (3.39). D'après (3.46), l'apprentissage du DRNN revient à adapter les sous-réseaux bouclés statiques qui composent les couches du réseau déplié en utilisant le rétro-propagation récurrente standard [Almeida 87, Pineda 87]. Il est intéressant de remarquer que la stabilité de (3.46) découle de la stabilité de (3.4) que nous avons posée en postulat. Ce n'est pas surprenant car Almeida a déjà observé que la rétro-propagation dans un réseau statique bouclé impliquant une opération de relaxation, converge nécessairement lorsque le réseau est stable [Almeida 87]. Cette assertion est confirmée dans le contexte du DRNN puisque la linéarisation de (3.2) au voisinage d'un attracteur implique que les valeur propres de la matrice $\mathbf{T}^{-1}(\mathbf{I} - \mathbf{G'}_k\mathbf{W}_k^0)^T$ vérifient $\mathrm{Re}(\lambda_k) > 0$. Comme $\mathbf{T}^{-1}(\mathbf{I} - \mathbf{W}_k^0\mathbf{G'}_k)$ et $\mathbf{T}^{-1}(\mathbf{I} - \mathbf{G'}_k\mathbf{W}_k^{0^T})$ ont les même valeurs propres, la stabilité de (3.46) autour d'un point fixe est assurée.

La solution stable de Eq. 3.39 peut s'écrire d'une manière plus concise

$$\left\{\begin{array}{lcl} \mathbf{y}_k & = & [\mathbf{I} - \mathbf{W}_k^0\mathbf{G'}_k]^{-1}\mathbf{e}_k \\ \mathbf{e}_k & = & \mathbf{e}_k + \sum_{n=1}^{D} \mathbf{W}_{k+n}^d\mathbf{G'}_{k+n}\mathbf{y}_{k+n} \end{array}\right. \tag{3.47}$$

Eq. 3.47 est intéressante puisqu'elle exprime $\mathbf{y}_k$ comme la somme de deux termes : le premier représente l'erreur courante, $\mathbf{e}_k$, quant au second, il correspond à une erreur additionnelle qui est, en essence, l'erreur équivalente transposée à l'instant présent, k, due à l'influence du vecteur d'état courant sur l'évolution future du réseau de neurones.

Cette représentation montre que $\mathbf{y}_k$, à l'instant k, est une combinaison linéaire de $\mathbf{e}_k$ et des erreurs futures $\mathbf{e}_{k+j}$ pour $j = 1, \ldots, \infty$. Désignons par $\mathbf{y}_k^{k+n}$ le vecteur d'erreur rétro-propagé à l'instant k lorsque tous les $\mathbf{e}_{k+j} = \mathbf{0}$ sauf le terme correspondant à $j = n$. Rappelons que $E_n = \mathbf{e}_n^T\mathbf{e}_n$ est l'erreur quadratique à l'instant n. Etant donné (3.37), nous pouvons écrire

$$\mathbf{y}_k^{k+n\,T} = \frac{\partial E_{k+n}}{\partial \mathbf{v}_k}[\mathbf{I} - \mathbf{G'}_k \mathbf{W}_k^{0\,T}]^{-1} \tag{3.48}$$

De par la linéarité de E en terme des E_n, il vient

$$\mathbf{y}_k = \sum_{n=0}^{k_f - k} \mathbf{y}_k^{k+n} \tag{3.49}$$

Maintenant, définissons

$$\boldsymbol{\delta}_k^{k+n} = \mathbf{G'}_k \mathbf{y}_k^{k+n} \tag{3.50}$$

La règle δ (3.44) peut être reformulée en incorporant les termes $\mathbf{y}_k^{k+n}$ et $\boldsymbol{\delta}_k^{k+n}$

$$\begin{aligned} \Delta \mathbf{W}^d &= \eta \sum_{k=d}^{k_f} \boldsymbol{\delta}_k \mathbf{v}_{k-d}^T \\ &= \eta \sum_{k=d}^{k_f} \sum_{n=0}^{k_f - k} \mathbf{G'}_k \mathbf{y}_k^{k+n} \mathbf{v}_{k-d}^T \\ &= \eta \sum_{k=d}^{k_f} \sum_{n=0}^{k_f - k} \boldsymbol{\delta}_k^{k+n} \mathbf{v}_{k-d}^T \\ &= \eta \sum_{n=d}^{k_f} \sum_{k=d}^{n} \boldsymbol{\delta}_k^{n} \mathbf{v}_{k-d}^T \end{aligned} \tag{3.51}$$

A un instant donné n, les termes $\mathbf{y}_k^n$, $k < n$ sont rétro-propagés à travers le réseau adjoint jusqu'à l'instant 0. Les gradients sont recombinés par la suite pour adapter la matrice de poids en fonction de l'erreur $\mathbf{e}_n$.

Le dépliage du réseau adjoint s'opère à la fin de l'itération n pour permettre le calcul des termes $\boldsymbol{\delta}_k^n$ pour $k = 0, \ldots, n$. Comme nous l'avons déjà souligné, la mémoire requise augmente au fur et à mesure que de nouveaux exemples sont présentés au réseau. C'est, du reste, le revers majeur de cette méthode.

Dans l'optique de prévenir l'expansion systématique du réseau de propagation des erreurs, nous discutons dans le chapitre suivant de la dynamique de $\boldsymbol{\delta}_k^n$ lorsque k varie. Cette étude répond aux problèmes de stabilité que soulève la procédure d'adaptation des poids.

La stabilité de la procédure d'adaptation des poids

La stabilité asymptotique de l'adaptation de la matrice de poids lorsque $k_f \rightarrow \infty$ est indispensable pour assurer le bon fonctionnement du réseau et de son apprentissage. L'analyse du comportement du gradient rétro-propagé ouvrira la voie à la conception d'un algorithme à propagation arrière similaire au précédent mais qui ne souffre pas des inconvénients dont nous avons parlé.

En raison du comportement dynamique du DRNN, les conditions initiales du système à l'instant 0, c'est-à-dire $\mathbf{v}_0$ et $\boldsymbol{\mathcal{W}}_0$, ainsi que la séquence des vecteurs d'entrée $\mathbf{x}_k$ déterminent complètement la trajectoire du réseau neuronal. Par conséquent, le moindre changement dans ces conditions initiales influence l'évolution future du système. Typiquement, les ajustements individuels des poids des connexions dupliquées dans le réseau déplié doivent être recombinés au terme de la rétro-propagation à l'instant final k_f, soit

$$\Delta \mathbf{W}^d = \sum_{k=d}^{k_f} \Delta \mathbf{W}_k^d \tag{3.52}$$

L'interrogation légitime que soulève cette formule a trait à la convergence de la somme lorsque $k_f \to \infty$. On se heurte malencontreusement à l'influence des entrées, indépendantes du système lui-même, sur le résultat de cette somme. En pratique les poids peuvent diverger, du moins atteindre les limites de saturation qui leurs sont imposées.

En fait, il semble qu'il n'y ait pas de réponse formelle à ce problème général. Mais comme nous nous intéressons à une adaptation en temps réel "en-ligne", ce n'est pas tant la stabilité *globale* de l'ajustement des poids durant l'époque $[0, k_f]$ qui nous préoccupe, mais sa convergence à chaque itération. Et cette convergence n'est *a priori* pas garantie puisque le nombre de couches du réseau adjoint déplié est égal à k_f. Aussi, l'adaptation en ligne peut pâtir du manque de stabilité de la rétro-propagation dans l'adjoint. Des conditions nécessaires doivent être établies pour garantir l'existence de la convergence de (3.52) à chaque instant. Il convient d'emblée de traduire analytiquement la notion de stabilité.

Définition 1 - *La procédure d'adaptation des poids dans le DRNN est dite* stable *lorsque*

$$\lim_{n \to \infty} \| \sum_{k=d}^{n} \boldsymbol{\delta}_k^n \mathbf{v}_{k-d}^T \| < \infty \tag{3.53}$$

Non seulement $\boldsymbol{\delta}_{n-j}^n$ doit tendre vers 0 lorsque $j \to n$ mais sa vitesse de convergence doit être exhibée pour permettre de conclure sur la convergence de la somme (3.53). Cette convergence fait l'objet du théorème suivant

Théorème 1 - *Considérons un réseau de neurones à relaxation opérant en temps discret dont le point fixe à chaque instant* k *est défini par les équations récurrentes Eq. 3.4. Soit* N *et* D *le nombre de neurones et l'ordre filtre AR,* $\mathbf{x}_k$ *le vecteur des entrées exogènes,* $\mathbf{v}_k$ *le vecteur d'état, et* $\mathbf{d}_k$ *le vecteur des sorties désirées* $(\mathbf{x}_k, \mathbf{v}_k, \mathbf{d}_k \in R^{[N\times 1]})$.

Le coût instantané est donné par le scalaire $E_n = \frac{1}{2}(\mathbf{d}_n - \mathbf{v}_n)^T(\mathbf{d}_n - \mathbf{v}_n)$, *et le vecteur d'erreur par* $\mathbf{y}_k^{k+n^T} = \frac{\partial E_{k+n}}{\partial \mathbf{v}_k}[\mathbf{I} - \mathbf{G'}_k {\mathbf{W}_k^0}^T]^{-1}$. $\mathbf{W}_k^d$ *représente la matrice de poids associée aux connexions de délai* d, *et* $\mathbf{G'}_k$ *ést définie en* (3.21).

Soit $k_f > 0$ *la durée de la trajectoire,* **si**

$$\|\mathbf{W}_k^0\| < 1 \quad \text{et} \quad \frac{\|\mathbf{W}_k^d\|}{1 - \|\mathbf{W}_k^0\|} < \frac{1}{D} \tag{3.54}$$

est satisfait $\forall d = 1, \ldots, D$, $\forall k = 0, \ldots, k_f$ **alors** $\exists \rho, 0 < \rho < 1$ *et* $\exists M > 0$ *tels que*

$$\|\mathbf{y}_{k_f-n}^{k_f}\| < M\rho^n, \tag{3.55}$$

pour tout $n = 0, 1, \ldots, k_f$.

Démonstration : Voir l'appendice à la page 79.

Remarquons que lorsque les poids ne sont modifiés qu'au terme de l'époque, les conditions (3.54) se réduisent à

$$\|\mathbf{W}^0\| < 1 \quad \text{et} \quad \frac{\|\mathbf{W}^d\|}{1 - \|\mathbf{W}^0\|} < \frac{1}{D} \quad \forall d = 1, \ldots, D \tag{3.56}$$

Le théorème dit en essence, que les termes $\mathbf{y}_{k_f-n}^{k_f}$ sont majoreś par une suite décroissante à vitesse exponentielle par rapport à n si (3.54) est vérifiée. Les simulations numériques conduites par l'auteur suggèrent fortement que la descente du gradient est stable pour une grande majorité des configurations même si les conditions (3.54) ne sont pas remplies. Le DRNN semble avoir une propension "naturelle" à la stabilité et la décroissance observée au cours des applications aux Chapitres 4 et 5 témoigne du bien fondé de nos résultats. C'est pourquoi nous parlerons de la décroissance exponentielle du gradient rétro-propagé dans le temps à travers le réseau adjoint déplié. Le terme exponentiel est à prendre au sens large puisque la décroissance n'es pas strictement monotone, des oscillations sont également observées. Une analyse exhaustive est menée dans l'appendice (page 79) et les résultats auxquels nous parvenons seront validés dans le chapitre concernant l'application du modèle à la prédiction des séries temporelles.

Au regard du Théorème 1, nous pouvons tirer bon augure de la rapide décroissance du gradient. Une série de propositions en découle.

Proposition 1 *Si les conditions* (3.54) *sont satisfaites, la stabilité de la procédure d'adaptation est assurée.*

Démonstration : Soit n l'itération courante. D'après (3.51) et (3.21), il vient

$$\begin{aligned}
\sum_{k=d}^{n} \boldsymbol{\delta}_k^n \mathbf{v}_{k-d}^T &\leq \sum_{k=d}^{n} \|\mathbf{G'}_k\| \|\mathbf{y}_k^n\| \|\mathbf{v}_{k-d}\| \\
&\leq 2\beta \sum_{k=d}^{n} \|\mathbf{G}_k\| \|\mathbf{I} - \mathbf{G}_k\| \|\mathbf{y}_k^n\| \|\mathbf{v}_{k-d}\| \\
&\leq \sum_{k=d}^{n} \|\mathbf{y}_k^n\| \|\mathbf{v}_{k-d}\| \\
&\leq N \sum_{k=d}^{n} \|\mathbf{y}_k^n\|
\end{aligned} \tag{3.57}$$

car $\beta = 1/2$ et $\|\mathbf{v}_{k-d}\|_p < N$ pour $p = 1, 2, \infty$. En accord avec le Théorème 1, on peut trouver $\rho < 1$ et $M > 0$ tels que

$$\sum_{k=d}^{n} \|\mathbf{y}_k^n\| \leq M \sum_{k=0}^{n} \rho^k \leq \frac{M}{1-\rho} \tag{3.58}$$

cqfd.

Le critère d'arrêt du dépliage

Sous les conditions de stabilité, tout ce que nous pouvons strictement affirmer à la lumière du Théorème 1, est la décroissance du gradient d'erreur vers l'origine. Mais ce que nous cherchons à connaître plus spécifiquement est le nombre de rétro-propagations requisent par la procédure pour maintenir une erreur négligeable dans l'évaluation du gradient d'erreur. Avec cet objectif à l'esprit, nous introduisons deux propositions utiles pour définir un critère d'arrêt *ad hoc* qui soit peu demandeur de ressources et tout à la fois efficace.

Proposition 2 *Si les conditions* (3.54) *sont vérifiées, alors quelque soit* ϵ*, si* $\|\mathbf{y}_{k+j}^{k_f}\| \leq \epsilon$ $(k < k_f - D)$ *est vérifié pour* $j = 1, \ldots, D$*, alors* $\|\mathbf{y}_{k-n}^{k_f}\| \leq \epsilon \;\; \forall n = 0, 1, \ldots, k$.

Démonstration : De $\|\mathbf{W}^0\| < 1$, il est démontré dans l'appendice à la page 79 que

$$\|\mathbf{y}_k^{k_f}\| \leq \frac{1}{1 - \|\mathbf{W}_k^0\|} \sum_{d=1}^{D} \|\mathbf{W}_{k+d}^d\| \|\mathbf{y}_{k+d}^{k_f}\| \tag{3.59}$$

De la seconde inégalité du Théorème 1, il vient que $\|\mathbf{y}_k^{k_f}\| \leq \epsilon \sum_{j=1}^{D} 1/D = \epsilon$. Par récurrence, on a $\|\mathbf{y}_{k-n}^{k_f}\| \leq \epsilon$ pour $n = 1, \ldots, k$, cqfd.

Proposition 3 *Considérons l'algorithme à propagation avant opérant après chaque exemple présenté, si les conditions* (3.56) *sont vérifiées, alors pour que l'approximation du gradient exacte soit aussi précise que désirée, il suffit de rétro-propager le gradient un nombre limité de fois dans le réseau adjoint.*

Démonstration : Soit n l'itération courante et supposons que l'adjoint soit déplié s fois. L'estimation de la matrice de poids est donnée par $\sum_{k=n-s}^{n} \boldsymbol{\delta}_k^n \mathbf{v}_{k-d}^T$, il résulte une erreur de calcul de $\sum_{k=d}^{n-s-1} \boldsymbol{\delta}_k^n \mathbf{v}_{k-d}^T$.

Au vu du Théorème 1 et de (3.57), il vient

$$\sum_{k=d}^{n-s-1} \|\boldsymbol{\delta}_k^n \mathbf{v}_{k-d}^T\| \leq \frac{N M \rho^{s+1}}{1-\rho} \tag{3.60}$$

Le terme de gauche tend exponentiellement vers zéro avec s, cqfd.

Proposition 3 suggère qu'une procédure peut être employée pour déterminer le nombre de rétro-propagations nécessaires à la fin de chaque itération si M et ρ sont déterminés au regard de la décroissance. Dans ce cas, l'expression à gauche dans (3.60) peut être convenablement majorée par une constante arbitraire s.

Une seconde approche, certes moins rigoureuse, mais qui répond mieux aux exigences de performance et de facilité d'exécution, est proposée. Selon la Proposition 2, il est suffisant, pour une bonne approximation du gradient à l'instant k, de rétro-propager un nombre de fois s tel que

$$\frac{\|\mathbf{y}_{k-s+j}^{k}\|}{\|\mathbf{y}_{k}^{k}\|} \leq \epsilon \quad \text{pour} \quad j = 1, \ldots, D \tag{3.61}$$

ϵ est arbitrairement fixé à 10^{-3} dans notre implémentation. Ce critère garantit une approximation aussi précise que voulue du gradient exacte. On peut louer les bienfaits de la descente exponentielle du gradient puisque celui-ci peut être tronqué sans encourir le risque d'une mauvaise estimation.

En ce qui concerne la mémoire requise, s à l'itération k dépend de $\mathbf{e}_k$ ce qui rend sont anticipation difficile. Or, il importe de déterminer le volume de mémoire qu'il faut allouer pour s'assurer de la bonne excecution du programme. La variable s varie erratiquement au fil des itérations, c'est pourquoi il est bon de réserver beaucoup d'espace mémoire en début d'apprentissage et de le réduire progressivement. Les simulations présentées dans le chapitre suivant illustrerons la variation de s au fil des itérations.

L'algorithme TRBP

A la lumière de la précédente discussion, nous décrivons un algorithme qui tire avantage de la décroissance exponentielle de $\boldsymbol{\delta}_k^n$ lorsque $n - k \to \infty$. Nous appelons cet algorithme incrémental, le "Temporal Recurrent Back-Propagation" (TRBP), c'est-à-dire la rétro-propagation temporelle récurrente.

L'algorithme à propagation récurrente arrière a été établi en supposant que les poids restent fixes au cours d'une époque. Puisque le gradient global est une combinaison linéaire des gradients locaux calculés à chaque itération en fonction de l'erreur courante, nous pouvons nous soustraire à cette contrainte et adapter les poids a l'issue de chaque itération. Les bénéfices de cette approche sont que i) la complexité spatiale ne dépend plus du nombre d'exemples, et ii) l'algorithme opère désormais en temps réel.

En contre partie, le calcul de $\partial E/\partial w$ n'est plus strictement exacte à cause de la variation des poids entre deux itérations. Il suffit de prendre η assez petit de sorte que $\|\Delta \mathbf{W}^d\|$ soit borné sur les trajectoires de longueur s. Cela permet de garder l'altération du gradient sous contrôle. De par la proposition 3, nous savons qu'un nombre limité de rétro-propagations suffit à calculer ce qui est nécessaire pour fournir une estimation du gradient aussi fidèle que possible.

L'implémentation d'une procédure opérant en temps réel repose sur le principe d'introduire des époques artificielles. Une époque artificielle est créée en supposant que l'itération courante est $k = k_f$. La première itération n'est plus 0 mais $k - s$ où s est déterminé par le critère d'arrêt. Cette procédure s'inscrit donc dans la continuité de l'étude précédente.

L'algorithme complet est implémenté en suivant, à chaque itération k, les étapes suivantes

- Relaxer une fois le DRNN jusqu'à l'obtention d'un point fixe, en utilisant les vecteur d'entrée $\mathbf{x}_k$ et toutes les vecteurs d'état remontant à $k - D$ suivant les équations

$$\mathbf{v}_k = f(\mathbf{x}_k, \ldots, \mathbf{x}_{k-D}, \mathbf{v}_k, \mathbf{v}_{k-1}, \ldots \mathbf{v}_{k-D}, \boldsymbol{\mathcal{W}}_k) \tag{3.62}$$

- Déterminer une approximation pour $\mathbf{y}^k_{k-n}$, désignée par $\hat{\mathbf{y}}^k_{k-n}$, pour $n > 0$ en dépliant le réseau adjoint et le relaxant suivant les équations

$$\mathbf{T}\frac{d\hat{\mathbf{y}}^k_{k-n}(t)}{dt} = [\mathbf{W}^0_k\mathbf{G'}_{k-n} - \mathbf{I}]\hat{\mathbf{y}}^k_{k-n}(t) + \sum_{n=1}^{D} \mathbf{W}^n_k\mathbf{G'}_{k-n+j}\hat{\mathbf{y}}^k_{k-n+j} \tag{3.63}$$

  Les conditions aux limites de (3.63) prennent la forme

$$\hat{\mathbf{y}}^k_k = \mathbf{y}^k_k = [\mathbf{I} - \mathbf{W}^0_k\mathbf{G'}_k]^{-1}\mathbf{e}_k \tag{3.64}$$

  et

$$\hat{\mathbf{y}}^k_{k+1}, \ldots, \hat{\mathbf{y}}^k_{k+D} = \mathbf{0} \tag{3.65}$$

  Remarquons que $\mathbf{y}^k_{k-n}$ ne peut être déterminé exactement car la matrice de poids $\mathbf{W}^j_{k-n+j}$ est approximée par la matrice courante $\mathbf{W}^j_k$. Clairement, plus la variation des poids est faible entre l'itération k et $k-n+j$ et plus précise sera l'approximation des termes $\hat{\mathbf{y}}^k_{k-n}$.

  Localement dans le réseau adjoint, les connexions sont inversées; plus précisément, le couplage w^d_{ij} entre les neurones i et j est substitué par $g'(s_j)w^d_{ji}$ du neurone j vers le neurone i. Le vecteur d'erreur est pris comme l'entrée courante et les fonction de transfert accomplissent une transformation affine.

- Stopper la récursion temporelle de (3.63) à l'itération $k - s$ satisfaisant

$$\frac{\|\mathbf{y}^k_{k-s+j}\|}{\|\mathbf{y}^k_k\|} \leq \epsilon \quad \text{pour} \quad j = 1, \ldots, D \tag{3.66}$$

  Les termes $\mathbf{y}^{k-j}_k$ sont mémorisés à chaque itération jusqu'à la fin du processus itératif.

- L'adaptation globale des poids, définie comme la somme des variations individuelles, est accomplie en appliquant la règle δ généralisée

$$\Delta\mathbf{W}^d = \eta\sum_{n=d}^{k} \boldsymbol{\delta}^k_n\mathbf{v}^T_{n-d} \tag{3.67}$$

- Les termes $g'(\mathbf{s}_{k-j})$ et $\mathbf{v}_{k-j}$ sont mémorisés pour $j = 0, 1, \ldots, L$; L est choisi assez grand pour vérifier $s < L$ à chaque instant. En pratique, L est un multiple de D incrémenté chaque fois que $s > L$ jusqu'à ce qu'une valeur convenable soit trouvée.

Cet algorithme préserve la symétrie entre la propagation avant des signaux et la propagation arrière du gradient. La symétrie contribue grandement à l'efficacité de l'implémentation parallèle de la procédure. Fig. 3.3 et 3.4 illustrent le comportement type de l'apprentissage des équations de Lorenz au gré des itérations.

Finalement, la prééminence de TRBP sur la rétro-propagation récurrente en arrière trouve son origine dans l'ajustement optimal du nombre de passes s, dans l'adjoint afin de réduire le volume d'opérations nécessaires. Les qualités d'une procédure de cet acabit sont patentes car les limitations des algorithmes à propagation arrière, tel que BPTT, dues à un nombre de passes fixes à travers l'adjoint sont levées sans porter préjudice à sa complexité.

L'apprentissage forcé - Il existe une variante intéressante à l'algorithme présenté au dessus qui semble être utile pour le traitement temporel. L'idée est de remplacer $\mathbf{v}_{k-j}$ par ses valeurs désirées $\mathbf{d}_{k-j}$ si ces dernières sont connues. Cette modification est appelée l'adaptation *équation error*, ou plus simplement le *teacher forcing* [Hertz 91, Williams 89]. L'objectif est de rapprocher le réseau au plus près de sa trajectoire idéale et d'accélérer l'apprentissage.

Parce que l'activité future du réseau est basée sur les états forcés du réseau plutôt que sur les états libres, il faut annuler les termes $y_{k-j}(i)$ pour $j > 0$ lorsque d_i est connu. Mais ce n'est pas tout; si $\exists w_{ij}^0 \neq 0$ où i est un neurone de sortie et j un neurone caché, alors il faut relaxer à nouveau le réseau en posant $V_i = d_i$ pour être conséquent avec le teacher forcing. Dans un scénario pratique, cette légère complication peut être omise toutefois.

Signalons que l'adaptation teacher forcing altère la dynamique libre du système d'une manière qualitative. De nouveaux attracteurs peuvent apparaître ou disparaître après la levée des contraintes [Hertz 91, Wan 93a]. En outre, le teacher forcing n'a pas de raison d'être, lorsque les valeurs passées de la séquence sont présentées en entrée du réseau.

Les ressources requises - Puisque s est de l'ordre D, la complexité spatiale est en $O(DN)$, en contraste à $O(DN^3)$ avec la propagation récurrente avant, pour mémoriser les précédents vecteurs d'état. Au regard de (3.63), on observe que la relaxation du réseau requiert $O(DN^2)$ opérations de sorte que $O(D^2N^2)$ opérations sont nécessaires, au lieu de $O(D^2N^4)$, à chaque itération pour ajuster les paramètres du réseau. La gestion parcimonieuse des ressources est payante pour de grandes valeurs de N (voir Table 3.5.1). Enfin, une inversion matricielle n'est plus nécessaire pour calculer le gradient. Dans ces circonstances, tout concourt à favoriser l'emploi de l'algorithme TRBP. Rappelons que le DRNN adapté avec l'algorithme TRBP généralise la plupart des architectures et procédures d'apprentissages à descente du gradient répertoriées à ce jour.

3.5.2 L'évanouissement du gradient d'erreur dans l'adjoint

Sans réfuter les conclusions auxquelles nous sommes arrivés, l'examen approfondi de la décroissance rapide du gradient dans l'adjoint soulève le problème de l'apprentissage à long terme. On entend par *apprentissage* au sens large, l'habileté à inférer les régularités et à mémoriser des événements passés de la séquence d'entrée influençant les sorties subséquentes. Pour qu'un système dynamique tel que le DRNN soit capable de mémoriser de l'information sur une durée arbitraire, Bengio et al. [Bengio 94] ont établi deux conditions élémentaires :

- que le système soit résistant au bruit (i.e. fluctuations aléatoires des entrées, quantification des paramètres, etc.).
- que le système converge en un temps raisonnable.

Le gradient doit être propagé le temps que dure la mémorisation de l'information jusqu'à son utilisation. Plaçons nous dans la cas où le réseau est stable au sens défini en (3.53); nous avons alors $\|\mathbf{y}_k^n\| \rightarrow 0$ lorsque $k \rightarrow 0$. Etant donné (3.48), il vient

$$\frac{\partial E_n}{\partial \mathbf{v}_k} = \mathbf{y}_k^{n\,T}[\mathbf{I} - \mathbf{G'}_k \mathbf{W}_k^{0\,T}] \tag{3.68}$$

D'où $\|\partial E_n / \partial \mathbf{v}_k\| \rightarrow 0$ lorsque $k \rightarrow 0$. Bien qu'il puisse exister un changement adéquat de la matrice de poids qui permette le transfert d'information sur une longue période de temps, le gradient ne reflète pas cette possibilité. L'apprentissage devient difficile en raison de l'influence des dépendances temporelles à court terme qui dominent dans le calcul du gradient. En conséquence, la stabilité de la procédure d'adaptation pénalise l'apprentissage à long terme. Une raison intuitive est la suivante : un réseau peut être divisé en deux régions A et B, A étant dédié à l'extraction, et B au stockage de l'information. Supposons qu'une solution au problème consiste à stocker dans B l'information que A aura détecté à l'instant 0. La sortie de B à un instant T sera utilisée ultérieurement en vue de calculer une erreur. Si B n'a pas été entraîné assez longtemps pour être capable de mémoriser l'information durant T itérations, alors le gradient d'erreur calculé à T par rapport à la sortie de A à l'instant 0, sera extrêmement faible parce que la sortie de A n'aura quasiment aucune influence sur l'erreur à l'instant T. A l'opposé, si B a été adapté de façon à stocker l'information, le vrai gradient peut être propagé; mais l'évanouissement du gradient dans le réseau adjoint nuit à l'adaptation de A. La valeur de T et le niveau de bruit sur cette plage temporelle contribuent à accentuer le phénomène.

Maintenant, si les lois d'adaptation sont instables, on a $\|\mathbf{y}_k^n\| \rightarrow \infty$ quand $k \rightarrow 0$, de sorte que la dérivée partielle $\|\partial E_n / \partial \mathbf{v}_k\| \rightarrow \infty$. Aussi, le réseau présente une dépendance exponentielle vis-à-vis des conditions initiales. Le bruit est considérablement amplifié conduisant à un comportement instable du réseau.

C'est la raison essentielle pour laquelle la descente du gradient est inadaptée pour capturer des dépendances à long terme. Une analyse fondée sur les attracteurs hyperboliques, conduit aux mêmes conclusions [Bengio 94]. Malgré ces limitations, l'attrait de la descente du gradient ne risque pas de succomber d'anémie. Les capacités prédictives du DRNN adapté avec TRBP sont clairement illustrées au Chapitre 4 dédié aux expérimentations sur les séries temporelles.

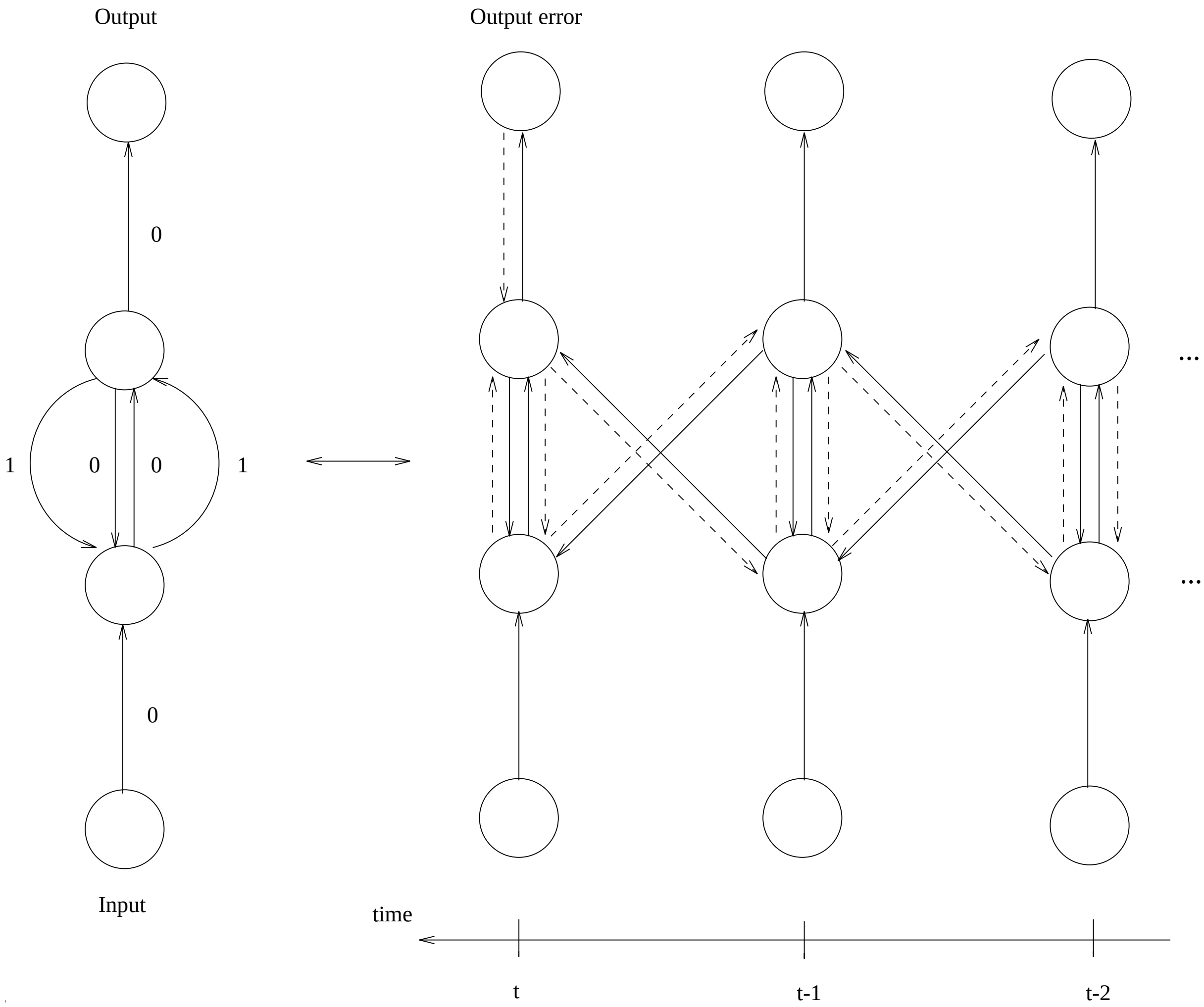

FIG. 3.2 - : Illustration du processus de dépliage pour un DRNN de 4 neurones. Les flèches en trait plein (resp. en pointillé) représentent les connexions du réseau original (resp. du réseau adjoint).

procédure \ complexité	temporelle	spatiale
TRBP	$O(D^2N^2)$	$O(DN)$
RTRL	$O(D^2N^4)$	$O(DN^3)$

TAB. 3.2 - : Complexité de la procédure TRBP, en contraste avec la complexité de la procédure à propagation avant qui est une version étendue de l'algorithme RTRL au DRNN [Williams 89].

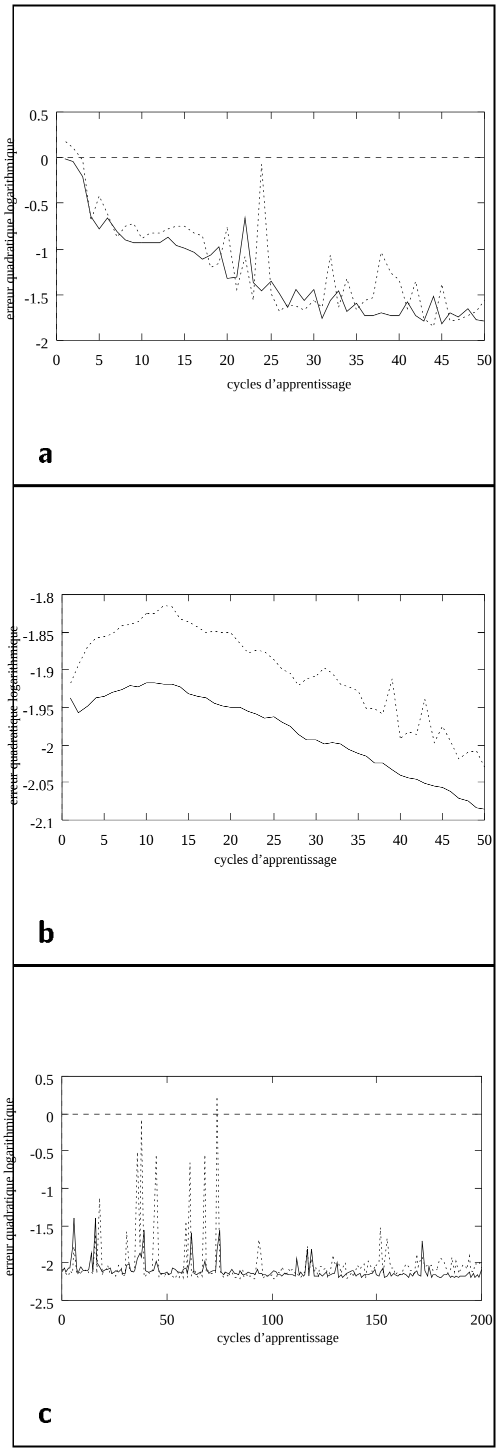

FIG. 3.3 - : Equations de Lorenz : Courbe d'apprentissage type pour un DRNN $1 \times 5 \times 2$ et d'ordre $10 : 1 : 0$ en utilisant l'algorithme TRBP avec un pas d'apprentissage a) $3 * 10^{-2}$, ensuite b) 10^{-2}, et un momentum de 0.9.

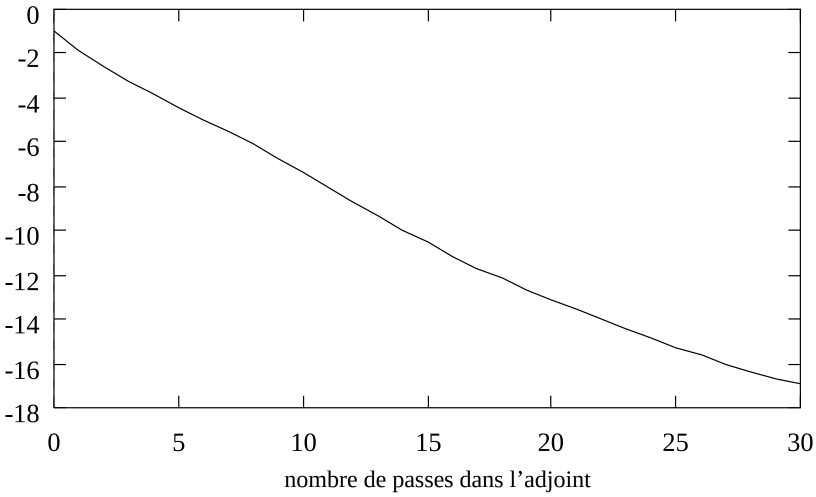

FIG. 3.4 - : Equations de Lorenz : $< \mathrm{Log}_{10}(\|\mathrm{y}^n_{n-j}\|_\infty >$, la moyenne du logarithme de la norme de y^n_{n-j} en fonction du nombre de rétro-propagations j dans le réseau adjoint.

3.6 Conclusion

Dans ce chapitre, nous avons exposé les fondements théoriques des réseaux de neurones récurrents et dynamiques. La modélisation des synapses par des filtres linéaires auto-régressifs est particulièrement bien adaptée pour préserver une mémoire de l'état du réseau. Le DRNN se prête bien au traitement temporel dans la mesure où la mémoire interne dont dispose le modèle, lui permet de trouver une représentation interne du vecteur d'état du système dynamique étudié. La présentation d'une fenêtre temporelle en entrée n'est plus nécessaire. Le DRNN est une architecture complètement générale qui englobe les architectures non-bouclées et semi-bouclées, à délai ou non.

L'accent a été mis sur la recherche d'un algorithme d'apprentissage de descente du gradient à faible coût. L'analyse de la convergence du gradient rétro-propagé à travers un réseau adjoint révèle une décroissance exponentielle sous certaines conditions relatives à la matrice de poids. Nous arguons qu'un nombre limité de passes à travers l'adjoint suffit à approximer le gradient d'erreur exact avec une précision arbitraire. Au reste, cette étude confirme l'inadéquation de la descente du gradient aux problèmes affichant de longue dépendances temporelles. Notre choix se porte sur une procédure à propagation récurrente arrière (TRBP) tirant profit de l'évanouissement du gradient dans l'adjoint. Le nombre d'opérations requises se limite au stricte minimum, et les opération non locales ou superflues sont bannies.

3.7 Appendice: Démonstration du Théorème

Dans cet appendice, nous évaluons le comportement asymptotique du gradient d'erreur propagé dans le réseau adjoint et démontrons le Théorème 1. Considérons le vecteur d'erreur rétro-propagé $\mathbf{y}_k^{k_f}$ $(0 \leq k \leq k_f)$, de dimension N défini à l'itération k lorsque un signal d'erreur $\mathbf{e}_{k_f} = \mathbf{d}_{k_f} - \mathbf{v}_{k_f}$ est appliqué au réseau adjoint à l'itération k_f. Posons $\mathbf{y}_k^{k_f} = 0$ pour $k > k_f$. Le comportement de $\mathbf{y}_k^{k_f}$ peut être déduit de l'équations récursive (3.39). Il suffit de poser $\mathbf{e}_j = 0$ pour $j = 0, 1, \ldots, k_f - 1$. Cela conduit à

$$\mathbf{y}_k^{k_f} = [\mathbf{I} - \mathbf{W}_k^0 \mathbf{G'}_k]^{-1} \sum_{d=1}^{D} \mathbf{W}_{k+d}^d \mathbf{G'}_{k+d} \mathbf{y}_{k+d}^{k_f} \tag{3.69}$$

pour $k < k_f - D$, avec les conditions aux limites suivantes

$$\mathbf{y}_{k_f}^{k_f} = [\mathbf{I} - \mathbf{W}_k^0 \mathbf{G'}_k]^{-1} \mathbf{e}_{k_f} \tag{3.70}$$

L'équation vectorielle (3.69) gouverne la dynamique des termes erreurs. La résolution de Eq. 3.70 se heurte à la dépendance temporelle des coefficient matriciels $\mathbf{G'}_k$. Quand bien même les poids demeurent fixes au cours d'une trajectoire $k = 0, \ldots, k_f$, $\mathbf{G'}_k$ dépend des entrées internes $\mathbf{s}_k$ qui, à leur tour, dépendent du vecteur d'entrée à l'instant k. Néanmoins, $\mathbf{G'}_k$ est par définition une matrice diagonale $N \times N$ construite avec $g'(s) = 2\beta g(s)(1-g(s))$ si bien que $g'(s) < 1$ avec $\beta = 1/2$. D'où $\|\mathbf{G'}_k\| < 1$. Par ailleurs, supposons que $\|\mathbf{W}_k^0\| < 1$ pour $k = 0. \ldots, k_f$ de sorte que la convergence de $\mathbf{I} + \mathbf{W}_k^0 + (\mathbf{W}_k^0)^2 + \ldots$ soit assurée [Wilkinson 65] (Notons que $\| \ \|$ désigne une norme matricielle *compatible* avec les normes vectorielles, i.e. $\|\mathbf{A}\|_p = \sup_{x \neq 0} \|\mathbf{A}x\|_p / \|x\|_p, p = 1, 2, \infty$). Par conséquent à l'itération $k = 0, \ldots, k_f - 1$, on a

$$\begin{aligned}
\|\mathbf{y}_k^{k_f}\| &\leq \|\mathbf{I} - \mathbf{W}_k^0 \mathbf{G'}_k\|^{-1} \sum_{d=1}^{D} \|\mathbf{W}_{k+d}^d\| \|\mathbf{G'}_{k+d}\| \|\mathbf{y}_{k+d}^{k_f}\| \\
&\leq \sum_{n=0}^{\infty} \|\mathbf{W}_k^0 \mathbf{G'}_k\|^n \sum_{d=1}^{D} \|\mathbf{W}_{k+d}^n\| \|\mathbf{G'}_{k+d}\| \|\mathbf{y}_{k+d}^{k_f}\| \\
&\leq \frac{1}{1 - \|\mathbf{W}_k^0\|} \sum_{d=1}^{D} \|\mathbf{W}_{k+d}^d\| \|\mathbf{y}_{k+d}^{k_f}\|
\end{aligned} \tag{3.71}$$

Soit

$$a_d = \mathrm{Max}_k \left(\frac{\|\mathbf{W}_{k+d}^d\|}{1 - \|\mathbf{W}_k^0\|} \right) \ 1 \leq d \leq D \tag{3.72}$$

D'après (3.69), (3.71) et (3.72), on déduit les inégalités suivantes

$$\|\mathbf{y}_k^{k_f}\| \leq a_1 \|\mathbf{y}_{k+1}^{k_f}\| + a_2 \|\mathbf{y}_{k+2}^{k_f}\| + \ldots + a_D \|\mathbf{y}_{k+D}^{k_f}\| \tag{3.73}$$

On obtient une équation récursive à coefficients constants. Cela permet l'analyse de la suite $\{u_n\}$, définie par

$$u_n = a_1 u_{n-1} + a_2 u_{n-2} + \ldots + a_D u_{n-D} \tag{3.74}$$

Le comportement de $\{u_n\}$ est instructif puisque nous avons $u_n \geq \|\mathbf{y}^{k_f}_{k_f-n}\|$ pour $n = 0, \ldots, k_f$, avec les conditions aux limites $u_n = \|\mathbf{y}^{k_f}_{k_f-n}\|$ pour $n = 0, \ldots, D-1$.

Le solution de Eq. 3.74 repose sur des opérations élémentaires d'algèbre linéaire. Ecrivons Eq. 3.74 sous une forme matricielle

$$\begin{bmatrix} u_n \\ u_{n-1} \\ u_{n-2} \\ \vdots \\ u_{n-D+1} \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 & \cdots & a_D \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} u_{n-1} \\ u_{n-2} \\ u_{n-3} \\ \vdots \\ u_{n-D} \end{bmatrix} \tag{3.75}$$

Pour résoudre ce système, on peut tirer bénéfice du pourcentage élevé de zéro dans la matrice de droite $\mathbf{A}$ pour déterminer le polynôme caractéristique d'ordre D, $P(\lambda) = \det[\mathbf{A} - \lambda\mathbf{I}]$, où $\mathbf{I}$ désigne la matrice identité $D \times D$. Si λ est une valeur propre de $\mathbf{A}$, et $\mathbf{v} = [v_1, \ldots, v_D]^T$ son vecteur propre correspondant, alors

$$\begin{cases} v_i = \lambda v_{i+1} \quad \text{pour} \quad i = 1, \ldots, D-1 \\ a_1 v_1 + a_2 v_2 + \ldots + a_D v_D = \lambda v_1 \end{cases} \tag{3.76}$$

Ces équations admettent une solution non-nulle si

$$P(\lambda) = a_D + a_{D-1}\lambda + \ldots + a_1 \lambda^{D-1} - \lambda^D = 0 \tag{3.77}$$

admet des solutions. Examinons le cas général où $P(\lambda) = 0$ possède D racines distinctes $\lambda = \lambda_k$ pour $k = 1, 2, \ldots, D$ (i.e. $\lambda_k \neq \lambda_j$ si $k \neq j$). u_n peut être représenté par une fonction des vecteurs propres

$$u_n = \sum_{j=1}^{D} b_j \lambda_j^n \tag{3.78}$$

où b_j sont des coefficients complexes constants déterminés par les conditions initiales $u_1, u_2, \ldots, u_D$. Puisque le polynôme $P(\lambda)$ est réel, les valeurs propres λ_k sont soit réelles, soit elles apparaissent en paires complexe-conjuguées. Dans ce dernier cas, $\lambda_j = \bar{\lambda}_{j+1} = \rho_j e^{i\theta_j}$ où $\bar{\lambda}$ désigne le conjugué de λ, et ρ, θ sont des coefficients réels.

Une condition nécessaire et suffisante pour assurer la stabilité de Eq. 3.74 est que ces valeurs propres soient inclues dans le cercle unité du plan complexe, i.e. $\rho_j < 1$ pour $j = 1, \ldots, D$. Dans ce cas, u_n converge vers l'origine lorsque $n \to \infty$.

A ce stade, il est important de dresser la liste les différents types de trajectoires auxquelles on peut s'attendre en fonction des valeurs propres obtenues. En combinant les solutions, j et $j+1$, il vient deux solutions *réelles* linéairement indépendantes,

$$g_j(n) = \frac{1}{2}(\lambda_j^n + \lambda_{j+1}^n) = \rho_j^n \cos(n\theta_j) \tag{3.79}$$

$$g_{j+1}(n) = \frac{1}{2i}(\lambda_j^n - \lambda_{j+1}^n) = \rho_j^n \sin(n\theta_j) \tag{3.80}$$

Si λ_j est réel ($\lambda_j = \rho_j$), on peut écrire $g_j(n) = \rho_j^n$. Eq. 3.78 devient

$$u_n = \sum_{j=1}^{D} c_j g_j(n) \tag{3.81}$$

Les coefficients c_j sont désormais réels, à la différence des b_j dand Eq. 3.78. L'expression (3.81) est facile à interpréter car elle exprime la suite comme une somme de termes décroissants exponentiellement vers zéro. De plus, le comportement asymptotique de u_n dans le cas stable est déterminé par $\mathrm{Min}_j(\rho_j) = \hat{\rho}$. Dans le cas où le coefficient associé, $\hat{\lambda}$, est réel, u_n oscille dans un premier temps puis décroît à vitesse exponentielle (de coefficient $\hat{\rho}$) lorsque $n \to \infty$. Sinon la décroissance de u_n ressemble à une oscillation amortie vers l'origine. Cette situation sera clairement illustrée Fig. 4.12, 4.18, 4.21 et 4.28.

Pour assurer la stabilité de la rétro-propagation, il faut établir des conditions suffisantes pour que les valeurs propres tombent à l'intérieur du cercle unité. Pour cela, nous cherchons des conditions suffisantes pour avoir $P(\lambda) \neq 0$ lorsque $|\lambda| > 1$. Ecrivons

$$\begin{aligned} \mid \sum_{n=0}^{D-1} a_{D-n} \lambda^n \mid \quad &\leq \quad \sum_{n=0}^{D-1} a_{D-n} \mid \lambda \mid^n \\ &\leq \quad \mathrm{Max}_i(a_i) \sum_{n=0}^{D-1} \mid \lambda \mid^n \\ &\leq \quad \mathrm{Max}_i(a_i) \frac{1- \mid \lambda \mid^D}{1- \mid \lambda \mid} \end{aligned} \tag{3.82}$$

Il est immédiat de noter que si $|\lambda| > 1$, les inégalités suivantes sont satisfaites

$$\frac{1}{D} \frac{1- \mid \lambda \mid^D}{1- \mid \lambda \mid} < \mid \lambda \mid^D \tag{3.83}$$

Par conséquent, on a $\mid P(\lambda) \mid > 0$ lorsque $\mid \lambda \mid 1$ et $\mathrm{Max}_i(a_i) < 1/D$, avec le résultat que les racines de $P(\lambda)$ gisent nécessairement à l'intérieur du cercle unité. Il vient

$$\|\mathbf{W}_k^0\| < 1 \ \text{ et } \ \frac{\|\mathbf{W}_k^d\|}{1 - \|\mathbf{W}_k^0\|} < \frac{1}{D} \quad \forall k \leq k_f, \forall d \leq D \tag{3.84}$$

sont des conditions suffisantes pour assurer la décroissance asymptotique de u_n, et donc de $\|\mathbf{y}_{k_f-n}^{k_f}\|$. Aussi, lorsque les conditions (3.84) sont vérifiées $\forall k_f > 0$, on peut toujours trouver $M > 0$ tel que

$$u_n < \|\mathbf{y}_{k_f-n}^{k_f}\| < M(\hat{\rho})^n, \ \ \forall n \tag{3.85}$$

cqfd.