Utilisation d'un ordinateur réservoir pour décrypter un message transmis via la cryptographie par chaos

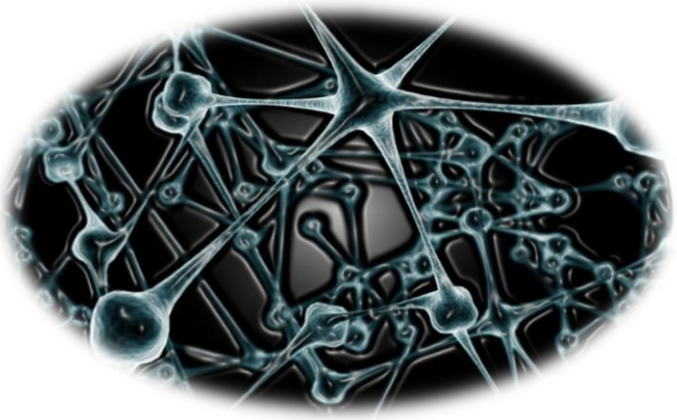
Gulina, Marvyn

Publication date:
2017

Document Version
Première version, également connu sous le nom de pré-print

Utilisation d'un ordinateur réservoir pour décrypter un message transmis via la cryptographie par chaos

Mémoire présenté en vue de l'obtention
du diplôme de Master en Sciences Physiques



Par

• MARVYN GULINA •

Sous la direction du Professeur **Serge MASSAR**

Avec l'assistance de **Piotr ANTONIK** et de **Jaël PAUWELS**

Faculté des Sciences – Département de Physique

Laboratoire d'Information Quantique (LIQ)

Année académique 2016 – 2017

UNIVERSITÉ LIBRE DE BRUXELLES ULB

Remerciements

Pour commencer, j'aimerais rappeler que ce mémoire représente la fin d'une vingtaine d'années de formation scolaire. Durant tout ce temps passé sur les bancs de l'école, ce sont ma famille, ma compagne, mes amis mais également mes professeurs qui ont sculpté, petit à petit, la personne que je suis devenue aujourd'hui. Ce sont toutes ces personnes qui, plus ou moins directement, ont contribué à la réalisation de ce travail et à qui je dois énormément.

Pour respecter une vieille promesse faite à moi-même, j'aimerais particulièrement remercier certains de mes anciens professeurs qui ont radicalement influencé mon parcours scolaire :

- ▷ Madame Brigitte LEROY qui m'a appris à lire et à compter en première primaire.
- ▷ Monsieur Vincent JOACHIM qui m'a fait apprécier l'école en sixième primaire.
- ▷ Madame Camelia MAIORU qui m'a fait découvrir la beauté des mathématiques à travers son accent inoubliable en quatrième secondaire.
- ▷ Mesdames Sylviane TAXHET et Mireille LOVATO qui m'ont transmis de bonnes bases en mathématiques et en physique en sixième secondaire.
- ▷ Madame Anne LEMAITRE qui m'a permis de suivre un cursus de 4 ans à l'Université de Namur (anciennement Facultés Universitaires Notre-Dame de la Paix) aboutissant à un double titre de bachelier en Sciences Physiques et Mathématiques.

Ensuite c'est mon binôme Adrien FIORUCCI que je mets à l'honneur. Nous nous sommes rencontrés en première année du bachelier et, depuis lors, il m'a accompagné, soutenu et extraordinairement aidé à gravir chaque échelon universitaire pour finalement arriver jusqu'ici.

Je tiens également à remercier le Professeur Serge MASSAR pour la direction attentionnée de ce mémoire et pour m'avoir toujours poussé à aller plus loin dans mes investigations numériques. Enfin, je suis très reconnaissant envers messieurs Piotr ANTONIK et Jaël PAUWELS qui m'ont encadré à tour de rôle et m'ont permis de réaliser ce travail dans les meilleurs conditions.

Pour terminer, je rends ce mémoire en Esprit à ceux sans qui rien de tout ceci n'aurait été possible : notre Père qui es aux cieux et son Fils Jésus Christ.

Marvyn GULINA le 15 mai 2017, avec Amour : ainsi soit-il ...

Résumé / Abstract

Nous commençons par introduire les notions de point fixe, d'orbite, de diagramme de bifurcation, d'attracteur, d'exposant de Liapounov et de chaos pour « planter le décor » relatif aux systèmes dynamiques. Ensuite, nous explicitons deux méthodes de communication utilisant la cryptographie par chaos. Nous continuons avec l'implémentation d'un ordinateur réservoir et nous voyons comment l'entraîner pour émuler un système dynamique. Enfin, nous montrons comment utiliser ce réservoir pour décrypter un message transmis via les méthodes de communication mentionnées ci-dessus.

Mots-clés : ordinateur réservoir ; réseau de neurones récurrents ; réseau d'état d'écho ; cryptographie par chaos.

We begin by introducing the notions of fixed point, orbit, bifurcation diagram, attractor, exponent of Lyapounov and chaos to « set the scene » for dynamical systems. Then, we explain two methods of communication using chaos based cryptography. We continue with the implementation and training of a reservoir computer to emulate a dynamical system. Finally, we show how this reservoir manages to decrypt a message transmited by the above-mentioned communications.

Key words : reservoir computer ; recurrent neural network ; echo state network ; chaos based cryptography.

Contenu

Listes

Figures

Tableaux

Définitions

Théorèmes

Résultats

Protocoles

Avant-propos

Motivations personnelles

Trouver un sujet de mémoire n'est pas nécessairement une chose aisée, surtout lorsque l'on débarque d'une autre université. Outre cette méconnaissance des lieux et des chercheurs qui y travaillent, l'auteur ne se connaissait même pas lui même : il ne savait pas comment orienter son choix. Il lui aura fallu plusieurs mois pour décider de se diriger vers le numérique. Ce choix étant fait, il était grand temps de trouver un sujet et un directeur.

Après avoir essuyé plusieurs refus, Alexandre Mayer et Benoît Frenay de l'Université de Namur proposèrent de travailler sur l'apprentissage automatique (*machine learning*). L'auteur en discuta alors avec le jury de l'Université Libre de Bruxelles qui le mit en relation avec son futur directeur : Serge Massar. Ce dernier accepta la proposition faite par les chercheurs de Namur mais il proposa également un sujet alternatif dans un domaine similaire : l'ordinateur réservoir (*reservoir computer*). Finalement, le choix s'est porté sur la seconde proposition.

Termes techniques

Dans tout travail scientifique (ou non) on voit apparaître des termes spécifiques au domaine concerné. Alors que la langue maternelle de l'auteur est le français, c'est plutôt l'anglais qui domine la littérature en Science Physique. Malgré l'attachement de l'auteur vis-à-vis de sa langue, il ne voulait pas nuire à son lecteur en lui présentant uniquement des traductions maladroites. En effet, pour certains il est préférable de parler d'« *echo state network* » qui sera un système bien connu alors que « le réseau d'état d'écho » paraîtra obscur pour tout le monde.

C'est en lisant sa Bible que l'auteur s'est finalement décidé à couper la poire en deux (comme le fit le roi Salomon aux versets 1 Rois 3.16-28 de [25]) et qu'il a choisi de mêler les deux langues. D'une part, tout lecteur sera amené à rencontrer les termes techniques exprimés en anglais et utilisés par la communauté scientifique. Ceci devrait faciliter la lecture des chapitres 3 et 4 concernant le réservoir comme celle d'articles spécialisés sur ce sujet. D'autre part, l'auteur aura la satisfaction de proposer un travail (presque) en français.

Objectifs

Toute notre problématique tourne autour de l'affirmation suivante, le titre de cet ouvrage :

Utilisation d'un ordinateur réservoir
pour décrypter un message transmis via la cryptographie par chaos.

L'ordinateur réservoir est un système dynamique basé sur les réseaux de neurones artificiels. On utilise l'approche d'état d'écho sur un réseau de neurones récurrents à fuite pour implémenter *numériquement* le réservoir. Ensuite, on l'entraîne à reproduire la dynamique d'un système chaotique donné : le système de Mackey - Glass.

Enfin, nous allons confronter le signal émulé par le réservoir à la cryptographie par chaos. Pour cela, nous explicitons deux schémas de communication : la superposition et le mélange non-linéaire. Dans chacune de ces situations, nous avons l'un des protagonistes (Alice) qui essaye de transmettre un message à son collègue (Bob) en le combinant à un signal chaotique. Les deux méthodes que l'on utilise se distinguent par leur comportement vis-à-vis du signal porteur. En effet, la première est caractérisée par le fait que l'ajout du message n'affecte pas la dynamique de la porteuse, tandis qu'il va la modifier dans la seconde.

Notons qu'en comparant le message récupéré à celui reconstruit (*cf.* ci-dessous), on aura accès à une mesure de la qualité de l'imitation du réservoir.

- ▷ **Message récupéré** : décrypté de manière traditionnelle en fournissant à Bob une copie du système chaotique d'Alice ;
- ▷ **Message reconstruit** : décrypté en fournissant à Bob un ordinateur réservoir entraîné à l'aide du système chaotique d'Alice.

Finalement, il est intéressant d'utiliser un ordinateur réservoir dans cette situation car si Bob parvient à récupérer l'information transmise par Alice en utilisant uniquement un *reservoir computer* entraîné, alors l'argument principal sur lequel se fonde la sécurité de la cryptographie par chaos se voit déstabilisé. En effet, Bob parviendrait à lire le message envoyé par Alice sans posséder le même système physique.

Au terme de ce travail nous montreront comment, dans chacune des deux méthodes de communication sus-mentionnées, nous avons réussi à décrypté un message « simple » à l'aide d'un réservoir.

Plan de lecture

Les quatre premiers chapitres constituent la « partie théorique » alors que le cinquième présente les résultats de nos expériences numériques. Nous donnons ci-dessous un bref descriptif de chaque chapitre afin de fournir au lecteur une vue d'ensemble du document :

Chapitre 1 : Nous commençons par introduire quelques généralités sur les systèmes dynamiques, en particulier la notion de chaos. Nous donnons également les équations relatives au système de Mackey - Glass que nous utiliserons tout au long de ce document.

Chapitre 2 : Dans ce chapitre nous décrivons et testons numériquement deux méthodes de communication basées sur le chaos.

Chapitre 3 : Nous continuons en montrant comment implémenter un ordinateur réservoir.

Chapitre 4 : Une fois implémenté, nous voyons comment entraîner, par apprentissage forcé, notre réservoir à l'émulation d'un système chaotique.

Chapitre 5 : Finalement, le dernier chapitre présente nos résultats quant à la reconstruction d'un message transmis via les méthodes basées sur le chaos que nous auront explicitées au chapitre 2.

Le lecteur trouvera quatre annexes en fin de document. Comme pour les chapitres, nous les présentons ici en quelques mots pour permettre au lecteur de s'organiser.

Annexe A : Complète le chapitre 1 mais n'est pas nécessaire à la compréhension du texte.

Annexe B : Nécessaire d'intégration numérique et application au système de Mackey - Glass. Nous en recommandons la lecture pour celui qui n'aurait jamais intégré numériquement une équation différentielle.

Annexe C : Complète le chapitre 3. Le lecteur peut s'en passer si la section 3.1 lui est suffisante à la compréhension du neurone formel et s'il admet la dynamique d'un réseau de neurones à fuite donnée à l'équation (3.3).

Nous avons placé cette partie en annexe pour ne pas alourdir la lecture du chapitre en question.

Annexe D : Nous donnons un hyperlien permettant le téléchargement de l'ensemble de nos codes implémentés en MATLAB® plutôt que de les inclure directement en annexe. A cela, nous ajoutons un « lisez-moi » afin d'aider le lecteur qui souhaite parcourir nos « quelques » fichiers *.m*.

Pour conclure ces prolégomènes, nous prévenons le lecteur que nous avons employé une approche très pédagogique dans notre rédaction. En effet, nous souhaitions atteindre un maximum de personnes en rendant notre contenu le plus accessible possible. De plus, nous espérons que le lecteur apprécie sa lecture et qu'il la termine en ayant appris quelque chose car, finalement, c'est peut-être bien le plus important...

Bonne lecture !

Chapitre 1

Généralités sur l'étude d'un système dynamique

Ce premier chapitre introduit les outils de base pour l'étude des systèmes dynamiques à partir d'un exemple simple mais non trivial : la famille d'applications en tente.
Ainsi, à l'issue de cette introduction, le lecteur comprendra des termes comme : point fixe, orbite, diagramme de bifurcation, attracteur, exposant de Liapounov et chaos.

Nous faisons d'ores et déjà l'hypothèse que toutes les applications considérées dans ce chapitre sont continues. En outre, nous supposons qu'elles sont « suffisamment lisses » pour ne pas avoir à l'écrire pour chacune des définitions ou chacun des théorèmes exposés. Nous précisons également que nous noterons $I \overset{not}{=} [0,1] \subset \mathbb{R}$ tout au long de cette introduction.

Enfin, nous terminerons ce chapitre en introduisant à la section 1.7 le système de Mackey - Glass qui nous sera utile dans la suite.

Le lecteur souhaitant s'attarder sur les systèmes dynamiques peut consulter les références données dans la section correspondante de la bibliographie.

1.1 La famille d'applications en tente

Bien qu'il existe de nombreuses façons de formuler la famille d'applications en tente, nous nous bornerons à travailler avec l'expression suivante :

$$T_\mu : I \to I : x \mapsto T_\mu(x) = \frac{\mu}{4}(1 - |2x - 1|) \tag{1.1}$$

avec $\mu \in]0\,,4]$. Toutes ces applications sont évidement continues. Outre sa simplicité propice à la pédagogie, T_4 est topologiquement conjuguée à la logistique $Q_4(x) = 4x\,(1-x)$, bien connue pour modéliser l'évolution d'une population. Pour plus de détails sur la conjugaison, voir annexe A.

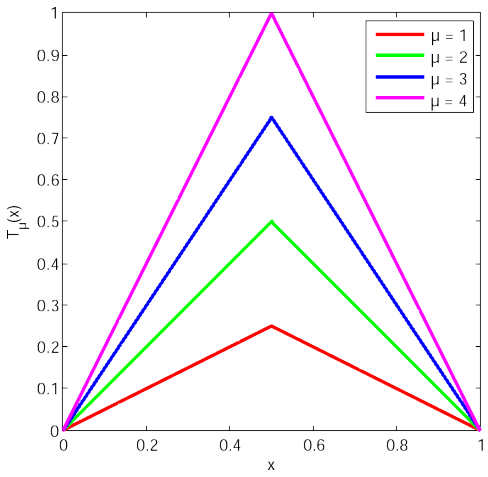


FIGURE 1.1 – Quelques exemples de la famille d'applications en tente.

Dans la suite nous aurons également besoin de la dérivée de T_μ. Le calcul est aisé et fournit :

$$T'_\mu(x) = \begin{cases} \mu/2 & 0 \leq x < 1/2 \\ -\mu/2 & 1/2 < x \leq 1 \end{cases} \Longrightarrow \forall x \in I \setminus \left\{\frac{1}{2}\right\} : \left|T'_\mu(x)\right| = \frac{\mu}{2} \tag{1.2}$$

Enfin, nous listons ici quelques propriétés utiles :

(1) **Symétrique** : $T_\mu\left(\frac{1}{2}+x\right) = T_\mu\left(\frac{1}{2}-x\right),\ \forall x \in \left[0\,,\frac{1}{2}\right]$.

(2) **Concave** : T_μ est strictement croissante sur $\left[0\,,\frac{1}{2}\right]$ et strictement décroissante sur $\left[\frac{1}{2}\,,1\right]$.

(3) **Unimodale** : T_μ est concave et atteint son unique maximum de $\mu/4$ en $x = \frac{1}{2}$.

1.2 Stabilité et dynamique au voisinage des points fixes

Les points fixes jouent un rôle important dans la dynamique de T_μ. Ils sont définit simplement comme :

Définition 1.1 (Point fixe)**.**

Soit f une application et $x \in \text{Dom}f$. On dira de x qu'il est un point fixe de f si et seulement si $f(x) = x$. On notera $\mathcal{E}(f)$ l'ensemble des points fixes de f.

Étudions les points fixes de T_μ. En solutionnant $T_\mu(x) = x$, on trouve facilement ce qui suit :

▷ $\boldsymbol{\mu < 2} \Longrightarrow \mathcal{E}(T_\mu) = \{0\}$.

▷ $\boldsymbol{\mu = 2} \Longrightarrow \mathcal{E}(T_\mu) = \left[0\,,\frac{1}{2}\right]$.

▷ $\boldsymbol{\mu > 2} \Longrightarrow \mathcal{E}(T_\mu) = \left\{0\,;\frac{\mu}{2+\mu}\right\}$ [1].

Comme leur nom l'indique, ces points resteront fixes sous l'action (éventuellement répétée) de T_μ. Dès lors, on comprend qu'étudier la dérivée de T_μ au voisinage de ses points fixes fournira des informations sur la dynamique locale de T_μ. Formalisons ce raisonnement à l'aide de la définition 1.2 et du théorème 1.1.

Définition 1.2 (Stabilité des points fixes)**.**

Soit f une application et $x \in \mathcal{E}(f)$, un point fixe de f. Alors x est :

▷ un **attracteur** si $|f'(x)| < 1$

▷ un **répulseur** si $|f'(x)| > 1$

▷ un point fixe **indifférent** si $|f'(x)| = 1$

1. Le second point est obtenu en trouvant l'intersection entre $d_1 \equiv y = \frac{\mu}{2}(1-x)$ et $d_2 \equiv y = x$.

Le théorème suivant permet d'illustrer la terminologie de cette définition mais cette dernière prendra vraiment tout son sens un peu plus loin, au paragraphe 1.3.

Théorème 1.1 (Dynamique au voisinage d'un point fixe)**.**

Soit $f : A \to A$ où A est un intervalle réel, et $p \in \mathcal{E}(f)$, un point fixe de f. Alors :

(1) Si p est un attracteur, alors il existe un voisinage V contenant p, ouvert dans A, tel que f est contractante sur V.

(2) Si p est un répulseur, alors il existe un voisinage W contenant p, ouvert dans A, tel que f est dilatante sur $W \setminus \{p\}$.

Preuve

Nous faisons la démonstration uniquement dans le cas où p est un attracteur, l'autre étant tout à fait analogue. Commençons par exploiter la définition de la dérivée au point p :

$$f'(p) = \lim_{x \to p} \frac{f(x) - f(p)}{x - p}$$

C'est-à-dire :

$$\forall \varepsilon > 0 : \exists \delta(p, \varepsilon) > 0 \text{ tel que } \forall x \in A : \left[|x - p| < \delta \Rightarrow \left| \frac{f(x) - f(p)}{x - p} - f'(p) \right| < \varepsilon \right]$$

Vu que l'on a $|f'(p)| < 1$, on peut prendre un $\tilde{\varepsilon} > 0$ tel que $|f'(p)| + \tilde{\varepsilon} \overset{def}{=} a < 1$.
Mais alors, si on considère la boule ouverte centrée en p et de rayon $\delta(p, \tilde{\varepsilon})$ il vient :

$$\forall x \in B(p, \delta(p, \tilde{\varepsilon})) : \left| \frac{f(x) - f(p)}{x - p} \right| = \left| \frac{f(x) - f(p)}{x - p} - \boldsymbol{f'(p) + f'(p)} \right| \leq \tilde{\varepsilon} + |f'(p)| = a < 1$$

$$\Longrightarrow |f(x) - f(p)| \leq a\,|x - p|$$

Dans cette boule l'application f est contractante : prendre $V = B(p, \delta(p, \tilde{\varepsilon}))$ convient. □

En ce qui concerne la famille d'applications que nous étudions on peut, compte tenu de l'équation (1.2), étudier la stabilité des points fixes de T_μ que l'on a déterminés plus haut :

- ▷ $\boldsymbol{\mu < 2} \Longrightarrow \left|T'_\mu(x)\right| < 1$ et le seul point fixe est un attracteur.
- ▷ $\boldsymbol{\mu = 2} \Longrightarrow \left|T'_\mu(x)\right| = 1$ et les points fixes sont indifférents.
- ▷ $\boldsymbol{\mu > 2} \Longrightarrow \left|T'_\mu(x)\right| > 1$ et les deux points fixes sont des répulseurs.

1.3 Orbites

Pour autant que $0 < \mu \leq 4$, l'application associée T_μ envoie l'intervalle I dans $[0, \mu/4] \subseteq I$ et donc on peut itérer de manière bien définie l'action de T_μ sur un point x_0 de I. On définit alors naturellement une orbite de la manière suivante :

Définition 1.3 (Orbite).

Soit $f : A \to A$. Une orbite de f est une suite de $n+1$ points, notée $(x_i)_{i=0}^n \subset A$, construite de manière itérative à partir du point initial $x_0 \in A$:

$$x_{i+1} = f(x_i) \ , \quad \forall i \in \{0, \dots, n-1\} \tag{1.3}$$

De cette définition découle un corollaire important du théorème 1.1 quant à la convergence d'une orbite au voisinage d'un point fixe.

Théorème 1.2 (Dynamique des orbites au voisinage d'un point fixe).

Soit $f : A \to A$ où A est un intervalle réel, et $p \in \mathcal{E}(f)$, un point fixe de f. Alors :

(1) Si p est un attracteur, alors il existe un voisinage V contenant p, ouvert dans A, tel que toute orbite avec donnée initiale $x_0 \in V$ converge vers p.

(2) Si p est un répulseur, alors il existe un voisinage W contenant p, ouvert dans A, tel qu'aucune orbite avec donnée initiale $x_0 \in W \setminus \{p\}$ restera dans W.

Preuve

(1) Le théorème 1.1 affirme l'existence d'un ouvert V contenant p dans lequel l'application f est contractante de constante a. Dès lors, pour $x_0 \in V$, on a :

$$|x_n - p| = |f(x_{n-1}) - f(p)| \leq a\,|x_{n-1} - p| = a\,|f(x_{n-2}) - f(p)| \leq \cdots \leq a^n\,|x_0 - p|$$

Puisque $|x_0 - p|$ est borné et que $a < 1$, à la limite où n tend vers l'infini, on a $x_n \to p$.

(2) La démonstration n'est pas tout à fait analogue. Le théorème 1.1 affirme l'existence d'un ouvert W contenant p dans lequel l'application f est dilatante de constante a. Dès lors, pour $x_0 \in W$, on a :

$$|x_n - p| = |f(x_{n-1}) - f(p)| \geq a\,|x_{n-1} - p| = a\,|f(x_{n-2}) - f(p)| \geq \cdots \geq a^n\,|x_0 - p|$$

Puisque $|x_0 - p|$ est borné non-nul et que $a > 1$, à la limite où n tend vers l'infini, on a $x_n \to \infty$ tant que $x_n \in W$. Or W est borné, donc : $\exists \tilde{n} \in \mathbb{N}$ tel que $x_{\tilde{n}} \notin W$. Ceci traduit bien l'impossibilité pour une orbite où $x_0 \in W \setminus \{p\}$ de rester dans W.

□

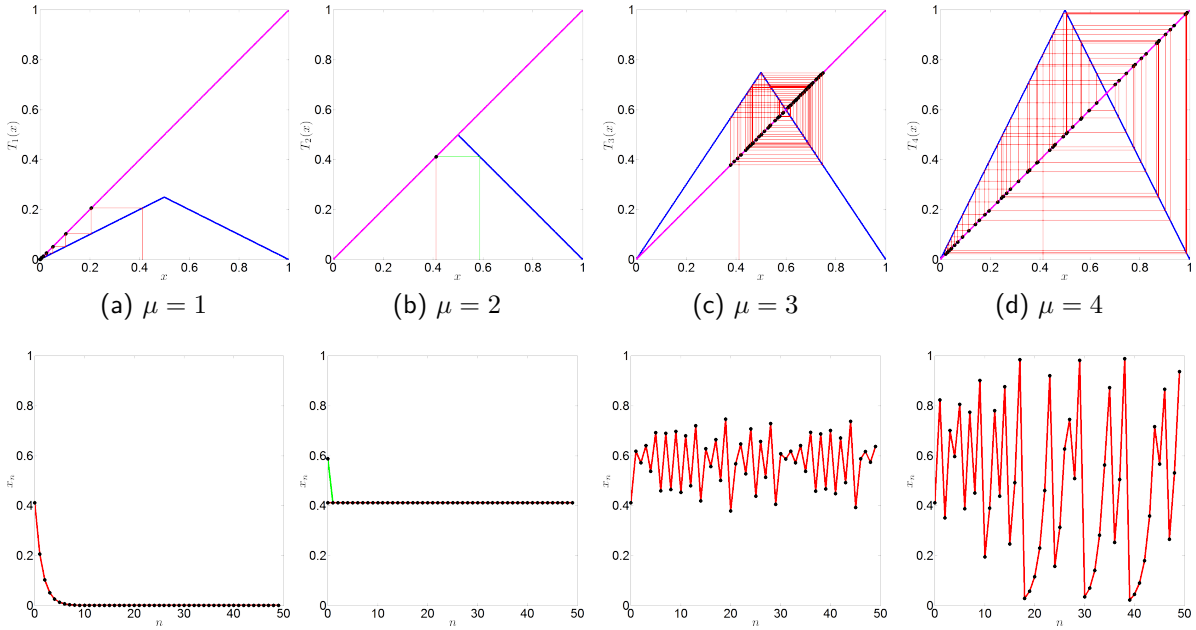
La série de figures 1.2 montre l'orbite de T_μ construite en suivant la règle itérative de la définition 1.3 à partir de $x_0 = 0.4123$ pour différentes valeurs de μ. Nous pouvons observer sur ces figures différentes dynamiques autour des points fixes :

▷ $\boldsymbol{\mu < 2}$: Toutes les orbites convergent vers l'origine (fig. 1.2a).

▷ $\boldsymbol{\mu = 2}$: Toutes les orbites convergent vers x_0 si $x_0 \leq \frac{1}{2}$ ou vers $1 - x_0$ si $x_0 > \frac{1}{2}$ (fig. 1.2b).

▷ $\boldsymbol{\mu > 2}$: L'orbite est bornée pour $\mu = 3$ (fig. 1.2c) et couvre tout I pour $\mu = 4$ (fig. 1.2d).

Notons que ces changements de dynamique s'opèrent bien lorsque l'ensemble des points fixes de T_μ passe de $\{0\}$ à $\left[0\,,\frac{1}{2}\right]$ puis à $\{0\,;\frac{\mu}{2+\mu}\}$ respectivement lorsque $\mu < 2$, $\mu = 2$ et $\mu > 2$. Ceux-ci illustrent bien l'influence des points fixes sur la dynamique du système et le théorème 1.2.



(a) $\mu = 1$ (b) $\mu = 2$ (c) $\mu = 3$ (d) $\mu = 4$

FIGURE 1.2 – Orbites constituées de 50 points (en rouge) de $T_\mu(x) = \frac{\mu}{4}(1 - |2x - 1|)$ (en bleu) au départ de $x_0 = 0.4123$ pour différentes valeurs de μ. La droite magenta est $f(x) = x$.

Les différents comportements observés nous conduisent à donner la définition suivante :

Définition 1.4 (Bassin d'attraction et attracteur)**.**

Soit $f : A \to A$ où A est un intervalle réel.
L'ensemble $B_E \subset A$ est le **bassin d'attraction** de $E \subset A$ si et seulement si pour toute donnée initiale $x_0 \in B_E$ le système f évolue irréversiblement vers E, en l'absence de perturbation. L'ensemble E sera alors appelé **attracteur**.

Nous discutons un peu plus cette notion dans la section suivante.

1.4 Diagramme de bifurcation

Afin de mieux comprendre le comportement des orbites, nous étudions le diagramme de bifurcation de la famille d'applications en tente qui est donné à la figure 1.3.

Définition 1.5 (Diagramme de bifurcation)**.**

Soit $f_\mu : A \to A$ une famille d'applications avec $\mu \in \mathbb{R}$ et où A est un intervalle réel.
Son diagramme de bifurcation représente les attracteurs E_μ de f_μ
en fonction du paramètre de bifurcation μ.

Ce dernier est construit numériquement, pour T_μ, de la manière suivante[2] :

▷ On discrétise les intervalles $[0\,,4]$ et $[0\,,1]$ en 4000 points pour définir les valeurs de μ et x qui formeront les pixels de la figure 1.3. Ensuite, pour chaque valeur de μ :

- On fixe arbitrairement une valeur de x_0 dans $I \setminus \left\{0\,;\frac{\mu}{2+\mu}\,;1\right\}$.
  Comme $\frac{\mu}{2+\mu}$ va parcourir $\left[\frac{1}{2}\,,\frac{2}{3}\right]$ lorsque μ sera dans $[2\,,4]$ (car ce point fixe n'existe pas pour $\mu < 2$), on peut prendre $x_0 = 0.75$ par exemple ;
- On stabilise l'orbite par 1000 premières itérations de T_μ ;
- On itère ensuite 100 000 fois en augmentant de 1 la valeur du pixel $(\mu\,,x)$ correspondant ;
- Enfin, on multiplie chaque pixel de cette colonne par le nombre de pixels non-nuls qu'elle contient en imposant que la valeur maximale pour un pixel soit 180 000. Cette opération permet simplement d'ajuster le contraste du diagramme. Une borne plus petite ou plus grande donnerait un rendu plus foncé ou plus clair respectivement.

▷ L'image finale est obtenue normalisant 180 000 à 255, puis en inversant les couleurs. En termes mathématiques, on a effectué la transformation $c \mapsto 255(1 - \frac{c}{180000})$ sur chaque pixel c.

Outre la représentation des attracteurs, ce diagramme nous permet de visualiser l'évolution de la complexité des orbites en fonction du paramètre μ. En effet, plus l'orbite de T_μ passe par le point x, plus le couple $(\mu\,,x)$ apparaîtra foncé dans le diagramme. Parmi ces points, on a les points périodiques :

Définition 1.6 (Point périodique)**.**

Soit $f : A \to A$, $x \in A$ est un point périodique de f de période n si et seulement si :

$$n \text{ est le plus petit naturel tel que } : f^{\circ n}(x) = x \tag{1.4}$$

En d'autres termes, x est un point fixe de $f^{\circ n}$, l'itérée $n^{\text{ème}}$ de f.

Remarquez que l'on a les inclusions suivantes : $\forall m \in \mathbb{N}_0, \forall n \in \mathbb{N}_0 : \mathcal{E}(f^{\circ n}) \subset \mathcal{E}(f^{\circ (mn)})$.

2. Bien que cette méthode puisse être adaptée à la construction d'autres diagrammes de bifurcation, nous précisons qu'elle n'est pas générale.

Dès lors, le diagramme de bifurcation nous montre également quels sont les points périodiques de T_μ en fonction du paramètre de bifurcation μ.

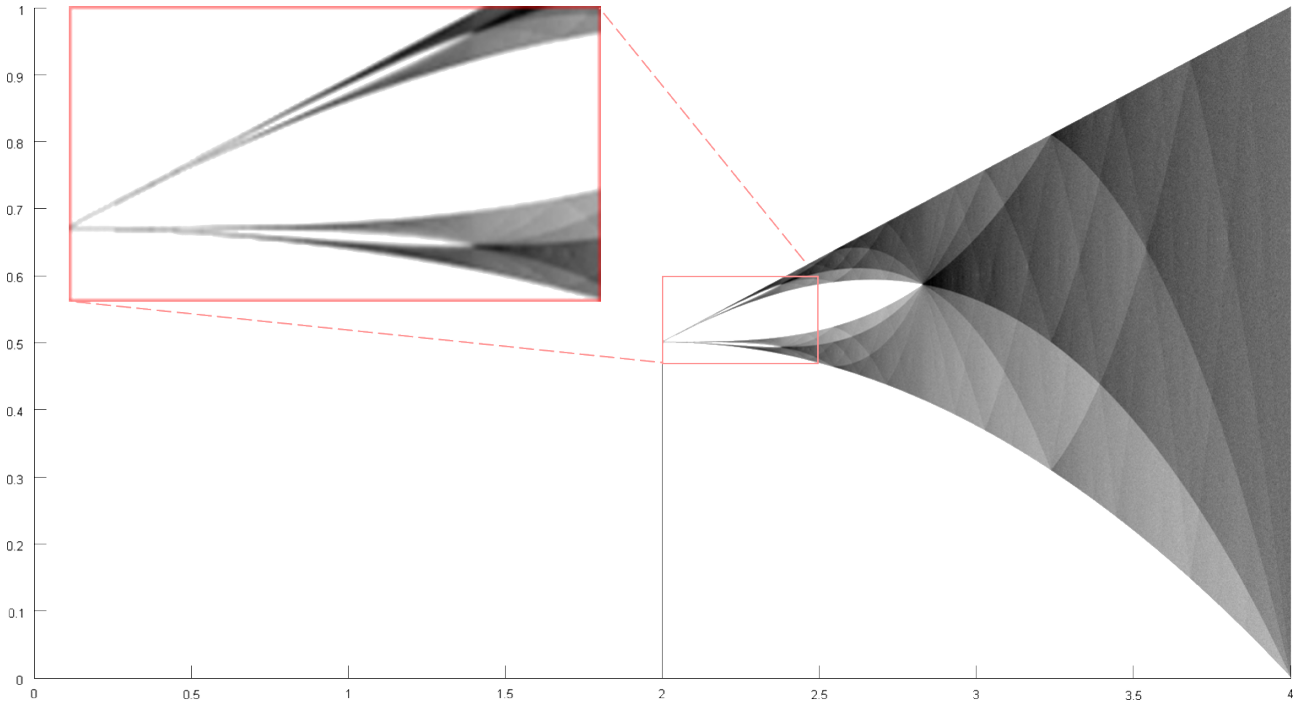


FIGURE 1.3 – Diagramme de bifurcation de la famille d'applications en tente T_μ où les points plus foncés ont plus de probabilités d'être obtenu pour la valeur de μ associée.

Voyons maintenant quelles sont les informations utiles que l'on peut tirer de la figure 1.3 :

- ▷ $\boldsymbol{\mu < 2}$: On a vu que l'origine attirait toutes les orbites (fig. 1.2a), elle est le seul point périodique et le diagramme se confond avec l'axe horizontal. L'attracteur est : $E_\mu = \{0\}$.
- ▷ $\boldsymbol{\mu = 2}$: Tous les points de l'intervalle $\left[0\,,\frac{1}{2}\right]$ bloquent la dynamique.
  Ceux de l'intervalle $\left[\frac{1}{2}\,,1\right]$ sont envoyés sur leur symétrique et ils y restent car ces derniers sont les points fixes. En effet, T_μ est symétrique (propriété (1)) et la branche montante de la tente se confond avec la droite $y = x$ pour $\mu = 2$. L'attracteur devient l'intervalle $E_2 = \left[0\,,\frac{1}{2}\right]$ qui contient bien les seuls points périodiques.
- ▷ $\boldsymbol{\mu > 2}$: Quatre branches naissent du point $\left(2\,,\frac{1}{2}\right)$ (voir zoom dans la figure 1.3). Les deux branches externes s'étendent jusque 0 et 1 au fur et à mesure que μ tend vers 4 et les deux branches internes se rejoignent lorsque $\mu \approx 2.8$. Finalement, l'ensemble des points périodiques (qui sont alors des répulseurs car $\mu > 2$) s'étend jusqu'à devenir dense dans I : c'est le chaos (voir section 1.6).

1.5 Exposant de Liapounov

L'exposant de Liapounov – ou Lyapunov dans sa « version anglaise » – généralement noté λ, est un nombre réel ou infini qui quantifie la stabilité d'un système dynamique.

Définition 1.7 (Exposant de Liapounov)**.**

L'exposant de Liapounov d'une application f définissant une orbite telle que (1.3) est le logarithme moyen de son accroissement :

$$\lambda = \lim_{n\to+\infty} \frac{1}{n} \sum_{i=1}^{n} \ln\left(|f'(x_{i-1})|\right) \tag{1.5}$$

Le raisonnement qui suit permet de comprendre l'expression (1.5).

▷ Considérons ε_0 une erreur initiale sur x_0. Il en résulte ε_n, l'erreur à l'itération n, c'est-à-dire :

$$f(x_{n-1}+\varepsilon_{n-1}) - f(x_{n-1}) = x_n + \varepsilon_n - x_n = \varepsilon_n \Longrightarrow \frac{\varepsilon_n}{\varepsilon_{n-1}} = \frac{f(x_{n-1}+\varepsilon_{n-1}) - f(x_{n-1})}{\varepsilon_{n-1}}$$

On observe immédiatement que lorsque ε_{n-1} tend vers 0, ε_n tend également vers 0 et le rapport $\varepsilon_n/\varepsilon_{n-1}$, qui n'est rien d'autre que la mesure de l'amplification instantanée de l'erreur, tend vers $f'(x_{n-1})$, la dérivée de f évaluée en x_{n-1}.

▷ Calculons l'amplification de l'erreur entre le point de départ et le $n^{ème}$ élément :

$$\left|\frac{\varepsilon_n}{\varepsilon_0}\right| = \prod_{i=1}^{n} \left|\frac{\varepsilon_i}{\varepsilon_{i-1}}\right| \to \prod_{i=1}^{n} |f'(x_{i-1})|$$

▷ On retrouve bien la définition 1.7 :

$$|\varepsilon_n| = e^{\lambda n} |\varepsilon_0| \Longrightarrow \lambda = \lim_{n\to+\infty} \frac{1}{n} \sum_{i=1}^{n} \ln\left(|f'(x_{i-1})|\right)$$

L'exposant de Liapounov indique le taux auquel vont se séparer deux orbites initialement proches.

Avant de donner le résultat pour T_μ interprétons le produit des dérivées qui apparaît ci-dessus :

$$\left.\frac{df^{\circ n}(x)}{dx}\right|_{x_0} = f'\left(f^{\circ(n-1)}(x_0)\right) \left.\frac{df^{\circ(n-1)}(x)}{dx}\right|_{x_0} = f'\left(x_{n-1}\right) \left.\frac{df^{\circ(n-1)}(x)}{dx}\right|_{x_0} = \cdots = \prod_{i=1}^{n} f'(x_{i-1}),$$

c'est-à-dire la dérivée le long de l'orbite construite à partir de x_0.

Résultat 1.1 (Exposant de Liapounov de la famille d'applications en tente)**.**

Soit T_μ la famille d'applications définie par (1.1), son exposant de Liapounov est :

$$\lambda_{T_\mu} = \ln\left(\frac{\mu}{2}\right) \tag{1.6}$$

Preuve

Reprenons l'équation (1.2) qui nous donnait l'expression de la dérivée de T_μ :

$$\forall x \in I \setminus \left\{\frac{1}{2}\right\} : \left|T'_\mu(x)\right| = \mu/2 \equiv \text{constante}$$

On applique alors la définition (1.5) pour obtenir directement λ_{T_μ} :

$$\lambda_{T_\mu} \triangleq \lim_{n\to+\infty} \frac{1}{n}\sum_{i=1}^{n} \ln\left(\left|T'_\mu(x_{i-1})\right|\right) = \ln\left(\frac{\mu}{2}\right) \lim_{n\to+\infty} \frac{1}{n}\sum_{i=1}^{n} 1 \Longrightarrow \lambda_{T_\mu} = \ln\left(\frac{\mu}{2}\right)$$

□

L'exposant de Liapounov de la famille d'applications en tente devient donc strictement positif lorsque $\mu > 2$. Ce changement de signe se ressent clairement dans le diagramme de bifurcation de la figure 1.3, il correspond exactement au moment où les 4 branches naissent. Comme nous allons le voir dans la section suivante, le régime chaotique n'est formellement atteint que lorsque $\mu = 4$ et donc la condition de positivité d'un exposant de Liapounov est nécessaire mais pas suffisante au chaos.

1.6 Chaos

Si l'on se réfère à un dictionnaire, le « chaos » se rapporte à un état de confusion général. Cependant, ce que nous appellerons « chaos » ne sera pas si confus ou aléatoire.

Définition 1.8 (Application chaotique)**.**

Soit $A \subset \mathbb{R}$ un intervalle et soit $f : A \to A$, alors f est chaotique si et seulement si :

(1) f a une dépendance sensible aux conditions initiales :

$$\begin{aligned} &\exists \delta > 0 \text{ tel que } \forall x \in A : \forall U \subset A \text{ tel que } x \in U : \\ &\exists y \in U \text{ et } \exists n \in \mathbb{N} \text{ tels que } \left|f^{\circ n}(x) - f^{\circ n}(y)\right| \geq \delta \end{aligned} \tag{1.7}$$

(2) f est topologiquement transitive :

$$\forall U_1, U_2 \subset A : \exists n \in \mathbb{N} \text{ tel que } f^{\circ n}(U_1) \cap U_2 \neq \varnothing \tag{1.8}$$

(3) f admet des points périodiques denses dans A

Essayons de comprendre ce que cela signifie :

(1) La première condition nous parle clairement de l'imprévisibilité du système bien que l'application f soit parfaitement connue.

(2) La seconde, quant à elle, peut paraître plus obscure. Celle-ci exprime le fait que le système ne pourra pas être décomposé en deux ouverts non-vide disjoints et invariant sous f. En d'autres termes, le système admet une orbite dense dans A (voir [5], chapitre 2, proposition 7).

(3) Finalement, la dernière condition traduit l'existence d'une certaine régularité derrière le comportement apparemment aléatoire d'un système chaotique.

Résultat 1.2 (T_4 est chaotique)**.**

Soit T_μ la famille d'applications en tente définie par (1.1), T_4 est chaotique.

Preuve

Afin d'alléger les notations de cette preuve, nous notons $T \overset{not}{=} T_4$.

Commençons par noter $\circledast$ l'affirmation évidente suivante :
Soit $n \in \mathbb{N}_0$, alors $T^{\circ n}$ envoie chaque intervalle $\left[\frac{k}{2^n}, \frac{k+1}{2^n}\right]$ sur I tout entier $\forall k \in K \overset{def}{=} \{0, 1, \cdots, 2^n - 1\}$.

Ce résultat préliminaire établit, vérifions que T vérifie les trois conditions de la définition 1.8 :

(1) **Dépendance sensible aux conditions initiales** :
Soit $x \in I$ et U un voisinage de x. Pour n suffisamment grand : $\exists k \in K$ tel que $x \in \left[\frac{k}{2^n}, \frac{k+1}{2^n}\right] \subset U$. Dès lors, $\circledast$ implique que $T^{\circ n}(U)$ est envoyée sur tout I, d'où l'existence d'un point $y \in U$ tel que $|T^{\circ n}(x) - T^{\circ n}(y)| \geq \frac{1}{2} \overset{def}{=} \delta$

(2) **Topologiquement transitive** :
Soit U_1 et U_2, deux intervalles ouverts de I. Appliquons le même raisonnement que précédemment, pour n suffisamment grand : $\exists k \in K$ tel que $\left[\frac{k}{2^n}, \frac{k+1}{2^n}\right] \subset U_1$ Dès lors, $\circledast$ implique que $T^{\circ n}(U_1)$ est envoyée sur tout I d'où $T^{\circ n}(U_1) \cap U_2 \neq \varnothing$.

(3) **Points périodiques denses** :
Par $\circledast$, il existe au moins un point périodique de période n dans chacun des intervalles $\left[\frac{k}{2^n}, \frac{k+1}{2^n}\right]$ et, à la limite où n tend vers l'infini, l'ensemble des points fixes devient dense dans I.

□

A l'issue de ce résultat, on s'aperçoit finalement que l'affirmation que l'on avait notée $\circledast$ se retrouve à la base des trois propriétés définissant le chaos. Ceci nous amène à introduire la notion d'application aux itérations tortueuses (*wiggly iterates*). Derrière ce doux nom se cache la définition suivante.

Définition 1.9 (Itérations tortueuses).

Une application $f : I \to I$ a des itérations tortueuses si et seulement si :

(1) $\forall n \in \mathbb{N}_0$: $f^{\circ n}$ est 2^{n-1} – modale.
Si on note $m \overset{not}{=} 2^{n-1}$, ceci signifie qu'il existe $m+1$ points : $0 = x_0 < ... < x_m = 1$ tels que f soit unimodale sur tous les intervalles de base $[x_i, x_{i+1}]$.

(2) La longueur du plus grand intervalle de base s'annule quand n tend vers l'infini :

$$m = 2^{n-1} \to \infty \Longrightarrow \max_{i \in \{0,\cdots,m-1\}} |x_{i+1} - x_i| \to 0$$

On peut donc terminer ce chapitre apéritif sur ce théorème général pour le chaos.

Théorème 1.3 (Itérations tortueuses et chaos).

Soit $f : I \to I$, une application aux itérations tortueuses, alors f est chaotique.

Preuve

Nous montrons qu'une application aux itérations tortueuses vérifie l'affirmation suivante, une généralisation de $\circledast$: pour tout intervalle $U \subset I$: $\exists n \in \mathbb{N}$ tel que $f^{\circ n}$ envoie U sur I tout entier.

La thèse résulte alors de ce qui a été fait dans la preuve du résultat 1.2.

Soit U, un intervalle dans I. Puisque f a des itérations tortueuses, on peut choisir un $\tilde{n}$ suffisamment grand de sorte que la distance entre deux zéros consécutifs de $f^{\circ \tilde{n}}$ soit majorée par la moitié de la longueur de U.
Alors, $[x_i, x_{i+1}] \subset U$ où x_i est le plus petit zéro de $f^{\circ n}$ dans U et x_{i+1} est le suivant.
Prendre $n = \tilde{n} + 1$ convient puisque f est unimodale sur $[x_i, x_{i+1}]$. □

1.7 Exemple : système de Mackey - Glass

Dans cette dernière section, nous donnons la définition d'un système que nous exploiterons dans la suite : le système de Mackey - Glass. Bien entendu, l'objectif ici n'est pas d'en faire l'étude complète mais simplement d'en donner les équations utiles pour la suite. Le but que nous visons, rappelons-le, est plutôt d'étudier un autre système non-linéaire, le *reservoir computer*, qui tentera d'imiter celui que nous présentons ici (voir section 4.4).
Le système de Mackey - Glass est gouverné par une équation différentielle non-linéaire à retard [2] :

$$\dot{y}(t) = -\gamma y(t) + \beta \frac{y_\tau(t)}{1 + y_\tau^n(t)}, \ \beta, \gamma, n > 0 \ , \tag{1.9}$$

où l'on a introduit la notation pointée pour la dérivée temporelle et où l'on a posé $y_\tau(t) = y(t-\tau)$ avec le retard $\tau \in \mathbb{R}_0^+$.
Qualitativement, cette équation décrit un système dans lequel la variable y, considérée sans unité, décroît au taux $\gamma[s^{-1}]$ et se reproduit au taux $\beta[s^{-1}]$. Notons que la production est en retard de $\tau[s]$ par rapport à la perte. En effet, l'équation de Mackey - Glass a été conçue pour décrire la concentration de cellules sanguines dont les taux de production et de perte ne sont pas nécessairement constants. Finalement, le paramètre sans unité n va influencer le comportement asymptotique de $y(t)$ (convergence, divergence et oscillations amorties ou non) pour autant que γ ne soit pas trop grand devant β car sinon le système tend plutôt à s'éteindre.

Afin de pouvoir intégrer numériquement (1.9), il faudra se donner une **histoire** $\mathbb{S}$, c'est-à-dire une infinité de conditions initiales :

$$\mathbb{S} = \{y(t) \mid t \in [-\tau, 0]\} \ ,$$

et fixer les paramètres (β, γ, n, τ), donnés dans « l'ordre alphabétique » et dans les unités explicitées plus haut. On utilisera $(0.2, 0.1, 10, 17)$ dans la suite et, sauf mention explicite, une histoire constante $\mathbb{S} \equiv 0.5$ sera considérée pour l'intégration (voir annexe B.3) de l'équation de Mackey - Glass. La figure 1.4 nous illustre ce choix de paramètres.

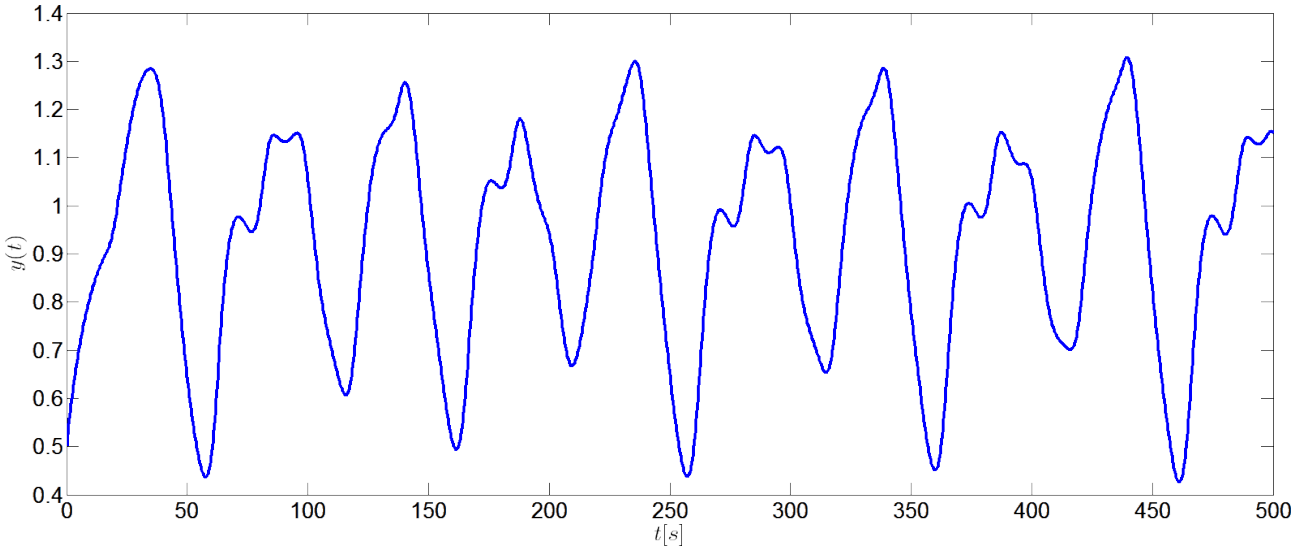


FIGURE 1.4 – Dynamique du système de Mackey - Glass paramétré avec $(0.2, 0.1, 10, 17)$ où $\mathbb{S} \equiv 0.5$.

Ce dernier a été fait afin de donner lieu à un comportement chaotique[3]. Pour s'en convaincre, observons l'attracteur donné à la figure 1.5. Cet attracteur nous présente $y(t-\tau)$ comme « fonction » de $y(t)$. On s'aperçoit alors du nombre considérable de valeurs passées de y donnant lieu à sa valeur actuelle et donc de la complexité de la dynamique.

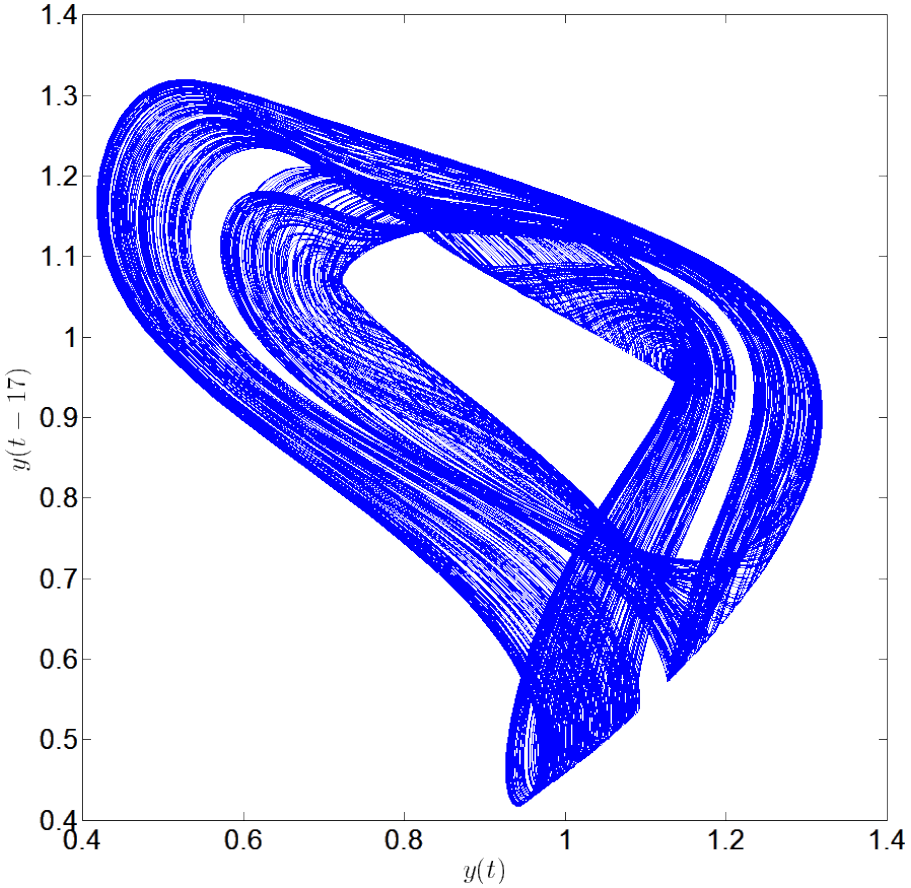


FIGURE 1.5 – Attracteur du système de Mackey - Glass paramétré avec $(0.2, 0.1, 10, 17)$.

Si le lecteur souhaite aller plus loin dans l'étude de ce système, nous lui recommandons de consulter [2] pour commencer et de poursuivre avec les références qui s'y trouvent.

3. Bien que la dépendance aux conditions initiales soit manifeste (voir figure 4.11), le caractère chaotique n'est pas encore tout à fait reconnu au système de Mackey - Glass [2] : on devrait plutôt parler de comportement complexe. Nous conserverons tout de même le terme « chaos » pour ne pas *complexifier* la lecture.

Chapitre 2

Introduction à la cryptographie par chaos

Ce nouveau chapitre présente deux méthodes de communication basées sur le chaos :

▷ Superposition du signal chaotique et du message ;

▷ Mélange non-linéaire du signal chaotique et du message.

La première s'inspire de ce qui est proposé dans [7] et la seconde se base sur [8].

2.1 Généralités

La cryptographie par chaos est caractérisée par l'utilisation de signaux extrêmement sensibles aux conditions initiales, que l'on qualifie de *chaotiques*, pour porter un message. C'est sur ce point que repose la sécurité de ce genre de méthodes. En effet, si le système transmetteur arbore un comportement chaotique, alors il est difficile de reproduire sa dynamique et, par conséquent, il sera difficile de récupérer le message. Nous attirons l'attention du lecteur sur le fait que cette branche de la cryptographie est très récente. En effet, celle-ci n'est apparue qu'après 1990, lorsque PECORA et CARROL ont découvert qu'il était possible de synchroniser des systèmes chaotiques [7].

On ne peut pas faire de cryptographie sans nos collaborateurs préférés : Alice et Bob. Tandis qu'Alice va crypter un message pour le transmettre à Bob, ce dernier, quant à lui, devra récupérer le message d'Alice à partir du signal crypté. Bien entendu, Bob dispose d'une copie du système d'Alice et il sait comment elle a crypté le message : il utilisera cette information pour générer son propre signal chaotique qui lui permettra de récupérer le message. De manière générale on considérera la situation de la figure 2.1 dont la terminologie est donnée ci-après.

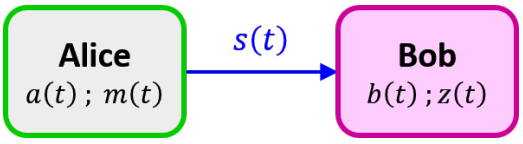


FIGURE 2.1 – Schéma général de communication entre Alice et Bob.

▷ $a(t)$ est le signal chaotique d'Alice ;

▷ $m(t)$ est le message caché que Bob doit décrypter ;

▷ $s(t)$ est le signal transmis par Alice ;

▷ $b(t)$ est le signal chaotique de Bob ;

▷ $z(t)$ est le signal de sortie généré par Bob en combinant $s(t)$ et $b(t)$.

Avant de terminer cette section introductive, nous attirons l'attention du lecteur sur deux points. Tout d'abord, le code couleur de la figure 2.1 sera conservé dans la suite : nous garderons le vert pour Alice et le magenta pour Bob.
Ensuite, nous insistons sur le fait que ce travail a entièrement été réalisé numériquement. C'est pourquoi nous pointerons du doigt les problèmes qui peuvent apparaître lorsque l'on sort du cadre des simulations (ex. : bruits expérimentaux) sans pour autant les traiter en détail ici.

2.2 Superposition du signal chaotique et du message

La première méthode utilise ce que nous appelons le verrouillage. Nous commençons par expliciter cette notion (§ 2.2.1) avant de l'utiliser dans la communication par superposition (§ 2.2.2).

2.2.1 Verrouillage

Considérons deux copies identiques et indépendantes $\mathcal{S}_1$ et $\mathcal{S}_2$ d'un système dynamique retardé dont on peut décrire l'évolution en temps discret par les orbites suivantes (voir définition 1.3) :

$$\mathcal{S}_i(t+1) = f\left[t,\ \mathcal{S}_i(t),\ \mathcal{S}_i(t-\tau)\right], \quad i = 1, 2 \tag{2.1}$$

où τ est le retard. $\mathcal{S}_1$ sera le système **primaire** tandis que $\mathcal{S}_2$ sera qualifié de **secondaire**. Construisons un nouveau système en perturbant $\mathcal{S}_2$ par $\mathcal{S}_1$:

$$\tilde{\mathcal{S}}_2(t+1) = pf\left[t,\ \tilde{\mathcal{S}}_2(t),\ \tilde{\mathcal{S}}_2(t-\tau)\right] + q\mathcal{S}_1(t+1) \tag{2.2}$$

où p et q sont des réels positifs vérifiant $p+q=1$. Pour $q \in]0\,,1[$ la proportion de $\mathcal{S}_1$ va prendre le dessus sur la dynamique de (2.2) et, au fil du temps, le système $\tilde{\mathcal{S}}_2$ **se verrouille** sur $\mathcal{S}_1$. En effet, comme le premier terme dépend bien de $\tilde{\mathcal{S}}_2$ et non pas de $\mathcal{S}_2$, à chaque pas de temps la dynamique du système $\tilde{\mathcal{S}}_2$ va se rapprocher de celle du système primaire. Le verrouillage est atteint lorsque $\tilde{\mathcal{S}}_2$ et $\mathcal{S}_1$ sont complètement synchronisés :

$$\tilde{\mathcal{S}}_2(t) = \mathcal{S}_1(t), \forall t \geq T_V \tag{2.3}$$

où T_V est le **temps du verrouillage**. Dès cet instant, en utilisant (2.3) et $p = 1-q$, (2.2) devient :

$$\tilde{\mathcal{S}}_2(t+1) = pf\left[t,\ \tilde{\mathcal{S}}_2(t),\ \tilde{\mathcal{S}}_2(t-\tau)\right] + q\tilde{\mathcal{S}}_2(t+1) \Longrightarrow \tilde{\mathcal{S}}_2(t+1) = f\left[t,\ \tilde{\mathcal{S}}_2(t),\ \tilde{\mathcal{S}}_2(t-\tau)\right]$$

et on a une équation similaire à (2.1) pour décrire l'évolution de $\tilde{\mathcal{S}}_2$: le verrouillage n'est plus nécessaire et $\tilde{\mathcal{S}}_2$ peut être découplé de $\mathcal{S}_1$ tout en conservant la dynamique de ce dernier.

Revenons sur le paramètre q. Celui-ci quantifie l'**efficacité du verrouillage**, en effet :

- ▷ $q = 0 \Longrightarrow \tilde{\mathcal{S}}_2 \equiv \mathcal{S}_2$: pas de verrouillage ;
- ▷ $q = 1 \Longrightarrow \tilde{\mathcal{S}}_2 \equiv \mathcal{S}_1$: verrouillage. instantané

La figure 2.2 illustre la situation où deux systèmes de Mackey - Glass sont considérés. De nouveau, on a utilisé $(0.2, 0.1, 10, 17)$ pour intégrer numériquement l'équation (1.9). Pour le primaire, on a

conservé $\mathbb{S} \equiv 0.5$ pour l'histoire tandis que pour le secondaire on a considéré $\mathbb{S} \equiv 0.1$. Si on note :

$$\mathcal{R}(t) \stackrel{def}{=} \log \left| \frac{\tilde{\mathcal{S}}_2(t) - \mathcal{S}_1(t)}{\mathcal{S}_1(t)} \right|,$$

le logarithme en base 10 de l'erreur relative entre $\tilde{\mathcal{S}}_2$ et $\mathcal{S}_1$, on observe sur la figure une décroissance (assez lente avec une efficacité de seulement $q = 0.25$) de $\mathcal{R}$ synonyme du verrouillage du secondaire sur le primaire. Au-delà de $T_V = 820$ s, les systèmes sont verrouillés à la précision machine [1] $\sim 10^{-16}$.

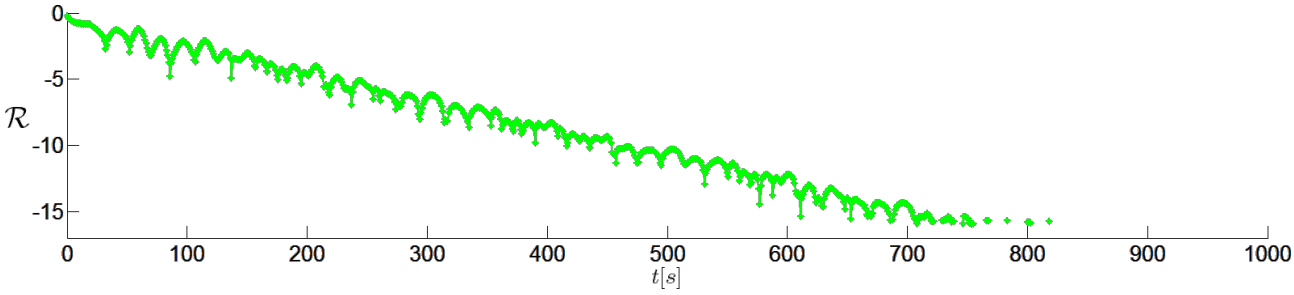


FIGURE 2.2 – Évolution de $\mathcal{R}$ montrant le verrouillage de deux Mackey - Glass avec $q = 0.25$.

Notons que le verrouillage se fera plus rapidement avec de plus grandes valeurs de q. Pour s'en convaincre, nous avons repris ci-dessous la situation de la figure 2.2 en considérant $q = 0.5$.

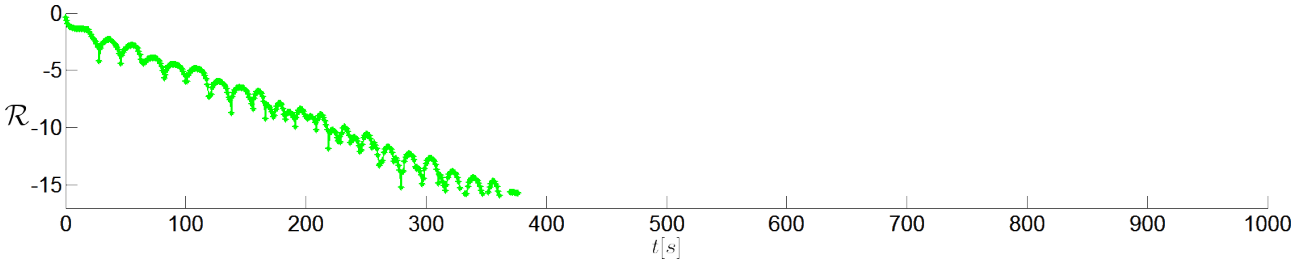


FIGURE 2.3 – Évolution de $\mathcal{R}$ montrant le verrouillage de deux Mackey - Glass avec $q = 0.5$.

2.2.2 Transmission d'un message

Soit $a(t)$ le signal chaotique du système d'Alice et $m(t)$ le message à transmettre. La méthode la plus naïve pour envoyer l'information est d'émettre $s(t)$ tel que :

$$s(t) = a(t) + m(t), \tag{2.4}$$

où la sécurité repose sur le caractère chaotique de $a(t)$ et sous l'hypothèse que $\forall t : |m(t)| \ll |a(t)|$. Compte tenu de (2.4), la dynamique du système de Bob doit être identique à celle du système d'Alice, c'est-à-dire $\forall t : b(t) = a(t)$. Bob obtiendra alors le message en sortie :

$$z(t) \stackrel{def}{=} s(t) - b(t) = a(t) + m(t) - a(t) = m(t)$$

1. Numériquement on a bien l'égalité (2.3). Cependant, on peut imaginer que sur de vrais systèmes physiques il reste une erreur $\varepsilon > 0$ de sorte que (2.3) devienne : $\left|\tilde{\mathcal{S}}_2(t) - \mathcal{S}_1(t)\right| < \varepsilon, \forall t \geq T_V$. Nous reviendrons sur ce point au paragraphe 2.2.5.

L'idée sera donc de demander à Alice de commencer à envoyer son signal chaotique sans message pendant un court instant noté T_P, c'est-à-dire d'envoyer $s(t) = a(t)$ jusque T_P fixé initialement par Alice et Bob. Pendant ce laps de temps, Bob devra verrouiller son système sur celui d'Alice. Il est clair que l'efficacité q devra être suffisamment grande de sorte que l'on aura bien $T_V \leq T_P$ sinon le verrouillage n'aura pas le temps de se terminer. Une fois les dynamiques de a et b synchronisées, le système de Bob (qui est une copie parfaite de celui d'Alice) ne quittera plus son orbite et reproduira à l'identique le signal chaotique d'Alice $a(t)$: on découple les systèmes en coupant le verrouillage, c'est-à-dire qu'on pose $q = 0$. Alice peut superposer le message et envoyer $s(t) = a(t) + m(t)$. Enfin, Bob effectue la différence $s(t) - b(t)$ énoncée ci-dessus et récupère $m(t)$ à la sortie.

2.2.3 Protocole pour la transmission superposée

Un schéma représentant cette méthode de communication est donné à la figure 2.4. Pour terminer, nous synthétisons cette méthode de communication de manière algorithmique de sorte que la transmission d'un message par superposition avec un signal chaotique suive le protocole suivant :

Protocole 2.1 (Transmission par superposition)**.**

(1) **Alice envoie son signal chaotique pur** : Alice émet $s(t) = a(t)$ jusque l'instant T_P fixé au préalable par Alice et Bob ;

(2) **Verrouillage de Bob sur Alice** : Bob choisit $q \in]0\,,1[$ tel que $T_V \leq T_P$. Avec cette efficacité, il verrouille son système sur celui d'Alice de sorte que $\forall t \geq T_V$: $b(t) = a(t)$;

(3) **Arrêt du verrouillage et superposition du message** : Bob pose $q = 0$ et il laisse son système évoluer librement sur la même orbite que celle d'Alice pendant qu'elle envoie le signal $s(t) = a(t) + m(t)$;

(4) **Récupération du message par Bob** : Bob décrypte le message en sortie en effectuant la différence $z(t) = s(t) - b(t) = m(t)$.

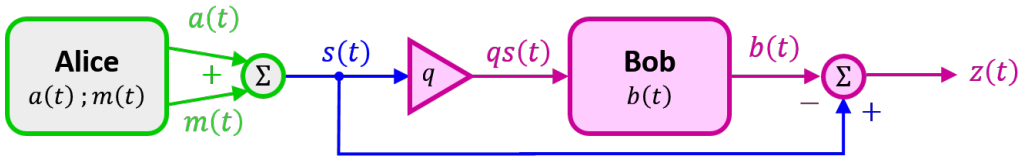


FIGURE 2.4 – Schéma pour la communication chaotique par superposition. Le message m et l'efficacité q ne sont pas toujours présent (voir le protocole 2.1).

Au chapitre 5, nous remplacerons le système de Bob par un *reservoir computer* qui sera entraîné afin d'émuler la dynamique du système de Mackey - Glass. Enfin, on le verrouillera sur le système d'Alice pour pouvoir appliquer le protocole 2.1 et, si on y parvient, on pourra également récupérer le message.

Notons au passage qu'un espion disposant d'un réservoir et d'une connaissance suffisante pour l'entraîner de la même manière pourrait intercepter la communication et lire, lui-aussi, le message !

2.2.4 Résultats

Cette méthode a été testée numériquement en considérant, encore une fois, deux Mackey - Glass $(0.2, 0.1, 10, 17)$ pour les systèmes d'Alice et Bob. Pour pouvoir diminuer T_P, nous avons choisi de prendre une grande efficacité pour le verrouillage : $q = 0.95$. Comme nous l'avons déjà signalé, cette façon de procéder est très naïve. En effet, comme le montre le paragraphe suivant, elle est extrêmement sensible au bruit. Quoi qu'il en soit, lors de tests numériques, on peut travailler sans bruit et le message est parfaitement récupéré par Bob.

Les résultats que l'on donne ici ont été obtenus en prenant $m(t)$ comme une suite aléatoire de bits 0 ou 1 d'amplitude 10^{-15} mais nous précisons qu'aucune hypothèse n'a été faite dans notre méthode sur la forme ou la longueur de $m(t)$: cette récupération parfaite et inconditionnelle découle directement de la réussite du verrouillage et donc de l'absence *totale* de bruit.

2.2.5 Sensibilité au bruit

Dans cette section nous essayons de mettre en évidence l'influence d'un bruit sur la méthode. Soit $\nu(t)$ du bruit d'amplitude A_ν de sorte que le signal reçu par Bob n'est plus $s(t)$ mais plutôt la somme $\tilde{s}(t) = s(t) + \nu(t)$. On parle alors d'un bruit externe additif car il est simplement ajouté au signal $s(t)$ et qu'il ne modifie en rien sa dynamique. Pour tenir compte de cette modification, on ajoute un terme dans l'équation (2.2) pour le verrouillage :

$$\tilde{\mathcal{S}}_2(t+1) = pf\left[t,\ \tilde{\mathcal{S}}_2(t),\ \tilde{\mathcal{S}}_2(t-\tau)\right] + q\mathcal{S}_1(t+1) + \nu(t) \tag{2.5}$$

La présence de ce terme supplémentaire nuit au verrouillage dont la précision sera au mieux de l'ordre de A_ν :

$$\left|\tilde{\mathcal{S}}_2(t) - \mathcal{S}_1(t)\right| \leq A_\nu, \forall t \geq T_V$$

La figure 2.5 illustre cette nouvelle situation où l'on a considéré du bruit généré uniformément avec une amplitude $A_\nu = 10^{-10}$.

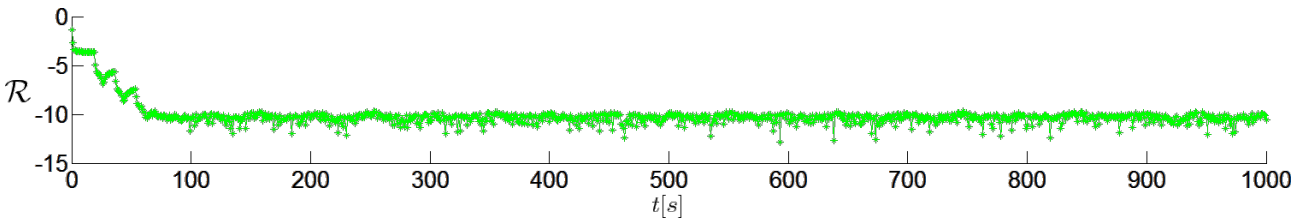


FIGURE 2.5 – Évolution de l'erreur relative $\mathcal{R}$ montrant le verrouillage des Mackey - Glass considérés à la figure 2.2 avec, ici, une efficacité $q = 0.95$ et du bruit uniforme d'amplitude $A_\nu = 10^{-10}$.

Reprenons le test précédent où, cette fois, $m(t)$ est une suite aléatoire de bits 0 ou 1 d'amplitude $A = 10^{-5}$. Ajoutons du bruit généré uniformément avec une amplitude $A_\nu = 10^{-10}$ lors de la transmission et appliquons le protocole. Il vient $\tilde{s}(t) = a(t) + \nu(t)$ jusque T_P puis $\tilde{s}(t) = a(t) + m(t) + \nu(t)$.

Lorsque l'on arrête le verrouillage, il subsiste une différence de l'ordre de A_ν entre les signaux $a(t)$ et $b(t)$. Les systèmes d'Alice et Bob étant chaotiques, il s'en suit l'évolution exponentielle de cette différence rendue visible à la figure 2.6.

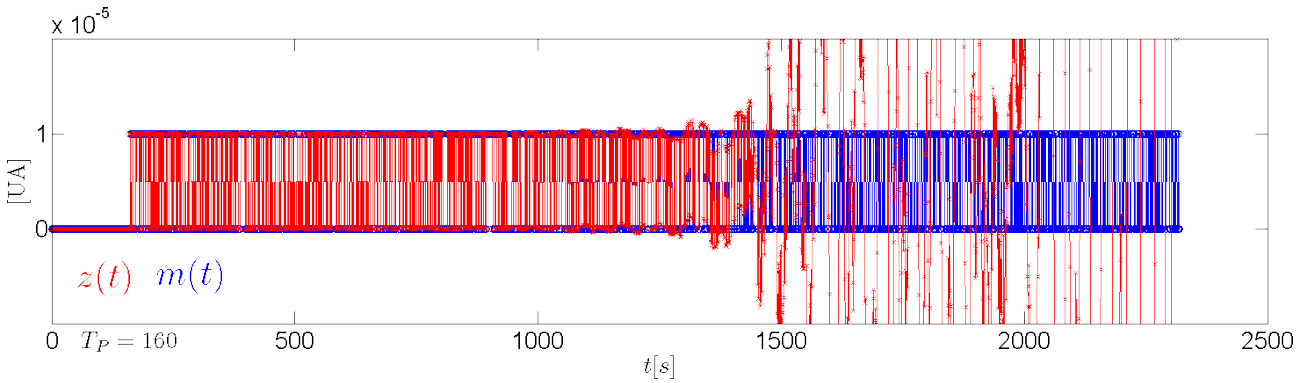


FIGURE 2.6 – Influence d'un bruit uniforme d'amplitude $A_\nu = 10^{-10}$ pour la transmission d'un message par superposition : $m(t)$ en bleu et $z(t)$ en rouge.

Les cinq ordres de grandeurs entre A et A_ν permettent de retrouver $m(t)$ au début comme s'il n'y avait aucun bruit car l'erreur due à ce dernier n'est pas encore trop importante. Celle-ci croît et finit par devenir suffisamment grande au voisinage de $t = 1500$ s : $a(t)$ et $b(t)$ sont désynchronisés et le message devient irrécupérable pour Bob.

Pour pallier ce problème, Alice peut supprimer régulièrement son message afin de permettre à Bob de se verrouiller à nouveau. Ainsi, Bob parviendrait à maintenir une synchronisation suffisante de son système et donc à communiquer plus longuement, même en présence de bruit. Bien entendu, ils doivent se mettre d'accord à l'avance de sorte que Bob sache quand le message est présent ou non et donc qu'il sache quand il doit activer ou désactiver le verrouillage.

2.3 Mélange non-linéaire du signal chaotique et du message

Dans cette section nous présentons la seconde méthode de communication basée sur le chaos que nous avions annoncée : le mélange non-linéaire. Celle-ci se démarque de la transmission par superposition par l'absence de verrouillage entre les systèmes d'Alice et Bob. En effet, le signal envoyé par Alice sera constamment utilisé pour générer celui de Bob et, à partir de la différence entre les deux, nous verrons comment retrouver le message.

2.3.1 Systèmes à retard

Considérons un système à retard régit par l'équation différentielle retardée suivante :

$$\varepsilon_0 \dot{x}(t) = -x(t) + f\left[x(t-\tau_0)\right] \tag{2.6}$$

Celui-ci se ramène aisément au système de Mackey - Glass décrit par (1.9) en prenant $\varepsilon_0 = 1/\gamma$ et

$$f\left[x(t-\tau_0)\right] = \frac{\beta}{\gamma} \frac{x(t-\tau_0)}{1+x^n(t-\tau_0)}$$

Cette nouvelle écriture est propice à la représentation schématique du système à retard donnée à la figure 2.7.

Les points marqués par des chiffres romains ou arabes sur la figure 2.7 indiquent les nœuds où un signal peut être ajouté ou envoyé, respectivement. Si on ajoute le signal $d(t)$ on aura trois équations possibles pour décrire la nouvelle dynamique :

$$\begin{cases} \varepsilon_0 \dot{x}(t) = -x(t) + f\left[x(t-\tau_0) + d(t-\tau_0)\right] & \text{(2.7a)} \\ \varepsilon_0 \dot{x}(t) = -x(t) + f\left[x(t-\tau_0) + d(t)\right] & \text{(2.7b)} \\ \varepsilon_0 \dot{x}(t) = -x(t) + f\left[x(t-\tau_0)\right] + d(t) & \text{(2.7c)} \end{cases}$$

Les équations (2.7) correspondent, dans l'ordre, à l'injection du message aux entrées I à III.

Les points identifiés par 1, 2 ou 3 correspondent aux nœuds où le signal parcourant ce dernier, noté $s(t)$, est émis. Notons que dans les trois cas où les nœuds d'injection et d'émission sont identiques, on ajoute $d(t)$ *avant* d'émettre. On voit apparaître neuf configurations possibles que l'on notera $\mathcal{I}/i$ où $\mathcal{I} \in \{I, II, III\}$ et $i \in \{1, 2, 3\}$.

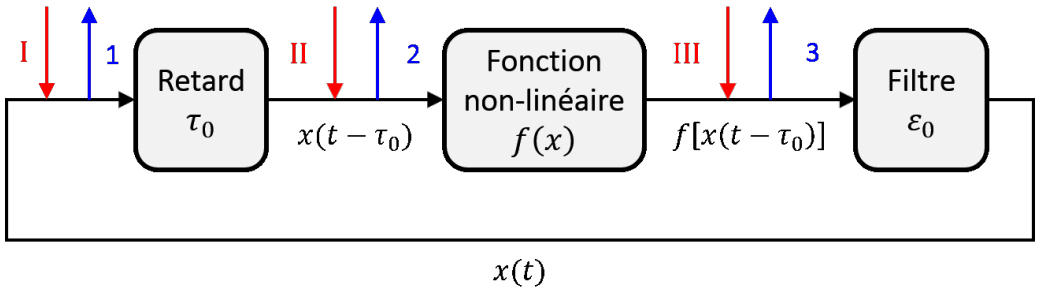


FIGURE 2.7 – Schéma d'un système non-linéaire à retard.

2.3.2 Transmission d'un message

Voyons comment exploiter ce système à retard pour permettre à Alice et Bob de communiquer de manière sécurisée. Soit $\mathcal{I}/i$ la configuration pour le système d'Alice. Considérons une copie de ce système que l'on « ouvre » en i pour Bob. Ce dernier va recevoir $s(t)$ en i et, une fois que ce signal aura parcouru tout le système de Bob au départ de i, on le notera $s'(t)$. Pour fixer les idées, prenons le cas $III/1$ illustré à la figure 2.8 et dont les équations d'évolution sont :

$$\begin{cases} \varepsilon_0 \dot{a}(t) = -a(t) + f\left[a(t-\tau_0)\right] + m(t) & \text{(2.8a)} \\ \varepsilon_0 \dot{b}(t) = -b(t) + f\left[a(t-\tau_0)\right] & \text{(2.8b)} \end{cases}$$

La première est donnée par (2.7c) avec $d \equiv m$ puisque $\mathcal{I} = III$ et la seconde est adaptée, compte tenu de la configuration $III/1$, de (2.6).

Le signal $s(t)$ reçu par Bob est dupliqué avant d'entrer dans son système afin de pouvoir construire le signal de sortie $z(t)$ comme la différence entre le signal $s(t)$ émis par Alice en $i = 1$ et le signal

$s'(t)$ généré par le système de Bob après que $s(t)$ l'ait parcouru entièrement, c'est-à-dire :

$$z(t) = s(t) - s'(t) \overset{III/1}{\Longrightarrow} z(t) = a(t) - b(t),$$

où le message n'apparaît plus explicitement car il est contenu dans $a(t)$ et dans $b(t)$, d'où le terme de « mélange non-linéaire ». Bob connait complètement le système couplé (2.8) et s'il effectue la différence (2.8a) − (2.8b), il trouve l'expression suivante pour le message :

$$\varepsilon_0[\dot{a}(t) - \dot{b}(t)] = -a(t) + f[a(t-\tau_0)] + m(t) + b(t) - f[a(t-\tau_0)] \Longrightarrow m(t) = \varepsilon_0 \dot{z}(t) + z(t)$$

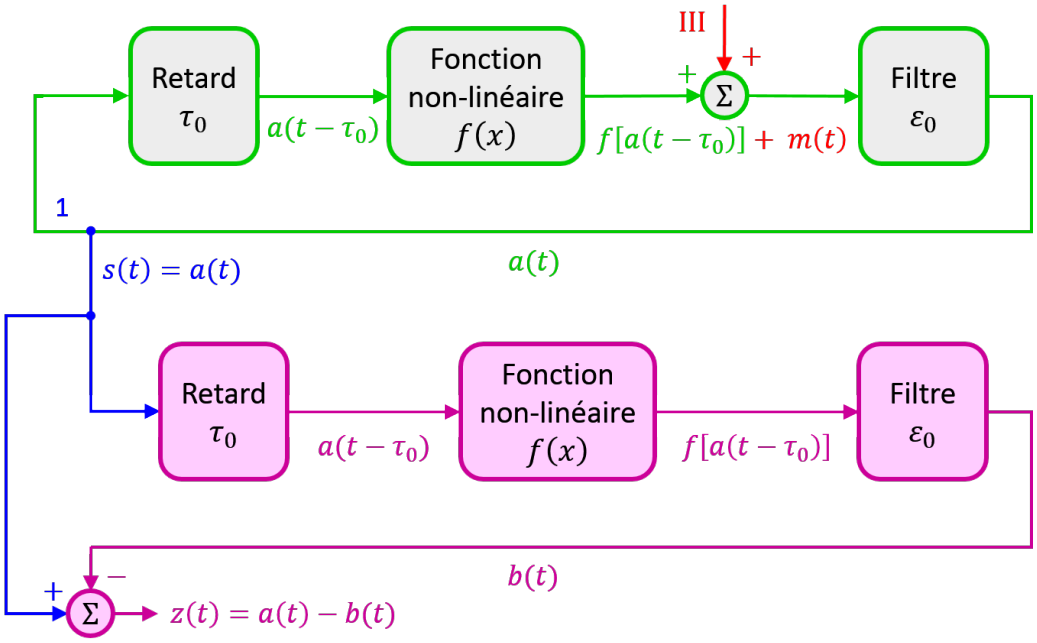


FIGURE 2.8 – Configuration $III/1$ pour la communication chaotique.

2.3.3 Protocole pour la transmission par mélange non-linéaire

En répétant le raisonnement du paragraphe précédent pour les autres configurations, on parvient à compléter le tableau 2.1 avec les expressions du signal obtenu en sortie $z(t)$.

	1	**2**	**3**
I	$m(t)$	$m(t-\tau_0)$	$f\left[a(t-\tau_0) + m(t-\tau_0)\right]$ $-f\left[a(t-\tau_0)\right]$
II	$a(t) - b(t) = -\varepsilon_0 \dot{z}(t)$ $+f\left[a(t-\tau_0) + m(t)\right] - f\left[a(t-\tau_0)\right]$	$m(t)$	$f\left[a(t-\tau_0) + m(t)\right]$ $-f\left[a(t-\tau_0)\right]$
III	$a(t) - b(t) = -\varepsilon_0 \dot{z}(t) + m(t)$	$a(t-\tau_0) - b(t-\tau_0)$	$m(t)$

TABLEAU 2.1 – Signal de sortie $z(t)$ dans les différentes configurations possibles

Comme on a supposé que $\forall t : m(t) \ll a(t)$, on peut utiliser un développement en série de Taylor pour approximer les termes de la forme suivante :

$$f[a(t) + m(t)] - f[a(t)] \approx m(t)\frac{df[a(t)]}{da}$$

Ceci introduira une légère distorsion dans la récupération du message dans les cas $I/3$, $II/1$ et $II/3$. L'expression de ce dernier est donnée au tableau 2.2 en fonction de $z(t)$ et de $\dot{z}(t)$ dans les neuf configurations.

	1	**2**	**3**
I	$z(t)$	$z(t+\tau_0)$	$z(t+\tau_0)\,(df[a(t)]/da)^{-1}$
II	$(z(t)+\varepsilon_0\dot{z}(t))\,(df[a(t-\tau_0)]/da)^{-1}$	$z(t)$	$z(t)\,(df[a(t-\tau_0)]/da)^{-1}$
III	$\varepsilon_0\dot{z}(t)+z(t)$	$\varepsilon_0\dot{z}(t-\tau_0)+z(t-\tau_0)$	$z(t)$

TABLEAU 2.2 – Expression du message $m(t)$ chez Bob dans les différentes configurations possibles.

Pour plus de détails sur ces tableaux le lecteur peut consulter [8]. Cependant nous lui recommandons d'être prudent à l'égard de toutes ces expressions. En effet, celle obtenue ci-dessus pour le message dans la configuration $III/1$ contenait une faute de signe dans l'article sus-mentionné et nous n'avons pas pris la peine de vérifier dans le détail toutes les autres car nous ne les utilisons pas ici. Néanmoins nous les avons tout de même mentionnées afin de montrer au lecteur comment le signal de sortie peut dépendre du message et donc comment le message est *mélangé* dans les signaux $a(t)$ et $b(t)$.

De manière général, le protocole à suivre est le suivant :

Protocole 2.2 (Transmission par mélange non-linéaire)**.**

(1) **Configuration** : Alice et Bob choisissent au préalable une configuration $\mathcal{I}/i$;

(2) **Alice injecte le message** : le message est injecté chez Alice en $\mathcal{I}$;

(3) **Alice envoie le signal** : Alice envoie le signal $s(t)$ du nœud i ;

(4) **Activation du système de Bob** : le système de Bob reçoit $s(t)$ en i et l'utilise pour produire $s'(t)$;

(5) **Construction du signal de sortie par Bob** : $z(t) = s(t) - s'(t)$ (tableau 2.1) ;

(6) **Récupération du message par Bob** : $m(t)$ est décrypté (tableau 2.2).

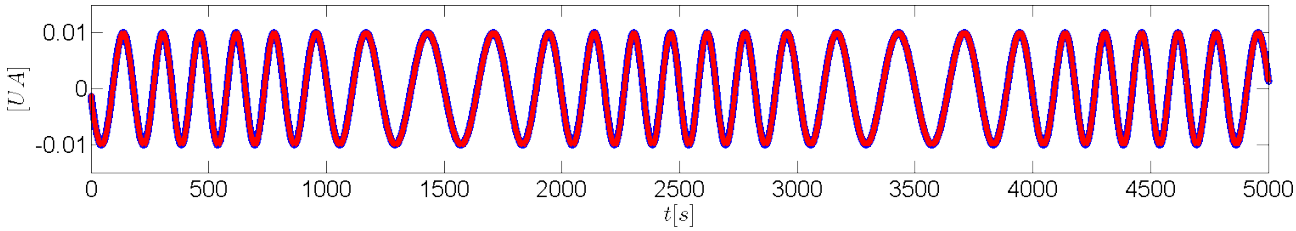
Finalement, si on parvient à entraîner un ordinateur réservoir à produire une bonne approximation de $s'(t)$ à partir de $s(t)$: Bob (ou un espion disposant des connaissances suffisantes pour effectuer sa manœuvre) pourra récupérer le message sans posséder lui-même une copie du système d'Alice ! Nous étudierons ce point au chapitre 5.

2.3.4 Résultats

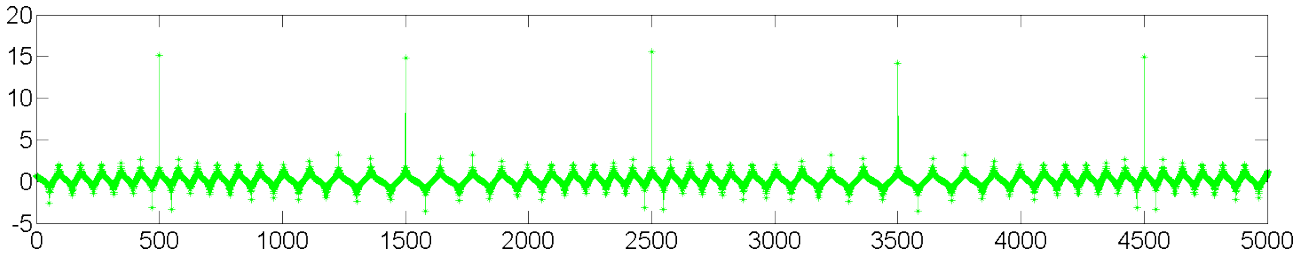
Comme dans [8], nous avons utilisé une fonction sinusoïdale modulée en fréquence comme message :

$$m(t) = A \sin\left[2\pi f_c t - B\cos(2\pi f_m t)\right],$$

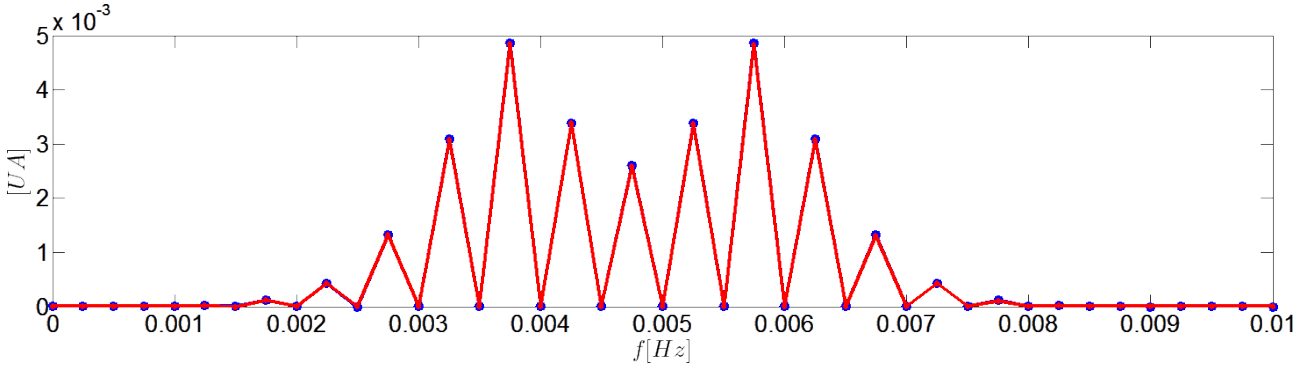
avec $A = 0.01$, $B = 3$, $f_c = 5 \cdot 10^{-3}\ Hz$, $f_m = 5 \cdot 10^{-4}\ Hz$. Pour obtenir les figures 2.9, nous avons choisi la configuration $III/1$. Le protocole 2.2 est appliqué : les signaux $a(t)$ et $b(t)$ sont obtenus par intégration numérique (§ B.3.3) des équations (2.8) et le calcul de $\dot{z}(t)$ a été fait à l'aide de la formule d'Euler (B.1) en considérant un pas unitaire pour la discrétisation du temps. Le message est correctement reçu, comme le montre la figure 2.9a, malgré les pics étranges dans l'erreur relative donnée à la figure 2.9b. Ceux-ci apparaissent lorsque l'erreur absolue est rapportée à 0, ce qui crée une divergence dans l'erreur relative. Finalement, cette dernière ne nous apporte pas vraiment d'information : on doit chercher ailleurs. C'est alors que la figure 2.9c nous montre que le spectre du message est parfaitement transmis. Notons que l'utilisation de la configuration $III/1$ nous a évité de devoir développer en série de Taylor la fonction non-linéaire f comme c'est le cas dans d'autres configurations telles que $II/3$. L'avantage est donc l'absence de distorsion dans le message récupéré qui aurait été due à la « Taylorisation » de f.



(a) Comparaison entre le message original en bleu et le message récupéré en rouge.



(b) Evolution du logarithme en base 10 de l'erreur relative entre le message original et récupéré.



(c) Comparaison des spectres du message original en bleu et récupéré en rouge.

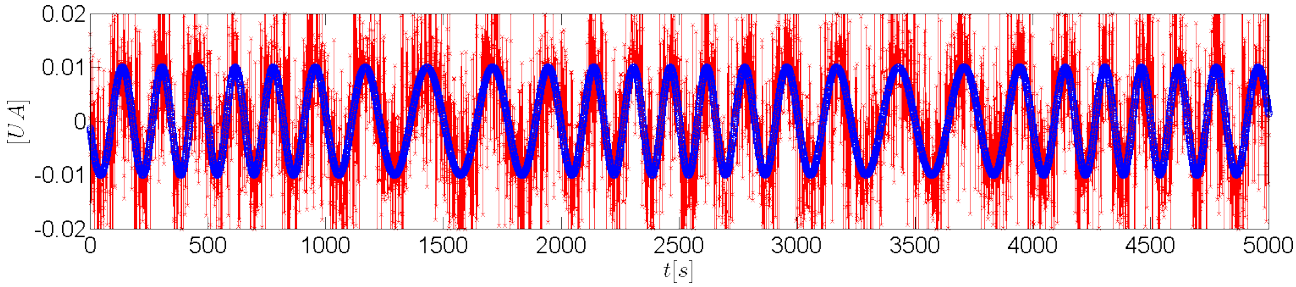
FIGURE 2.9 – Résultats de la communication non-bruitée via le mélange non-linéaire.

2.3.5 Sensibilité au bruit

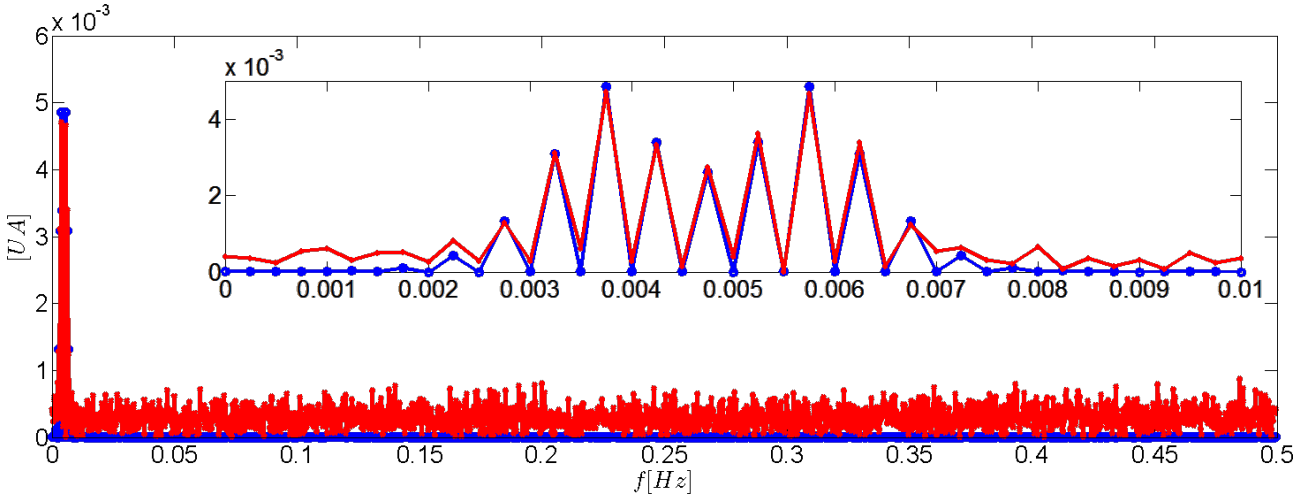
Comme au paragraphe 2.2.5, nous allons ajouter du bruit externe dans la communication $III/1$ pour tester la robustesse de la méthode. Ainsi, plutôt que de recevoir $s(t) = a(t)$, Bob doit injecter dans son système $\tilde{s}(t) = a(t) + \nu(t)$, où $\nu(t)$ est, de nouveau, un bruit uniforme d'amplitude A_ν.

Nous avons porté aux figures 2.10 les résultats obtenus en considérant la même situation qu'au paragraphe précédent mais avec un bruit important d'amplitude $A_\nu = 0.01$. Malgré ce niveau de bruit aussi élevé que celui du message, Bob parvient à récupérer un signal qui arbore une dynamique similaire à celle du message original en restant borné autour de celui-ci (fig. 2.10a).
Finalement, si on compare les spectres des message original et récupéré, on voit dans le zoom de la figure 2.10b que l'on retrouve très bien les pics plus importants. Ceux de plus faible amplitude, quant à eux, sont confondus avec le bruit de fond induit par $\nu(t)$.

C'est donc parce que le message est constitué de fréquences assez basses que le bruit n'impacte pas de manière significative la zone fréquentielle qui le concerne et que le spectre est bien récupéré.



(a) Comparaison entre le message original en bleu et le message décrypté en rouge.



(b) Comparaison des spectres du message original en bleu et décrypté en rouge.

FIGURE 2.10 – Résultats de la communication bruitée via le mélange non-linéaire.

Chapitre 3

Implémentation d'un ordinateur réservoir

Dans ce nouveau chapitre, on introduit différents réseaux de neurones artificiels de manière à amener petit à petit les concepts utiles pour aboutir à l'équation de mise à jour de l'ordinateur réservoir (*reservoir computer*) que l'on pourra implémenter. Une fois ce premier objectif atteint, nous donnons quelques définitions mathématiques pour pouvoir formaliser et comprendre l'état d'écho.

Ce chapitre et le suivant sont consacrés à l'étude de l'ordinateur réservoir, de nombreuses références ont été nécessaires à leur élaboration et le lecteur souhaitant approfondir un point particulier ou simplement curieux est invité à consulter la section Ordinateur réservoir de la bibliographie.

3.1 Réseau de neurones avançant / *Feedforward neural network*

Nous introduisons ce premier type de réseau pour sa simplicité à illustrer les concepts généraux liés aux réseaux de neurones artificiels. Sans plus attendre, voici la première définition :

Définition 3.1 (Feedforward neural network)**.**

Un *feedforward neural network* est un système dynamique où les unités qui le constituent interagissent entre elles et dont la structure **ne** contient **pas** de cycle.

L'absence de boucle est une topologie[1] particulière qui implique le caractère unidirectionnel des signaux parcourant le réseau, d'où le qualificatif « avançant ». La figure 3.1a nous montre un schéma générique de ce type de réseau tout en illustrant la terminologie suivante :

- ▷ Les **états internes** : ces N états forment la **couche cachée**[2] du réseau, nous les notons x ;
- ▷ L'**entrée** : par le biais de ces K variables que la couche cachée reçoit le signal d'entrée u ;
- ▷ La **sortie** : L signaux, notés y, construits en combinant les signaux internes x ;
- ▷ Le **masque d'entrée** : $W_{in} \in \mathbb{R}^{N\times K}$ couplant $u \in \mathbb{R}^K$ à $x \in \mathbb{R}^N$;
- ▷ Les **poids de sortie** : $W_{out} \in \mathbb{R}^{L\times N}$ couplant $x \in \mathbb{R}^N$ à $y \in \mathbb{R}^L$.

Les équations associées à ce réseau sont très simples :

$$\begin{cases} x = \sigma\left(W_{in}u\right) & \text{(3.1a)} \\ y = \varphi\left(W_{out}x\right) & \text{(3.1b)} \end{cases}$$

1. Pour un réseau, la topologie est synonyme de structure ou d'architecture.
2. Nous avons simplifié le système. En effet, cette couche cachée pourrait être constituée de plusieurs couches mais cela introduirait des dépendances en x dans l'équation (3.1a) qui compliqueraient inutilement notre introduction.

où on a les **fonctions d'activation d'entrée et de sortie** σ et φ respectivement. La première de ces équations donne les états internes à partir du signal d'entrée et la seconde génère la sortie du réseau en combinant les états internes. Notons que ce système ne fait pas de mise à jour : il reçoit une entrée et fabrique une sortie indépendamment de ce qui est déjà passé ou de son état initial. Nous dirons qu'il n'a aucune « mémoire ».

Ce réseau permet d'implémenter n'importe quelle fonction logique. Prenons par exemple l'opérateur binaire correspondant à la disjonction exclusive (voir figure 3.1b). Nous laissons au lecteur le soin de vérifier qu'avec :

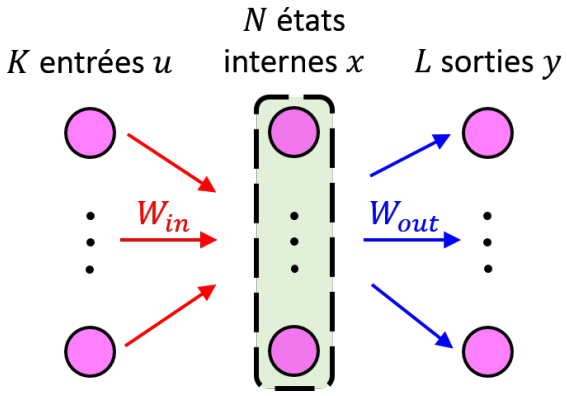
$$W_{in} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} ; W_{out} = \begin{pmatrix} -1 \, , 2 \, , -1 \end{pmatrix} \text{ et } \begin{cases} \sigma(z) = z \\ \varphi(z) = z \mod 2 \end{cases}$$

on a bien le tableau suivant :

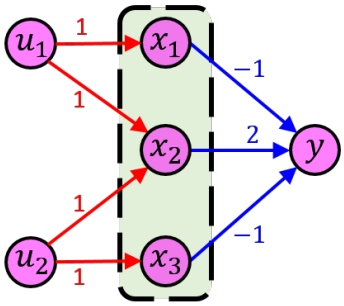
u^T	x^T	y
$(0\,,0)$	$(0\,,0\,,0)$	0
$(1\,,0)$	$(1\,,1\,,0)$	1
$(0\,,1)$	$(0\,,1\,,1)$	1
$(1\,,1)$	$(1\,,2\,,1)$	0

TABLEAU 3.1 – Réseau de neurones avançant pour la disjonction exclusive.

où la notation $(\cdot)^T$ est utilisée pour la transposée. Il suffit d'appliquer successivement les équations (3.1a) et (3.1b) pour chaque valeur de u donnée dans le tableau 3.1.



(a) Schéma générique d'un réseau de neurones avançant avec *une* couche cachée d'états internes.



(b) Disjonction exclusive.

FIGURE 3.1 – Réseau de neurones avançant.

3.2 Réseau de neurones récurrents / *Recurrent neural network*

La structure du *reservoir computer* est plutôt basée sur celle d'un *recurrent neural network* à temps discret, qui sera noté n, définit de la manière suivante :

Définition 3.2 (Recurrent neural network)**.**

Un *recurrent neural network* est un système dynamique où les unités qui le constituent interagissent entre elles et dont la structure contient **au moins un** cycle.

Le qualificatif *recurrent* indique la présence d'au moins une boucle, c'est ce qui le distingue du *feedforward neural network*. Un nouveau schéma est donné à la figure 3.2, ce dernier introduit d'une part deux nouvelles matrices de poids :

- ▷ Les **connexions internes** : $W \in \mathbb{R}^{N\times N}$ couplant $x(n)$ à sa valeur future $x(n+1)$.
- ▷ Les **poids de retour** : $W_{fb} \in \mathbb{R}^{N\times L}$ couplant $y(n)$ à $x(n+1)$. Cette matrice récupérera le signal de sortie pour le réinjecter dans le réservoir de la même manière que W_{in}, d'où l'indice fb pour *feedback*.

et, d'autre part, modifie légèrement les poids de sortie : $W_{out} \in \mathbb{R}^{L\times(N+K)}$ qui couplent maintenant $x \in \mathbb{R}^N$ et $u \in \mathbb{R}^K$ à $y \in \mathbb{R}^L$. En d'autres termes, le signal de sortie sera généré à partir de l'état généralisé $S(n) = [x(n)\,;u(n)] \in \mathbb{R}^{(N+K)}$ où la notation $[\cdot\,;\cdot]$ est utilisée pour la concaténation.
La *mise à jour* de ce réseau se fera via les équations suivantes :

$$\begin{cases} x(n+1) = C\sigma\left(W_{in}u(n+1) + Wx(n) + W_{fb}y(n)\right) & \text{(3.2a)} \\ y(n) = \varphi\left(W_{out}S(n)\right), & \text{(3.2b)} \end{cases}$$

où on a introduit explicitement une constante $C > 0$ fixant l'échelle temporelle (voir § 3.4.5). Remarquons que les équations (3.2) font effectivement intervenir la variable temporelle discrète alors qu'elle n'apparaissait pas dans les équations (3.1). En effet, la présence de cycles a complètement modifié la dynamique puisque maintenant le système utilise son passé : il aura une certaine « mémoire ».

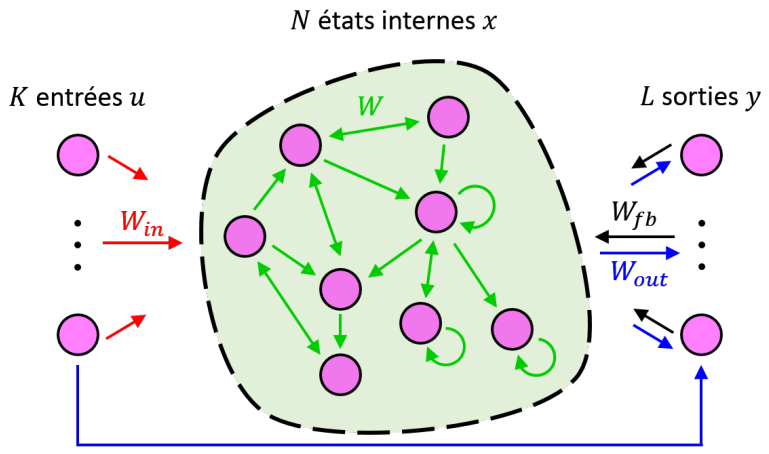


FIGURE 3.2 – Schéma d'un réseau de neurones récurrents.

Bien que la topologie du *recurrent neural network* soit celle du *reservoir computer*, les neurones de ce dernier *peuvent* être plus subtils. En effet, jusqu'ici nous les avons implicitement considéré comme des fonctions de plusieurs variables et à valeurs réelles, autrement dit comme des **neurones formels** (*artificial neuron*). Un autre type de neurone est l'**intégrateur à fuite** (voir annexe C) : on parle alors de réseau de neurones intégrateurs à fuite (*leaky integrator network*). L'évolution d'un tel réseau est régie, en temps continu, par l'équation différentielle suivante :

$$\dot{x} = C\left[-ax + \sigma\left(W_{in}u + Wx + W_{fb}y\right)\right], \tag{3.3}$$

où on a le terme de fuite $-ax$ avec $a > 0$ le taux de fuite. On discrétise (3.3) en utilisant la formule d'Euler progressive (voir section B.1) en notant $h > 0$ le pas de discrétisation, il vient :

$$\begin{cases} x(n+1) = (1 - hCa)x(n) + hC\sigma\left(W_{in}u(n+1) + Wx(n) + W_{fb}y(n)\right) & \text{(3.4a)} \\ y(n) = \varphi\left(W_{out}S(n)\right), & \text{(3.4b)} \end{cases}$$

On retrouve les équations (3.2) en annulant le premier terme, c'est-à-dire en choisissant $h = 1$ pour le pas de discrétisation et $a = 1/hC$ pour le taux de fuite[3].

Les équations (3.4) ont la bonne forme pour décrire la mise à jour du réservoir. Cependant, on peut encore affecter des gains aux trois matrices de poids. On notera α et β les gains de W_{in} et de W_{fb} respectivement. Le gain des connexions internes n'est pas vraiment un paramètre utile, c'est plutôt le rayon spectral[4] de W qui jouira d'une grande importance (voir théorème 3.1) : on fixera ce dernier pour ensuite déterminer le gain à appliquer à W et on notera $\rho\tilde{W}$ la matrice résultante où $\tilde{W} = W/\Lambda$ avec Λ le rayon spectral de W et ρ le rayon spectral final souhaité. Compte tenu de ces derniers ajouts, les équations d'évolution deviennent :

$$\begin{cases} x(n+1) = (1 - hCa)x(n) + hC\sigma\left(\alpha W_{in}u(n+1) + \rho\tilde{W}x(n) + \beta W_{fb}y(n)\right) \\ y(n) = \varphi\left(W_{out}S(n)\right) \end{cases}$$

Enfin, dans la suite, les fonctions d'entrée σ et de sortie φ que nous considéreront seront toujours la sigmoïde $\tanh$ et l'identité respectivement. La première, non-linéaire tandis que la seconde simplifiera l'entraînement (voir chapitre 4). En effet, en prenant $\varphi \equiv \mathbb{1}$ la sortie devient une simple combinaison linéaire très facile à manipuler et à implémenter. Dès lors, la dynamique de notre ordinateur réservoir sera dictée par le système d'équations suivant :

$$\begin{cases} x(n+1) = (1 - hCa)x(n) + hC\tanh\left(\alpha W_{in}u(n+1) + \rho\tilde{W}x(n) + \beta W_{fb}y(n)\right) & \text{(3.5a)} \\ y(n) = W_{out}S(n), & \text{(3.5b)} \end{cases}$$

3. On écrit explicitement h dans l'expression de a pour la cohérence des unités. En effet, même si on a pris $h = 1$, c'est ce dernier qui transporte l'unité de temps qui *doit* apparaitre inversée dans a.

4. Rappel : si $\lambda_1, \cdots, \lambda_n$ sont les valeurs propres d'une matrice $A \in \mathbb{C}^{n\times n}$, alors son rayon spectral est donné par $\rho(A) = \max\{|\lambda_1|, \cdots, |\lambda_n|\}$.

3.3 Réseau d'état d'écho / *Echo state network*

Dans cette dernière section nous souhaitons introduire le lecteur aux mathématiques qui se cachent derrière le réseau d'état d'écho (*echo state network*). On commence donc par donner quelques définitions nécessaires à la compréhension des termes « espace compact » et « espace complet ». Une fois cela fait, nous présentons l'*echo state property* pour pouvoir enfin donner la définition du réseau d'état d'écho.

3.3.1 Compacité et complétude d'un espace métrique

La première chose dont nous avons besoin est d'avoir un espace métrique. Ce dernier est définit comme suit :

Définition 3.3 (Métrique — Espace métrique)**.**

Une **métrique** sur un ensemble X est une fonction positive :

$$d : X \times X \to \mathbb{R}^+ : (x\,, y) \mapsto d(x, y)$$

vérifiant, $\forall x, y, z \in X$, les trois axiomes suivants :

(1) Non-dégénérescence : $d(x, y) = 0 \iff x = y$;

(2) Symétrie : $d(x, y) = d(y, x)$;

(3) Inégalité triangulaire : $d(x, z) \leq d(x, y) + d(y, z)$.

Le couple formé de X muni de la fonction d, noté (X, d), est appelé **espace métrique**.

Ensuite vient la définition d'espace topologique, essentielle pour pouvoir avoir la compacité.

Définition 3.4 (Topologie — Espace topologique)**.**

Une **topologie** sur un ensemble X est une classe $\mathcal{T}$ de parties de X telle que :

(1) $\varnothing \in \mathcal{T}$ et $X \in \mathcal{T}$;

(2) $\mathcal{T}$ est stable par intersections finies : $\forall\, U, V \in \mathcal{T} : U \cap V \in \mathcal{T}$;

(3) $\mathcal{T}$ est stable par unions quelconques : $\forall (U_i)_{i\in I} : \bigcup_{i\in I} U_i \in \mathcal{T}$.

Les éléments de $\mathcal{T}$ sont les **ouverts** et le couple formé de X muni de la topologie $\mathcal{T}$, et noté $(X\,, \mathcal{T})$, est appelé **espace topologique**.

On peut montrer qu'il est toujours possible de construire une topologie sur un espace métrique à l'aide de la métrique d : on parle alors de la **topologie induite** et on la note $\mathcal{T}_d$.
Forts de ces définitions, nous sommes en mesure d'introduire la compacité pour les espaces métriques.

Définition 3.5 (Espace métrique compact).

Soit (X, d) un espace métrique muni de la topologie $\mathcal{T}$. Il est **compact** si et seulement si chaque fois qu'il est recouvert par un ensemble d'ouverts, on peut en extraire un nombre *fini* d'entre eux qui recouvrent toujours X.

Finalement, nous *complétons* cette série de définitions avec les espaces complets.

Définition 3.6 (Suite de Cauchy — Espace complet).

Soit (X, d) un espace métrique. Une suite (x_n) est dite **de Cauchy** si et seulement si :

$$\forall \varepsilon > 0,\ \exists\, N(\varepsilon) \in \mathbb{N} \text{ tel que } \forall m, n \geq N(\varepsilon),\ d(x_n, x_m) < \varepsilon$$

Nous dirons que (X, d) est **complet** si et seulement si toutes ses suites de Cauchy sont convergentes vers un point de X.

3.3.2 État d'écho

Attaquons la description du réservoir. On considère pour l'instant un *recurrent neural network* de neurones formels sans retour ($W_{fb} \equiv 0$), celui-ci peut être vu comme un système piloté par $u \in U$ que l'on note $g : U \times X \to X$ où U est l'espace complet des fonctions d'entrées et X, muni de la métrique d, est l'espace métrique compact des variables internes.

Enfin, on introduit la propriété caractéristique d'un *echo state network* :

Définition 3.7 (Echo state property).

Soit $g : U \times X \to X$ où (X, d_X) est compact et (U, d_U) est complet.
Une séquence $x_{]-\infty,0]} \overset{def}{=} (\ldots, x_{-1}, x_0) \subset X$ est dite **compatible** avec $u_{]-\infty,0]} = (\ldots, u_{-1}, u_0) \subset U$ quand $x_{k+1} = g(u_k, x_k) \forall k < 0$.
Le système g possède l'***echo state property*** par rapport à U si et seulement si pour n'importe quelle $u_{]-\infty,0[} \subset U$ telle que les séquences $x_{]-\infty,0]} \subset X$ et $y_{]-\infty,0]} \subset X$ sont toutes deux compatibles avec $u_{]-\infty,0[}$, on a l'égalité $x_0 = y_0$.

En d'autres termes, l'état actuel du réseau sera déterminé de manière univoque par la « fin » de la séquence d'entrée et donc indépendamment de son état initial. Cette définition constitue les fondations mathématiques de l'*echo state network*. En effet, celui-ci est simplement définit comme :

Définition 3.8 (Echo state network).

Un *echo state network* est un *recurrent neural network* qui possède l'*echo state property*.

Reprenons notre réservoir dont la mise à jour est faite par (3.5a) où l'on se donne un signal $\hat{y}$ pour y plutôt que d'utiliser (3.5b). Ce dernier sera appelé signal d'apprentissage (voir section 4.1) mais, pour l'instant, interprétons-le comme une entrée secondaire. La définition 3.7 a donc du sens pour l'ordinateur réservoir et nous montrons au théorème 3.1 une condition suffisante pour que le *reservoir computer* possède l'*echo state property*.

Théorème 3.1 (Echo state network)**.**

Considérons notre réservoir en apprentissage forcé, c'est-à-dire le *recurrent neural network* mis à jour par (3.5a) avec $y \equiv \hat{y}$ fixé.
Si $|1 - hC(a - \rho)| < 1$, alors c'est un *echo state network*.

Preuve

Reprenons l'équation de mise à jour (3.5a) :

$$x(n+1) = (1 - hCa)x(n) + hC \tanh\left(\alpha W_{in} u(n+1) + \rho \tilde{W} x(n) + \beta W_{fb} y(n)\right)$$

En remplaçant la sortie y par le signal d'apprentissage $\hat{y}$, il vient $g : U \times X \to X$:

$$g(u, x) = (1 - hCa)x + hC \tanh\left(\alpha W_{in} u + \rho \tilde{W} x + \beta W_{fb} \hat{y}\right)$$

Si x_1 et x_2 sont deux états du réseau, alors la métrique d sur X nous permet de montrer que g est lipschitzienne sur sa deuxième variable indépendamment de u et de $\hat{y}$. En effet, en linéarisant la $\tanh$ on peut se débarrasser de u et de $\hat{y}$ et aboutir à la majoration suivante :

$$\begin{aligned} d\left(g(u, x_1), g(u, x_2)\right) &\leq d\left(\left[(1 - hCa)\mathbb{1} + hC\rho\tilde{W}\right] x_1, \left[(1 - hCa)\mathbb{1} + hC\rho\tilde{W}\right] x_2\right) \\ &= \left\| \left[(1 - hCa)\mathbb{1} + hC\rho\tilde{W}\right] (x_1 - x_2) \right\| \\ &\leq |1 - hC(a - \rho)|\, d(x_1, x_2) \end{aligned}$$

Puisque par hypothèse $|1 - hC(a - \rho)| < 1$, la distance entre les états x_1 et x_2 tend à s'annuler, ce qu'il fallait démontrer. □

3.4 Choix des paramètres

Nous commençons par donner la liste de tous les paramètres introduits jusqu'ici et qui jouent un rôle dans la mise à jour du réseau établie à l'équation (3.5a) :

- ▷ Taille du réservoir : N
- ▷ Matrices de poids : W_{in}, $\tilde{W}$ et W_{fb}
- ▷ Gains : α, β et ρ
- ▷ Taux de fuite : a
- ▷ Échelle de variation temporelle : h et C

La discussion qui suit essaye de résumer ce qui est fait dans [19] afin de donner au lecteur une idée de comment paramétrer le réservoir.

3.4.1 Taille du réservoir

Le premier paramètre que l'on considère est N, le nombre d'unités internes et donc la taille du réservoir. En règle générale, on peut se dire que plus le réservoir est grand, mieux c'est. En effet, l'équation (3.5b) nous indique que la sortie générée par le réservoir est une combinaison linéaire des états internes : plus il y en a, plus le résultat pourra être satisfaisant.
Malheureusement ce n'est pas si simple car plus le réservoir est gros et plus il risque de faire du sur-apprentissage (*overfitting*), voir section 4.2.2. En outre, nous avons toujours le problème de la puissance de calcul des ordinateurs : si l'on veut des résultats plus rapidement il ne faut pas prendre un réservoir trop volumineux et se limiter à un ou deux milliers de neurones.

3.4.2 Poids

Parmi les quatre matrices de poids présentent dans les équations (3.5), seule W_{out} doit être calculée, c'est ce que l'on appellera l'entraînement. Les trois autres matrices sont fixées dès le départ :

- ▷ Pour donner leur sens aux gains α et β (voir § 3.4.3), nous utiliserons une distribution uniforme entre -1 et 1 pour déterminer les éléments de W_{in} et W_{fb} ;
- ▷ En ce qui concerne les connexions internes, nous utilisons la même distribution mais nous normalisons à 1 le rayon spectral de la matrice $\tilde{W}$ et nous imposons en plus que la **connectivé** γ (la proportion d'éléments non-nuls) de la matrice soit de l'ordre de 1%. De faibles valeurs de γ permettent d'accélérer les calculs en utilisant les fonctions appropriées fournies par MATLAB®.

3.4.3 Gains

Les gains permettent de contrôler la non-linéarité du réservoir. En effet, celle-ci se manifeste dans (3.5b) via le terme :

$$\tanh\left(\alpha W_{in}u(n+1) + \rho\tilde{W}x(n) + \beta W_{fb}y(n)\right)$$

La figure 3.3, rappelle le comportement de la fonction $\tanh$.

▷ Si les gains sont suffisamment petits, alors les termes $\alpha W_{in} u(n+1)$ et $\beta W_{fb} y(n)$ ne contribuent presque pas et l'expression ci-dessus peut être développée en série de Taylor autour de l'origine. Ainsi supprime-t-on la non-linéarité au premier ordre.

▷ Si au contraire les gains sont plus élevés, on quitte la zone où le comportement de la tangente hyperbolique est linéaire et le terme que l'on considère devient bivalué : -1 ou 1.

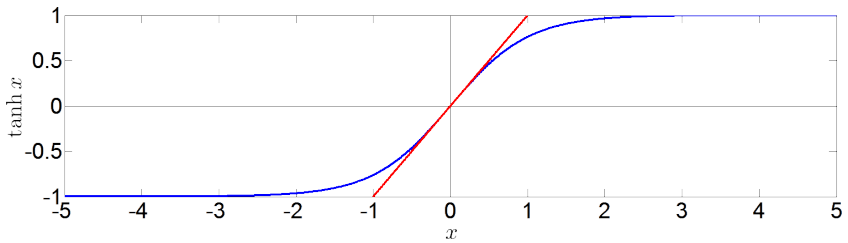


FIGURE 3.3 – Comportement de la fonction $\tanh x$, linéaire autour de l'origine.

La contribution du réseau dans le terme non-linéaire est contrôlée par ρ. Comme le montre le théorème 3.1, pour $(1-hCa)$ fixé, ce rayon spectral régule la vitesse à laquelle le *reservoir computer* se stabilise et donc la vitesse à laquelle l'influence d'une entrée injectée dans le réservoir s'éteint : plus ρ est grand, plus l'entrée restera en mémoire longtemps.

3.4.4 Taux de fuite

Nous traitons plus en détail l'intégrateur à fuite en annexe (voir annexe C). En résumé, nous y montrons que le système va « oublier » exponentiellement sa valeur actuelle au taux a. Le taux de fuite peut donc s'interpréter comme la vitesse à laquelle le réservoir se met à jour.
On prendra une valeur de a telle que la dynamique du réservoir corresponde à celle de l'entrée et/ou de la sortie mais, malheureusement, il n'y a pas de méthode générale pour cela et le choix se fait surtout par essai-erreur, à l'intuition.

3.4.5 Échelle de variation temporelle

Les paramètres h et C apparaissent toujours ensemble dans les équations (3.5), on a donc choisi de prendre $h = 1$ et de discuter de C. Notons que ce paramètre n'est pas repris dans [19] mais plutôt dans [14]. Ce dernier régule l'échelle de variation temporelle et, s'il est suffisamment petit, permettra d'émuler des systèmes lents.

Il faudra donc aiguiser son intuition et s'armer de patience pour réaliser de nombreux tests afin d'aboutir à un jeu de paramètres acceptable.

Chapitre 4

Entraînement d'un ordinateur réservoir

L'implémentation du *reservoir computer* est terminée, il est grand temps de le faire travailler. Le premier objectif de ce chapitre est de présenter une méthode d'entraînement pour le réservoir : l'apprentissage forcé (*teacher-forced*), en tenant compte de la régularisation de Tikhonov.
Nous proposons ensuite deux exemples (donnés uniquement à titre pédagogique) : la classification de signaux et la reproduction d'un signal périodique.
Enfin, le second objectif est d'entraîner le réservoir à réaliser une tâche plus utile pour nous : l'émulation de systèmes dynamiques et particulièrement du système de Mackey - Glass.

4.1 Apprentissage forcé

4.1.1 Calculs des poids de sortie

L'idée de l'apprentissage forcé est de considérer, dans un premier temps, un **signal d'apprentissage** $\hat{y}$ de longueur T_{train} et de forcer le réservoir à le reconstruire. Pour réaliser cette prouesse, on détermine les poids de sortie W_{out} optimaux qui minimisent l'erreur quadratique moyenne entre la sortie du réservoir $y(n) \overset{(3.5b)}{=} W_{out}S(n)$ et le signal d'apprentissage $\hat{y}(n)$.

Définition 4.1 (Erreur quadratique moyenne)**.**

Soit y le signal en sortie du réservoir et $\hat{y}$ le signal d'apprentissage.
L'erreur quadratique moyenne est définie de la manière suivante :

$$\mathcal{M} = \frac{1}{N}\sum_{n=1}^{N}\left(y(n) - \hat{y}(n)\right)^2 \tag{4.1}$$

Voici, étape par étape, comment réaliser l'apprentissage forcé :

(1) On construit le réservoir en fixant les trois matrices de poids W_{in}, $\tilde{W}$ et W_{fb}, les gains d'entrée et de retour α et β respectivement, le pas h, la constante globale C et le taux de fuite a.

(2) On injecte dans le réservoir $u(n)$ et $\hat{y}(n)$ dans l'entrée et le retour respectivement et on collecte ainsi les T_{train} **états généralisés** $S(n) = [x(n)\,;u(n)]$ en utilisant l'équation (3.5a). On a donc une matrice $S \in \mathbb{R}^{(N+K)\times T_{train}}$ dont les colonnes sont $S(n)$.

(3) Enfin, en injectant (3.5b) dans (4.1) et en minimisant $\mathcal{M}$ on détermine les poids W_{out} :

$$W_{out} = \underset{W\in\mathbb{R}^{L\times(N+K)}}{\arg\min}\left(\frac{1}{T_{train}}\sum_{n=1}^{T_{train}}\left(WS(n) - \hat{y}(n)\right)^2\right) \tag{4.2}$$

Le théorème 4.1 donne la solution de ce problème d'optimisation.

Théorème 4.1 (Entraînement d'un ordinateur réservoir par apprentissage forcé)**.**

Considérons notre réservoir en apprentissage forcé, c'est-à-dire mis à jour par (3.5a) avec $y \equiv \hat{y}$ fixé, le signal d'apprentissage de longueur T_{train}.
Les poids W_{out} solutions de (4.2) sont donnés par :

$$W_{out} = (\hat{y}S^T)(SS^T)^\dagger, \tag{4.3}$$

où on a les états généralisés $S \in \mathbb{R}^{(N+K)\times T_{train}}$ et $(\cdot)^\dagger$ est le pseudo-inverse.

Preuve

Minimisons l'erreur quadratique moyenne pour trouver la matrice $W_{out} \in \mathbb{R}^{L\times(N+K)}$ optimale. Pour cela, reprenons $\mathcal{M}(W)$ obtenu en (4.2) :

$$\mathcal{M}(W) = \frac{1}{T_{train}} \sum_{n=1}^{T_{train}} (WS(n) - \hat{y}(n))^2$$

On peut raisonner séparément sur les composantes de $\mathcal{M}(W)$ et entraîner individuellement chacune des L sorties du réservoir. Soit donc la composante $l \in \{1, \cdots, L\}$. En explicitant tous les indices et en distribuant le carré, on peut l'écrire comme :

$$\mathcal{M}_l(W) = \frac{1}{T_{train}} \sum_{n=1}^{T_{train}} \left[\left(\sum_{i=1}^{N+K} W_{li}S_{in} \right) \left(\sum_{j=1}^{N+K} W_{lj}S_{jn} \right) - 2\hat{y}_{ln} \left(\sum_{i=1}^{N+K} W_{li}S_{in} \right) + (\hat{y}_{ln})^2 \right]$$

Enfin, on annule la dérivée pour minimiser cette expression :

$$\frac{\partial \mathcal{M}_l(W)}{\partial W_{lk}} = \frac{1}{T_{train}} \sum_{n=1}^{T_{train}} \left[2S_{kn} \left(\sum_{j=1}^{N+K} W_{lj}S_{jn} \right) - 2S_{kn}\hat{y}_{ln} \right] \stackrel{!}{=} 0$$

On se débarrasse du facteur $2/T_{train}$ pour arriver à :

$$\sum_{n=1}^{T_{train}} \left[S_{kn} \left(\sum_{j=1}^{N+K} W_{lj}S_{jn} \right) \right] = \sum_{n=1}^{T_{train}} 2S_{kn}\hat{y}_{ln} \iff WSS^T = \hat{y}S^T$$

Finalement, on obtient les poids optimaux en introduisant le pseudo-inverse :

$$W = (\hat{y}S^T)(SS^T)^\dagger$$

□

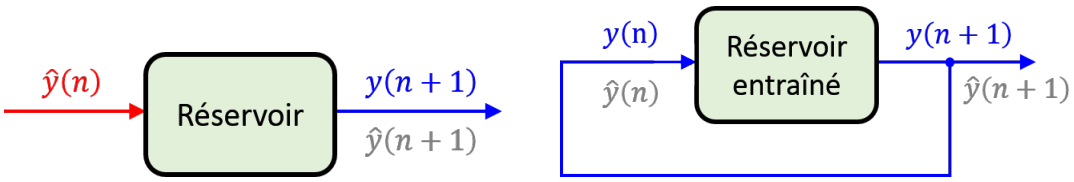
Les poids de sortie sont donnés par (4.3) : l'apprentissage forcé est terminé et le réservoir est prêt à tourner tout seul ! Pour ce faire, on remplace simplement le signal d'apprentissage par la sortie générée via la relation (3.5b) pour $n > T_{train}$.

4.1.2 Prédiction de séries temporelles

Après l'apprentissage forcé, l'équation (3.5b) pour la sortie du réservoir prend la forme :

$$y(n) = W_{out}S(n) \approx \hat{y}(n)$$

Ainsi, lorsque l'on substitue le signal d'apprentissage par le signal de sortie du réservoir, le système devient fermé et il va « prolonger » $\hat{y}(n)$: il reçoit $y(n) \approx \hat{y}(n)$ et on s'attend à ce qu'il produise $y(n+1) \approx \hat{y}(n+1)$. Les figures 4.1 illustrent la situation où les valeurs grisées sont les valeurs idéales pour la sortie du réservoir.



(a) Apprentissage forcé (b) Evolution libre

FIGURE 4.1 – Entraînement pour la prédiction de séries temporelles.

Notons que le signal d'entrée u ne sera pas nécessairement utile ici : tout se passe dans le *feedback*.

4.2 Améliorations

Dans ce paragraphe nous donnons au lecteur deux améliorations qui peuvent aider à obtenir de meilleurs résultats pendant et/ou après l'entraînement :

(1) Ajouter un transient avant l'entraînement pour se défaire de l'état initial ;

(2) Ajouter de la régularisation pour éviter le surapprentissage (*overfitting*).

4.2.1 Transient

La première chose que l'on peut faire est de laisser le temps au *reservoir computer* de se stabiliser avant de faire l'entraînement. En effet, comme l'ordinateur réservoir est initialement dans l'état $x \equiv 0$, il lui faudra un certain temps, qu'on va appeler **transient**, avant de se stabiliser par rapport à la dynamique du signal d'apprentissage.
Ce transient sera d'autant plus long que ρ sera grand. En effet, on a vu au paragraphe 3.4.3 que : « ρ régule la vitesse à laquelle l'influence d'une entrée injectée dans le réservoir s'éteint » et donc ρ régule également la vitesse à laquelle ce dernier va oublier son état initial.
En général, on calculera les poids de sortie sans tenir compte du premier millier d'états généralisés. Ainsi, les variables internes auront complètement oublié leur état initial et l'entraînement sur le signal d'apprentissage n'en sera que meilleur.

4.2.2 Régularisation de Tikhonov

Bien que nous ne l'avons pas utilisée dans la plupart des tâches que nous avons considérées, la régularisation de Tikhonov fut une astuce utile pour obtenir les résultats finaux quant à l'utilisation du réservoir en cryptographie. Nous allons voir, brièvement, en quoi cela consiste. Reprenons la solution du problème d'optimisation, autrement dit les poids donnés par (4.3). La régularisation de Tikhonov, que l'on appelle également régression d'arête (*ridge regression*), propose plutôt :

$$W_{out} = (\hat{y}S^T)(SS^T + \delta\mathbb{1})^\dagger, \tag{4.4}$$

où δ est le **coefficient de régularisation**. Cette solution pour les poids de sortie correspond en fait à un nouveau problème d'optimisation :

$$W_{out} = \underset{W\in\mathbb{R}^{L\times(N+K)}}{\arg\min} \left(\frac{1}{T_{train}} \sum_{n=1}^{T_{train}} \left(WS(n) - \hat{y}(n)\right)^2 + \delta\bar{W} \right), \tag{4.5}$$

où $\bar{W}_l = \sum_{i=1}^{N+K} W_{li}^2$. On peut s'en convaincre avec une preuve tout à fait analogue à celle qui a été donnée pour le théorème 4.1. Cette modification va jouer deux rôles :

- ▷ **Limiter les poids de sortie** : L'idée est de choisir $\delta > 0$ pour ne pas autoriser les poids de sortie trop grands qui peuvent créer des instabilités dans la dynamique du réservoir. En effet, si certaines valeurs de W_{out} sont trop importantes, elles vont amplifier le signal des nœuds correspondants. Il en résulte que, si ces signaux subissent de faibles déviations par rapport à la séquence d'entraînement, ces petites différences vont se voir multipliées à chaque mise à jour du réservoir et ce dernier deviendra instable.

- ▷ **Empêcher le surapprentissage** : Ce terme de régularisation permet d'empêcher un surapprentissage (*overfitting*) des données et, au final, d'améliorer la capacité du réservoir à généraliser les données. Illustrons le surapprentissage avec un exemple simple. Considérons un ensemble de données dont la représentation se rapproche d'une droite.

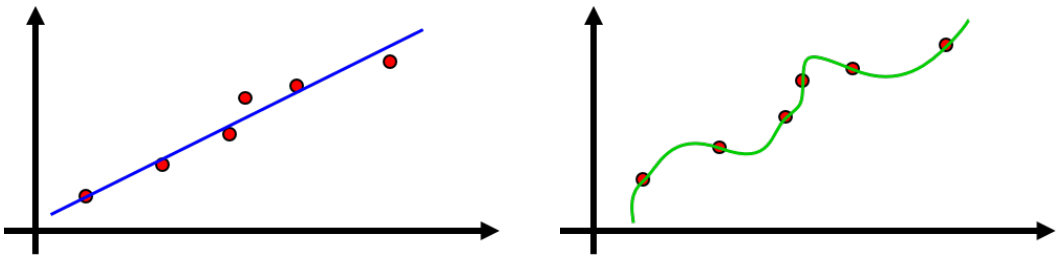
  Guidés par nos réflexes pavloviens, nous faisons passer une droite à travers ce nuage de points en effectuant une régression linéaire (fig. 4.2a). Bien que l'erreur d'apprentissage ne soit pas nulle dans ce cas, ce modèle linéaire dégage une propriété importante de nos donnés et il est capable de prolonger celles-ci sans que l'erreur n'augmente brutalement par rapport à l'entraînement.

  Imaginons maintenant qu'un polynôme de degré élevé permettrait d'obtenir une erreur d'apprentissage nulle en interpolant les données (fig. 4.2b). Ce nouveau modèle passe complètement à côté du caractère linéaire des données et il est incapable de fournir la moindre valeur en dehors des données sans une augmentation importante de l'erreur.

L'avantage de ce coefficient δ est qu'il ne dépend pas de l'entraînement. En effet, on peut l'optimiser en utilisant l'équation (4.4) tout en conservant en mémoire les états généralisés S : il n'est pas nécessaire de les recollecter lorsque l'on change le coefficient de régularisation.

Une méthode alternative consiste à ne pas modifier la diagonale de SS^T comme dans (4.4) et d'ajouter du bruit gaussien dans le réservoir. Nous avons choisi d'injecter le bruit *dans* l'argument de la $\tanh$ comme le fait [14], à l'inverse de [19], où le bruit est additionné *en dehors*. Cette régularisation alternative fournit une meilleure immunisation au bruit, puisque ce dernier se propage dans tous les états internes, mais au prix de l'indépendance à l'entraînement sus-mentionnée : il faudra collecter les états généralisés à chaque lancement du programme.

Finalement, un coefficient de régularisation bien dosé peut améliorer l'entraînement et stabiliser la dynamique une fois le signal d'apprentissage remplacé par la sortie du réservoir.



(a) Régression linéaire (b) Interpolation

FIGURE 4.2 – Exemple illustrant le surapprentissage.

4.3 Exemples

Dans cette section nous présentons deux exemples afin d'illustrer tous ce qui a été vu sur le *reservoir computer* :

(1) Classification de signaux ;

(2) Reproduction d'un signal périodique quelconque.

Le lecteur n'ayant pas ce besoin peut directement passer à la section 4.4 où l'on introduit l'émulation du système de Mackey - Glass. En effet, les exemples donnés ici le sont à titre purement pédagogique et ne seront pas utilisés plus loin.

4.3.1 Classification de signaux

Objectif et fonction d'apprentissage

Pour cette tâche, le réservoir recevra à l'entrée une fonction $u(n)$ qui oscillera entre -1 et 1 tantôt comme un sinus, tantôt comme un signal carré. L'objectif sera de subdiviser et classer le signal $u(n)$ en construisant $y(n)$ qui prendra les valeurs 0 ou 1 selon que l'entrée était un sinus ou un signal carré respectivement. Notons que toutes les unités sont arbitraires.

La figure 4.3 représente le début des signaux d'entrée et d'apprentissage u et $\hat{y}$ respectivement. Dans l'exemple que nous traitons, u est constituée de 150 périodes de longueur 20 pour une longueur totale $T = 3000$.

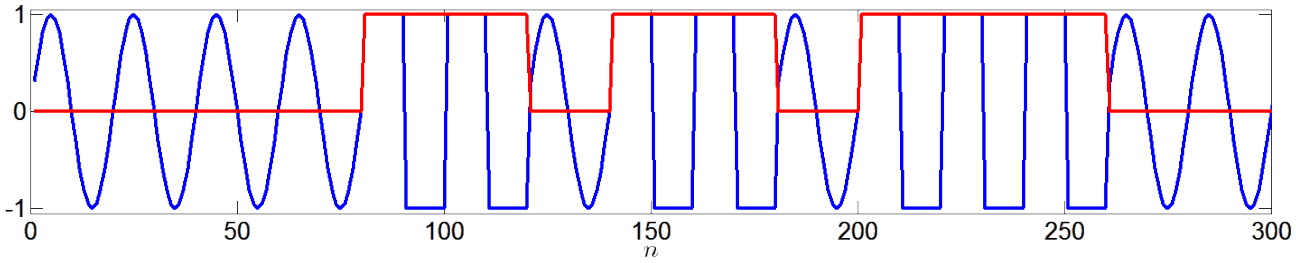


FIGURE 4.3 – Représentation des 15 premières périodes des signaux d'entrée et d'apprentissage pour la classification en bleu et rouge respectivement.

Initialisation du réservoir

Pour réaliser cette tâche, nous considérons un petit *reservoir computer* de 50 unités internes. Les trois matrices de poids prennent leurs valeurs uniformément entre -1 et 1 et la matrice des connexions internes a une connectivité $\gamma = 0.01$ et un rayon spectral $\rho = 0.9$. On applique un gain de sortie $\beta = 0.005$ et tous les autres paramètres, α, h, C et a sont fixés à 1 de sorte que l'équation de mise à jour (3.5a) prenne la forme de (3.2a) :

$$x(n+1) = \tanh\left(W_{in}u(n+1) + 0.9\tilde{W}x(n) + 0.005W_{fb}\hat{y}(n)\right)$$

Entraînement à la classification

Le réservoir passe par un transient de 1000 itérations pour se défaire de son état initial et, ensuite, il est entraîné sur 1000 autres itérations par apprentissage forcé. Le résultat de cet entraînement est donné à la figure 4.4.

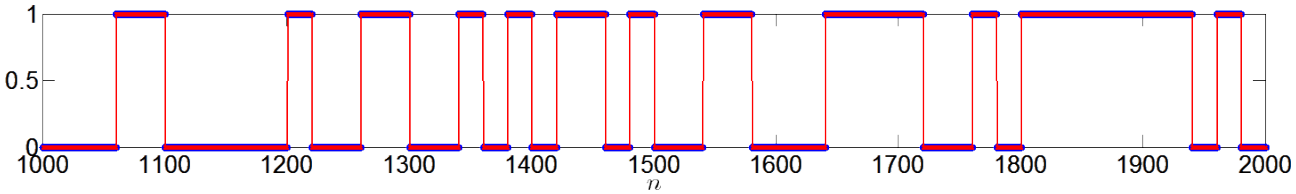


FIGURE 4.4 – Entraînement du réservoir à la classification : la fonction d'apprentissage est en bleu tandis que la fonction générée par le réservoir est en rouge.

Notons que, spécifiquement pour cette tâche, l'équation (3.5b) pour la sortie du réservoir est légèrement modifiée. En effet, *après* avoir entraîné le réservoir et déterminé les poids de sorties W_{out}, le résultat du produit $W_{out}S(n)$ est systématiquement arrondi. En d'autres termes, plutôt que de prendre $\varphi(z) = z$, on a pris $\varphi(z) = \bar{z}$, où $\bar{z}$ est l'arrondi de z à l'entier le plus proche. Ainsi, le réservoir sera très stable car le signal généré par ce dernier prendra ses valeurs dans $\{0, 1\}$ comme pour la fonction d'apprentissage.

Classification par le réservoir en évolution autonome

Le réservoir est livré à lui-même pour les 1000 dernières itérations et la figure 4.5 présente ses résultats.

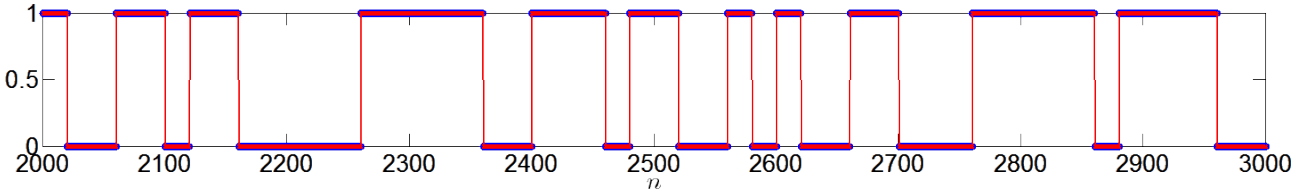


FIGURE 4.5 – Classification du réservoir : la fonction attendue est en bleu tandis que la fonction générée par le réservoir est en rouge.

La classification par le réservoir est parfaite.

4.3.2 Reproduction d'un signal périodique quelconque

Objectif et fonction d'apprentissage

Le problème posé ici est d'entraîner le réservoir à prédire une fonction périodique quelconque. L'entrée ne sera pas utile ici, le réservoir recevra uniquement la fonction d'apprentissage dont une période est construite en choisissant des valeurs aléatoirement entre 0 et 1 pour chaque valeur de n. La figure 4.6 nous montre une de ces périodes dont la longueur est $P = 100$. Le signal final contiendra 30 périodes pour une longueur totale $T = 3000$.

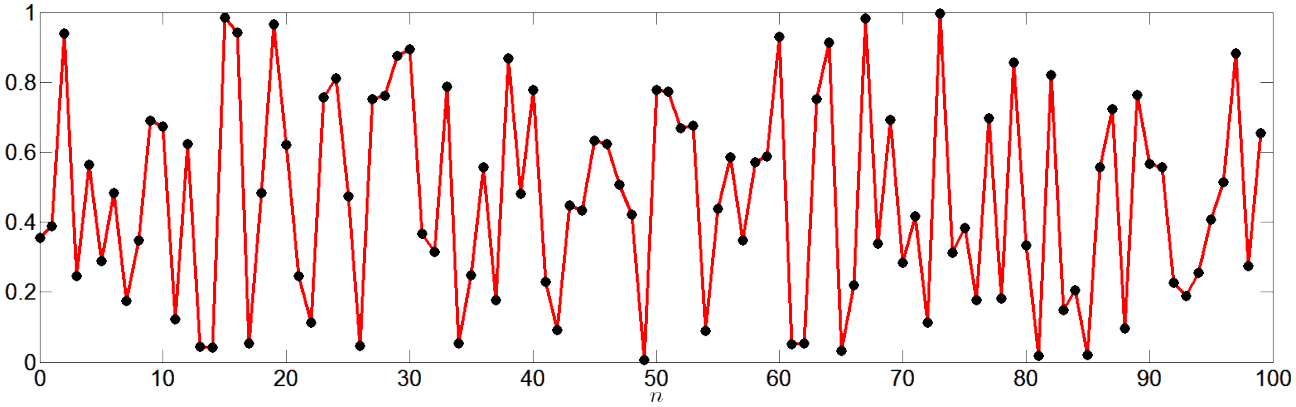


FIGURE 4.6 – Première période de la fonction d'apprentissage.

Initialisation du réservoir

Pour réaliser cette tâche, nous considérons un *reservoir computer* assez conséquent : il est constitué de 1000 unités internes. Les éléments des matrices W et W_{fb} sont choisis uniformément entre -1 et 1 et la matrice des connexions internes a une connectivité $\gamma = 0.05$ et un rayon spectral $\rho = 0.9$. Tous les autres paramètres, α, β, h, C et a sont fixés à 1 et la mise à jour des états internes se fera comme :

$$x(n+1) = \tanh\left(0.9\tilde{W}x(n) + W_{fb}\hat{y}(n)\right)$$

Entraînement et évolution libre

Le réservoir passe par un transient de 1000 itérations pour se défaire de son état initial. Ensuite, il est entraîné sur 1000 autres itérations et il devient autonome pour les 1000 dernières itérations. La figure 4.7 présente les résultats de l'entraînement et de la prédiction où :

$$\mathcal{R}(n) = \log\left|\frac{y(n) - \hat{y}(n)}{\hat{y}(n)}\right|$$

est l'erreur relative entre le signal cible et celui reproduit par le réservoir.

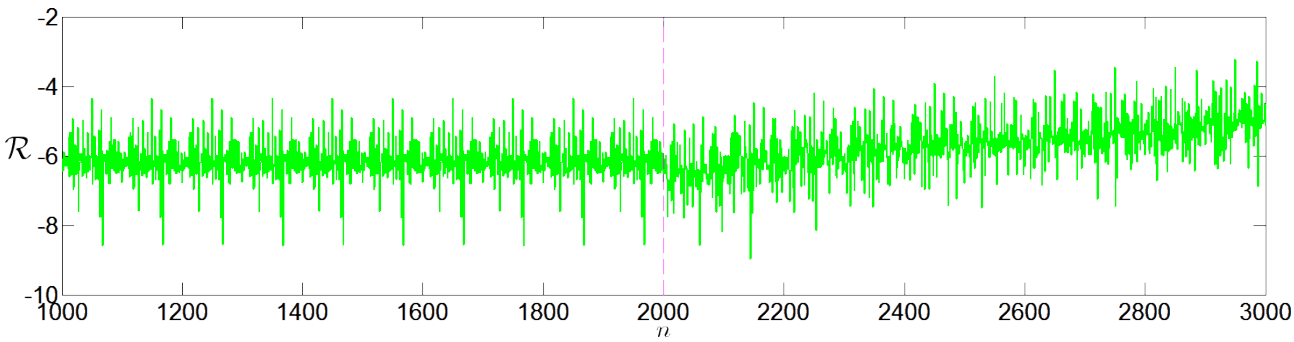


FIGURE 4.7 – Évolution de $\mathcal{R}$. Le trait vertical magenta sépare l'entraînement de l'évolution libre.

La reproduction par le réservoir est excellente : l'erreur est de l'ordre de 10^{-6}. Cependant, dès que l'entraînement est arrêté, l'erreur va commencer à diverger.

4.4 Émulation du système de Mackey - Glass

Attaquons enfin l'émulation du système de Mackey - Glass présenté à la section 1.7.

Objectif et fonction d'apprentissage

Si on note $\tilde{y}$ le signal obtenu en intégrant (1.9) avec le plus petit pas proposé par MATLAB® et sous-échantillonné avec un pas de 0.5 s (c'est-à-dire $0.5n = t$), alors la fonction d'apprentissage sera $\hat{y}(t) = \tanh(\tilde{y}(t) - 1)$. Cette transformation permet de ramener $\hat{y}$ dans l'intervalle $[-1\,,1]$ (voir [12]). Alors que la figure 1.4 nous montrait $\tilde{y}$, la figure 4.8 nous donne le début de $\hat{y}(t)$.

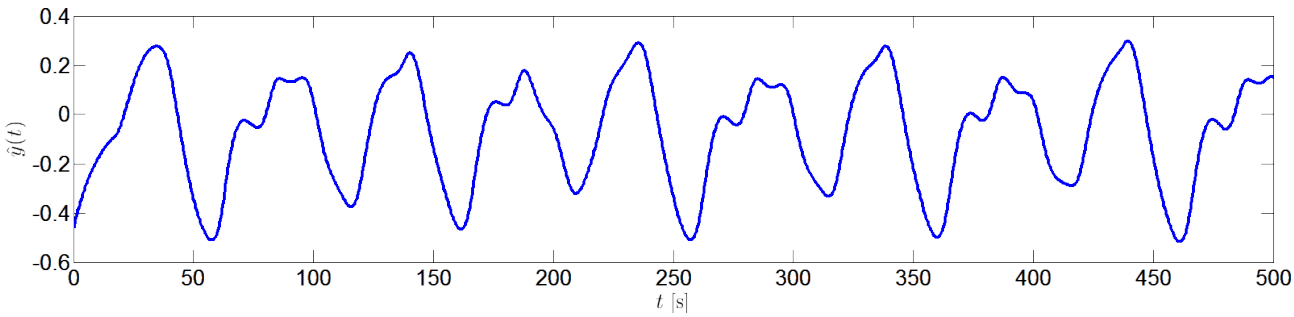


FIGURE 4.8 – Fonction d'apprentissage pour émuler le Mackey - Glass : $\hat{y}(t) = \tanh(\tilde{y}(t) - 1)$.

Bien que nous présentions cette transformation sur la fonction d'apprentissage, notons déjà qu'au

chapitre 5 nous travaillerons directement avec $\tilde{y}$.

Pour l'entrée, on prend une constante $u \equiv 0.2$ qui sert de biais.

Initialisation du réservoir

Pour réaliser cette tâche, nous considérons un *reservoir computer* de 1500 unités internes. Les matrices de poids sont choisies uniformément entre -1 et 1 où la matrice des connexions internes a une connectivité $\gamma = 0.01$ et un rayon spectral $\rho = 0.79$. Les paramètres C et a sont fixés à 0.44 et 0.9 respectivement et on prend un pas $h = 0.5$ alors que les autres paramètres : α et β sont maintenus à 1. Notons que l'on a bien $1 - 0.5 \cdot 0.44(0.9 - 0.79) \approx 0.976 < 1$ et la mise à jour des états internes se fait via :

$$x(n+1) = (1 - 0.5 \cdot 0.44 \cdot 0.9)x(n) + 0.5 \cdot 0.44 \tanh\left(W_{in}u(n+1) + 0.79\tilde{W}x(n) + W_{fb}\hat{y}(n)\right)$$

Entraînement et évolution libre

Le réservoir passe par un transient de 500 secondes pour se défaire de son état initial. Ensuite, il est entraîné sur 1500 secondes et il devient autonome pour les 1500 secondes suivantes. Rappelons-nous que $\hat{y}(t)$ a été échantillonné avec un pas de 0.5 s : le nombre de points d'intégration est donc le double du nombre de secondes.

La figure 4.9 présente les résultats de l'entraînement et de la prédiction. Précisons que pour cette figure et toutes celles qui vont suivre, nous sommes retournés dans les unités de départ, c'est-à-dire que l'on a appliqué $\operatorname{atanh}(\cdot) + 1$ à y et à $\hat{y}$ avant de les réaliser. Par exemple, la figure 4.9 nous donne l'évolution de l'erreur relative suivante :

$$\mathcal{R}(t) = \log\left|\frac{\operatorname{atanh}(y(t))+1 - \operatorname{atanh}(\hat{y}(t))-1}{\operatorname{atanh}(\hat{y})(t) + 1}\right|$$

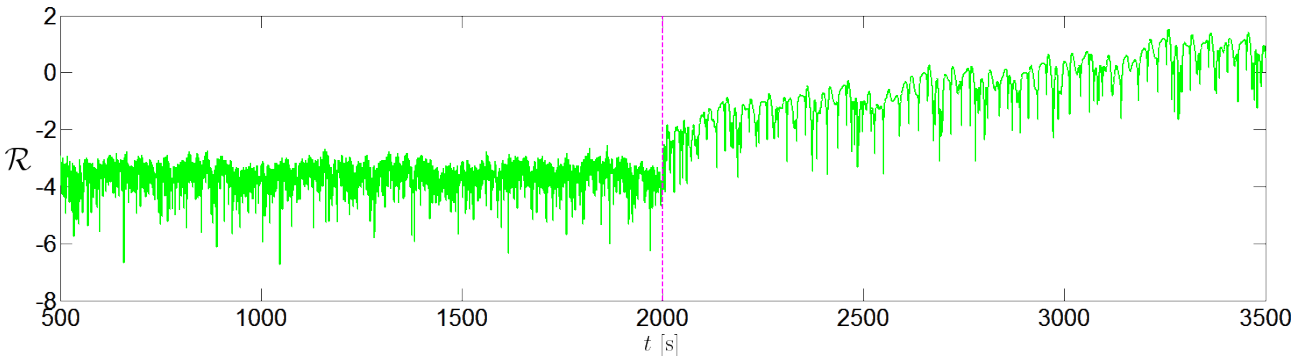
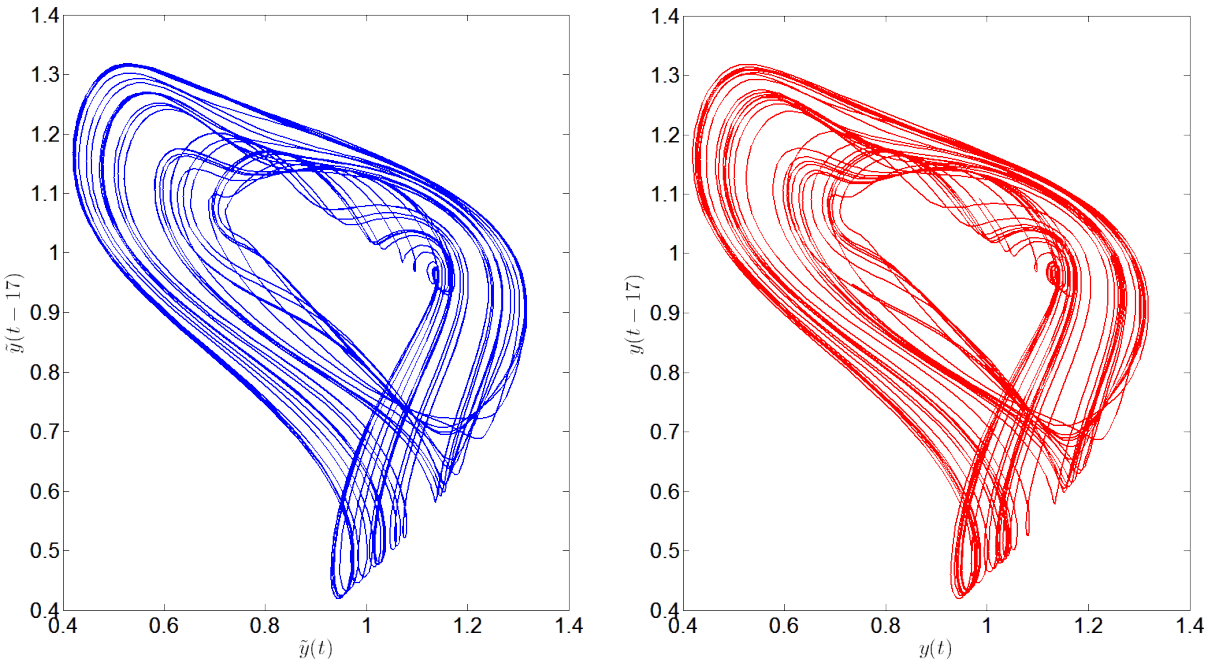


FIGURE 4.9 – Évolution de $\mathcal{R}$. Le trait vertical magenta sépare l'entraînement de l'évolution libre.

L'émulation par le réservoir est plutôt bonne : l'erreur relative est de l'ordre de 10^{-4} mais, de nouveau, dès que l'entraînement est arrêté la différence entre les signaux commence à croître jusqu'à saturer autour de 10^{1}. A ce stade, on pourrait penser que ca ne va pas du tout. En effet, un facteur 10 entre le signal généré par le système original et celui émulé ce n'est pas très satisfaisant mais regardons

plutôt les attracteurs donnés à la figure 4.10.



(a) Intégration numérique (b) Réservoir

FIGURE 4.10 – Comparaison des attracteurs obtenus par intégration numérique et par le réservoir.

Le réservoir arrive à reproduire correctement l'attracteur : il a donc bien cerné la dynamique du système de Mackey - Glass. L'erreur importante observée ci-dessus traduit en fait la forte dépendance aux conditions initiales de l'équation (1.9), ceci est rendu visible à la figure 4.11.

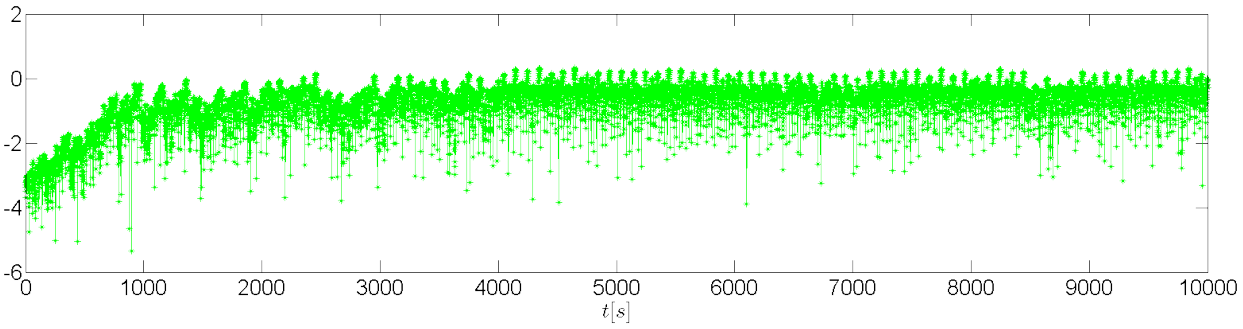


FIGURE 4.11 – Évolution de $\log\left|\frac{y_2(t)-y_1(t)}{y_1(t)}\right|$ montrant la forte dépendance aux conditions initiales du système de Mackey - Glass où y_1 intégré avec $\mathbb{S} \equiv 0.5$ et y_2 avec $\mathbb{S} \equiv 0.5001$.

L'erreur relative entre y_1 et y_2 sature et ne dépasse pas la valeur de $10^{0.4}$. En effet, les signaux sont bornés supérieurement et inférieurement. Si on approxime ces bornes par 1.4 et 0.4 respectivement pour la borne supérieure et inférieure, alors l'erreur maximale est atteinte quand l'un des systèmes se trouve au maximum et l'autre au minimum. Par un simple calcul, on retrouve cette saturation de l'erreur :

$$\frac{1.4-1}{0.4} = 2.5 \Longrightarrow \log(2.5) \approx 0.4$$

Enfin, la figure 4.12 compare la dynamique de $\tilde{y}(t)$ a celle de $y(t)$ entre $t = 3000$ s et $t = 3500$ s.

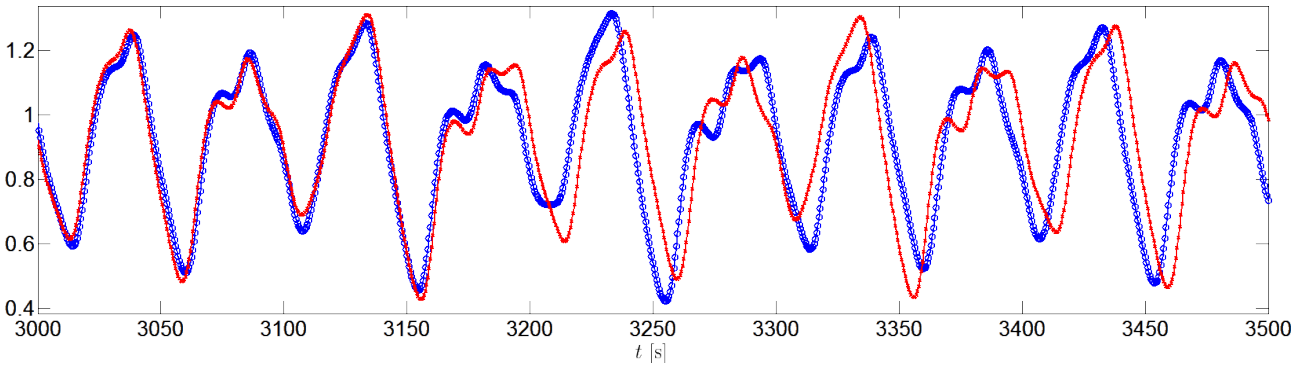


FIGURE 4.12 – Comparaison des dynamiques du signal d'apprentissage $\tilde{y}(t)$ en bleu et de la sortie du réservoir $y(t)$ en rouge dans les dernières secondes du l'évolution libre.

On observe alors que la dynamique du réservoir entraîné est bien celle d'un système de Mackey - Glass, mais *peut-être* que ce n'est pas tout à fait celle du système sur lequel le *reservoir computer* s'est entraîné. *Peut-être* qu'il existe un jeu de paramètres $(\bar{\beta}, \bar{\gamma}, \bar{n}, \bar{\tau})$ et/ou une autre histoire $\mathbb{S}$ tels que le signal généré par le réservoir soit identique à celui de ce nouveau système de Mackey - Glass. C'est une question que l'on s'est posée mais qui demeure ouverte.

Chapitre 5

Utilisation d'un ordinateur réservoir pour décrypter un message transmis via la cryptographie par chaos

Dans ce chapitre nous allons reprendre les protocoles de communication introduits au chapitre 2 en changeant le système de Bob. En effet, ce dernier possédait une copie parfaite du système d'Alice et, de ce fait, il lui était assez aisé de récupérer le message $m(t)$ envoyé par Alice en suivant les protocoles 2.1 et 2.2.

Bob est maintenant muni d'un ordinateur réservoir qu'il va d'abord entraîner chez Alice. Ensuite, nous allons voir si, une fois chez lui, il parvient à décrypter $m(t)$ dans chacune des situations abordées au chapitre 2.

Enfin, nous proposerons un nouveau protocole pour la transmission par superposition via l'utilisation de l'ordinateur réservoir.

5.1 Superposition du signal chaotique et du message

Nous avions appliqué le protocole 2.1. Celui-ci est rappelé ci-dessous :

(1) **Alice envoie son signal chaotique pur** : Alice émet $s(t) = a(t)$ jusque l'instant T_P fixé au préalable par Alice et Bob ;

(2) **Verrouillage de Bob sur Alice** : Bob choisit $q \in]0\,,1[$ tel que $T_V \leq T_P$. Avec cette efficacité, il verrouille son système sur celui d'Alice de sorte que $\forall t \geq T_V : b(t) = a(t)$;

(3) **Arrêt du verrouillage et superposition du message** : Bob pose $q = 0$ et il laisse son système évoluer librement sur la même orbite que celle d'Alice pendant qu'elle envoie le signal $s(t) = a(t) + m(t)$;

(4) **Récupération du message par Bob** : Bob génère le message en sortie en effectuant la différence $z(t) = s(t) - b(t) = m(t)$.

Si nous voulons remplacer le système de Bob par un ordinateur réservoir, alors nous devons commencer par l'entraîner comme à la section 4.4 afin qu'il puisse reproduire la dynamique d'un système de Mackey - Glass. Ainsi, Bob disposera d'une copie *imparfaite* du système transmetteur. Avec celle-ci, il va appliquer le protocole de communication 2.1 et nous verrons si l'entraînement a été suffisamment bien réalisé pour décrypter entièrement ou en partie le message.

Après avoir testé l'utilisation du réservoir dans le protocole 2.1, nous en proposerons un nouveau pour la transmission par superposition qui entraînera différemment l'ordinateur réservoir.

5.1.1 Test du protocole avec un ordinateur réservoir entraîné

L'utilisation du protocole 2.1 requiert le verrouillage du *reservoir computer entraîné* sur le système de Mackey - Glass d'Alice comme à la section 4.4. Un schéma illustrant la méthode pour Bob avec ou sans réservoir est donné à la figure 5.1. Dans les figures à venir, le magenta concernera Bob alors qu'il use du système de Mackey - Glass tandis que l'orange se rapportera à l'utilisation d'un ordinateur réservoir comme système receveur.

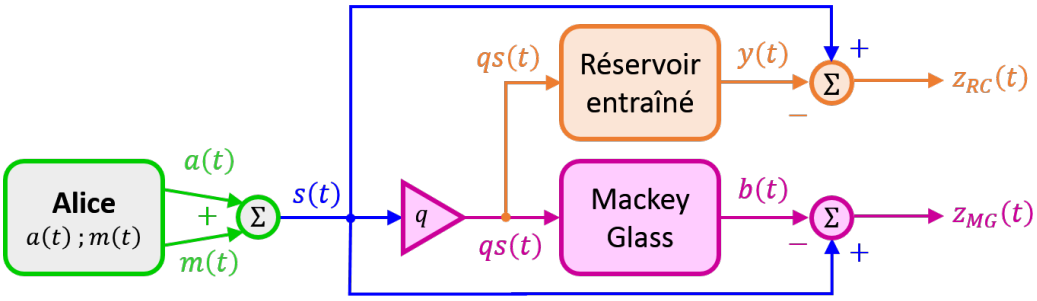


FIGURE 5.1 – Schéma pour la communication chaotique par superposition avec ou sans réservoir. Le message m et l'efficacité q ne sont pas toujours présents (voir le protocole 2.1).

La figure 5.2 présente l'évolution de l'erreur relative $\mathcal{R}$ entre le système de Mackey - Glass et l'ordinateur réservoir entraîné alors que ce dernier se verrouille sur le premier avec $q = 0.95$ jusque $t = 1000$ s. Au-delà, le systèmes sont découplés : $q = 0$.

Au paragraphe 4.4, nous avions une erreur relative pendant l'entraînement qui était de l'ordre de 10^{-4} (fig. 4.9) : on retrouve évidemment cette valeur dans le verrouillage. En effet, l'erreur présentée ici ne descend pas de manière significative sous cette limite. Ensuite, lorsque l'on arrête le verrouillage, le réservoir se retrouve en évolution libre et l'erreur commence à augmenter comme lorsque l'entraînement avait cessé à la figure 4.9.

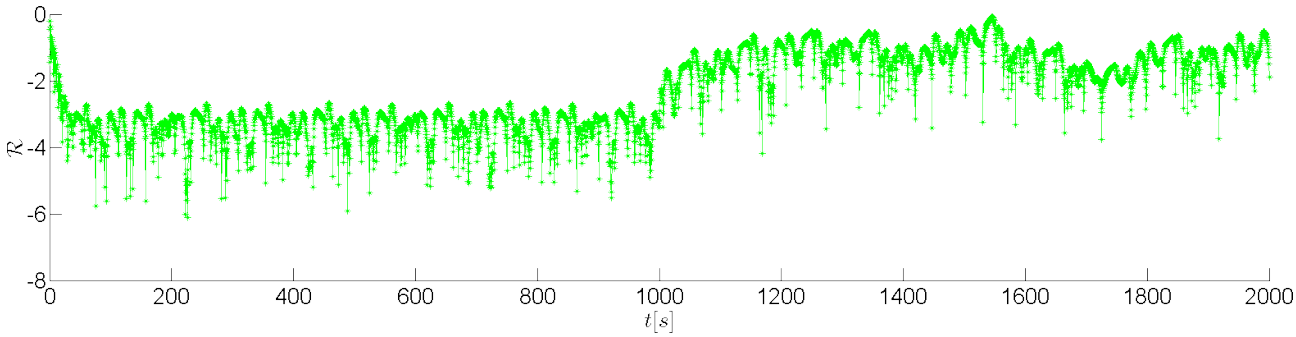


FIGURE 5.2 – Évolution de $\mathcal{R}$ montrant le verrouillage d'un *reservoir computer* entraîné sur un Mackey - Glass avec $q = 0.95$ jusque $t = 1000$ s et $q = 0$ ensuite.

Au vu du comportement de l'erreur discuté ci-dessus, on s'attend à ce que les premiers bits *après le verrouillage* soient correctement décryptés. C'est en effet ce qui se passe, comme le montre la figure 5.3. Celle-ci est obtenue en appliquant le protocole 2.1 avec un message constitué d'une suite de bits d'amplitude 10^{-2} où les 160 premiers sont nuls pour laisser le temps à l'ordinateur réservoir de Bob de se verrouiller : $T_P = 160$ s.

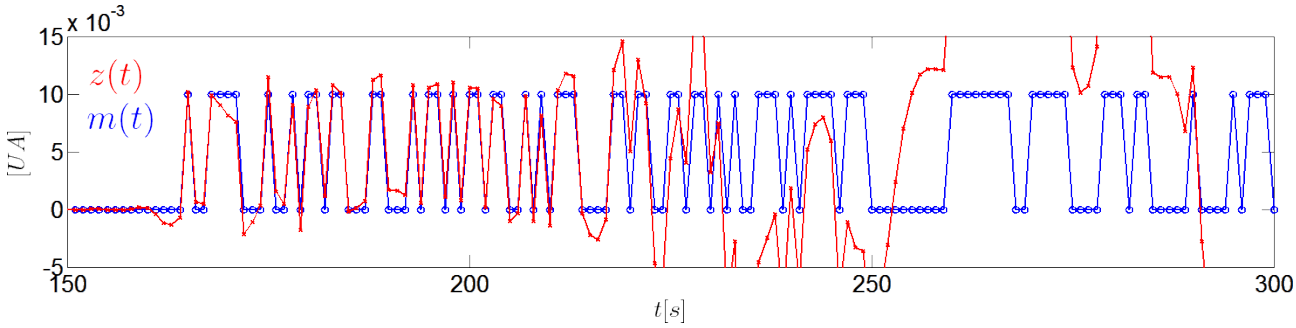


FIGURE 5.3 – Transmission par superposition via l'utilisation d'un ordinateur réservoir.

Nous sommes confrontés à un problème similaire à l'ajout de bruit dans la communication qui avait été discuté au paragraphe 2.2.5. En effet, la seconde étape du protocole 2.1 n'est pas correctement réalisée puisque l'on arrive pas exactement à $\forall t \geq T_V : y(t) = a(t)$. La solution déjà proposée en 2.2.5 permettrait d'obtenir un protocole viable utilisant l'ordinateur réservoir.

Outre ce problème de désynchronisation, l'amplitude du message que nous avons considérée ici était fixée à 10^{-2} au lieu de 10^{-5} comme à la section 2.2. En effet, comme le réservoir a une précision de 10^{-4}, une message de l'ordre 10^{-5} ne peut pas être décrypté.

5.1.2 Nouveau protocole pour la transmission par superposition avec réservoir

Notre idée est la suivante : Bob va entraîner son réservoir à reproduire le signal d'Alice $a(t)$ en recevant $s(t) = a(t) + m_e(t)$, où $m_e(t)$ est un exemple de message. Ainsi, lorsqu'il recevra le signal contenant le message caché, $s(t) = a(t) + m(t)$, il sera en mesure d'extraire le signal porteur $a(t)$ pour pouvoir décrypter le message puisque ce dernier aura la même forme que l'exemple m_e.

L'entraînement est très similaire à ce qui a été fait en 4.4. Cependant, la tâche n'est plus la même et l'entraînement est adapté en conséquence. Nous explicitons ces quelques changements ci-dessous :

- ▷ **Paramétrisation du réservoir** : nous reprenons les mêmes paramètres que pour l'émulation du système de Mackey - Glass présentée au paragraphe 4.4 à ceci près que $h = 1$ et que nous enlevons le *feedback* en prenant $W_{fb} \equiv 0$. La mise à jour des états internes se fait via :

$$x(t+1) = (1 - 0.44 \cdot 0.9)x(n) + 0.44 \tanh\left(W_{in}u(t+1) + 0.79\tilde{W}x(t)\right)$$

- ▷ **Apprentissage forcé** : nous injectons $u(t) = a(t) + m_e(t)$ à l'entrée du réservoir afin de collecter les états généralisés nécessaires au calcul des poids de sortie. Notez que les signaux sont bien exprimés en temps discret t où le pas de discrétisation h est unitaire. Bien que $W_{fb} \equiv 0$, on a tout de même le signal d'apprentissage $\hat{y}(t) = a(t)$ qui intervient dans l'équation (4.3) pour le calcul les poids de sortie.

Une fois l'entraînement terminé avec l'exemple $m_e(t)$, Alice superpose son message caché $m(t)$ à son signal chaotique $a(t)$ et elle envoie $s(t) = a(t) + m(t)$. Bob récupère $s(t)$ et l'injecte à l'entrée de son réservoir qui génère $y(t) \approx a(t)$. Finalement, il effectue la même différence qu'auparavant

pour approximer le message caché en sortie :

$$z(t) = s(t) - y(t) = a(t) + m(t) - y(t) \approx m(t) \tag{5.1}$$

Nous donnons à la figure 5.4 un nouveau schéma pour bien mettre en évidence l'injection constante du signal $s(t)$ dans le réservoir : c'est sur ce point que les protocoles 2.1 et 5.1 diffèrent.

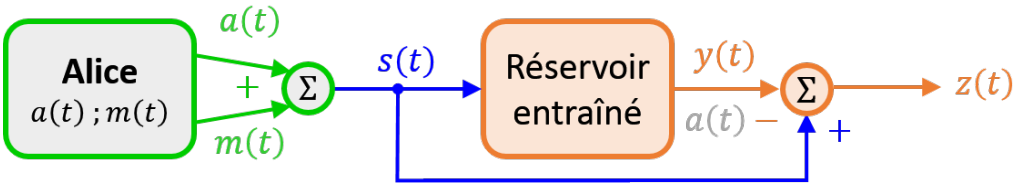


FIGURE 5.4 – Schéma alternatif pour la communication chaotique par superposition avec réservoir.

Avant de donner le protocole à appliquer, nous devons encore discuter de quelques points.

▷ **Forme des messages** : Alice et Bob fixent la forme des messages utilisés afin d'en simplifier la récupération. Nous avons considéré des chaînes de N_b bits valant 0 ou 1 d'amplitude $A_m = 10^{-2}$ où chaque bit b est répété $r = 50$ fois. En d'autres termes, comme on a $h = 1$ pour le pas de discrétisation, le message aura la forme d'une fonction étagée où chacun des N_b intervalles, noté I_b, sous-tendant le bit b a une longueur de 50 s.

▷ **Filtre en sortie** : le signal obtenu en sortie par (5.1) ne prendra pas ses valeurs dans $\{0, A_m\}$. On reconstruit alors la forme du message en appliquant le filtre suivant :

- *Normalisation* : les valeurs de $z(t)$ strictement inférieures ou supérieures à $A_m/2$ sont ramenées à 0 ou 1 respectivement.
- *Moyennisation* : on effectue une moyenne arithmétique μ_b sur chaque intervalle I_b du signal $z(t)$ normalisé.
- *Reconstruction* : on fixe finalement la valeur de $z(t)$ à 0 ou A_m sur l'intervalle I_b selon que μ_b soit strictement inférieur ou supérieure à $A_m/2$ respectivement.

Bien entendu, ce filtre ne pourra être appliqué que si Bob connait *déjà* le nombre de répétitions r ainsi que l'amplitude du message envoyé A_m.

▷ **Taux d'erreur** : enfin, on définit le taux d'erreur (*Symbol Error Rate*).

Définition 5.1 (Taux d'erreur)**.**

Soit $x(t)$ et $y(t)$ deux signaux de longueur L où $t \in \{0, \cdots, L-1\}$.
Le taux d'erreur (*Symbol Error Rate*) entre x et y s'écrit :

$$SER = \frac{\#\text{ erreurs}}{L}$$

où on compte une erreur pour chaque t tel que $x(t) \neq y(t)$.

Ce dernier est idéal pour comparer des signaux discrets.

Le nouveau protocole est alors le suivant :

Protocole 5.1 (Transmission par superposition avec ordinateur réservoir).

(1) **Configuration** : Alice et Bob fixent la forme des messages : A_m et r ;

(2) **Entraînement sur un exemple** : chez Alice, Bob entraîne son ordinateur réservoir à produire $a(t)$ en recevant $s(t) = a(t) + m_e(t)$;

(3) **Superposition du message** : Alice envoie le signal $s(t) = a(t) + m(t)$ chez Bob ;

(4) **Récupération du message par Bob** : Bob génère $z(t) = s(t) - y(t) \approx m(t)$;

(5) **Décryptage du message par Bob** : Bob applique le filtre en sortie sur $z(t)$ pour reconstruire $m(t)$.

Résultats

Sans plus attendre, voici ce que l'on a obtenu avec un entraînement sur un message contenant 400 bits. Le message caché, quant à lui, contenait 800 bits : la figure 5.5 nous montre le décryptage des cinquante premiers. Sur l'ensemble du message on a un taux d'erreur de 2.62%.

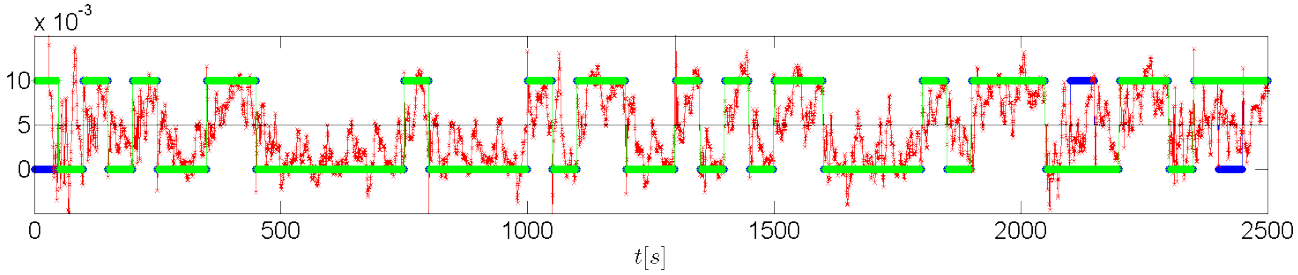


FIGURE 5.5 – Résultats de la communication par superposition non-bruitée où Bob utilise un ordinateur réservoir entraîné plutôt qu'une copie du système d'Alice. Le message original est en bleu tandis que celui reconstruit est en vert. Ce dernier est obtenu en filtrant la sortie $z(t)$, en rouge.

Bien que ca ne soit pas parfait, l'essentiel du message est bien décrypté : l'entraînement du réservoir a porté ses fruits.

Sensibilité au bruit

Souvenez-vous, le verrouillage était extrêmement sensible à l'ajout d'un bruit externe dans le signal $s(t)$ transmis par Alice (§ 2.2.5). Nous testons la robustesse du protocole 5.1 de la même manière : on reprend la situation du paragraphe précédent en ajoutant du bruit uniforme d'amplitude $A_\nu = A_m = 10^{-2}$. Cet ajout est fait dans l'expression de $s(t)$ pendant l'entraînement avec l'exemple ainsi que pendant la communication. La figure 5.6 nous présente les cinquante premiers bits où le taux d'erreur sur le message entier est passé de 2.62% à seulement 4.13%.

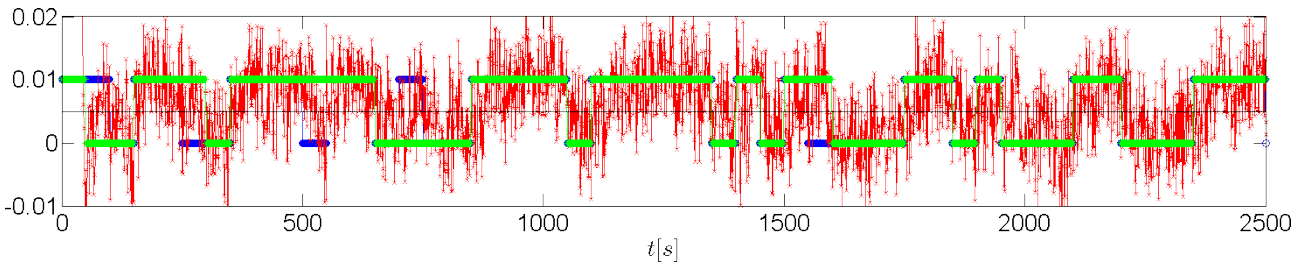


FIGURE 5.6 – Résultats de la communication par superposition bruitée où Bob utilise un ordinateur réservoir entraîné plutôt qu'une copie du système d'Alice. Le message original est en bleu tandis que celui reconstruit est en vert. Ce dernier est obtenu en filtrant la sortie $z(t)$, en rouge.

Bien que nous ayons répété les bits afin de permettre le bon fonctionnement du protocole 5.1, cette modification n'améliorait pas les performances de récupération lorsque Alice et Bob utilisent le protocole 2.1. En effet, pour ce dernier, l'ajout du bruit fait que les systèmes d'Alice et Bob vont se désynchroniser indépendamment du nombre de répétition et le message sera toujours irrécupérable tant que Bob ne se verrouille pas régulièrement sur le système d'Alice.

L'utilisation du protocole 5.1 plutôt que le 2.1 est donc beaucoup plus robuste au bruit et se passe de verrouillage au prix d'une répétition importante des bits.

5.2 Mélange non-linéaire du signal chaotique et du message

Comme pour la section précédente, on commence par rappeler le protocole 2.2 :

(1) **Configuration** : Alice et Bob choisissent au préalable une configuration $\mathcal{I}/i$;

(2) **Alice injecte le message** : le message est injecté chez Alice en $\mathcal{I}$;

(3) **Alice envoie le signal** : Alice envoie le signal $s(t)$ du nœud i ;

(4) **Activation du système de Bob** : Bob reçoit $s(t)$ en i et l'utilise pour produire $s'(t)$;

(5) **Construction du signal de sortie par Bob** : $z(t) = s(t) - s'(t)$ (tableau 2.1) ;

(6) **Récupération du message par Bob** : $m(t)$ est décrypté (tableau 2.2).

On se place à nouveau dans la configuration $III/1$ présentée au paragraphe 2.3.2 dont la dynamique des systèmes d'Alice et Bob était dictée par les équations suivantes :

$$\begin{cases} \varepsilon_0 \dot{a}(t) = -a(t) + f\left[a(t-\tau_0)\right] + m(t) & \text{(5.2a)} \\ \varepsilon_0 \dot{b}(t) = -b(t) + f\left[a(t-\tau_0)\right] & \text{(5.2b)} \end{cases}$$

Des équations (5.2), on avait extrait une expression pour le message :

$$m(t) = \varepsilon_0 \dot{z}(t) + z(t),$$

avec le signal de sortie donné par $z(t) \overset{def}{=} s(t) - s'(t) \overset{III/1}{=} a(t) - b(t)$.

5.2.1 Entraînement du réservoir pour le mélange non-linéaire

On va apprendre au réservoir à générer $s'(t)$ à partir de $s(t)$. Plus particulièrement, en considérant la configuration $III/1$, on va entraîner le *reservoir computer* à reproduire $b(t)$ à partir de $a(t)$ où les dynamiques de a et b sont régies par les équations (5.2). Pour ce faire, voici ce que nous avons implémenté (un schéma est donné à la figure 5.7) :

- ▷ **Paramétrisation du réservoir** : nous listons ici les paramètres qui diffèrent de ceux utilisés à la section 4.4. Tout d'abord, les signaux sont discrétisés avec un pas $h = 0.1$ que l'on reprend pour le pas du réservoir. Ensuite, on a pris $\alpha = 0.9$ pour le gain d'entrée et nous avons annulé W_{fb} puisque le système original de Bob était ouvert. Enfin, on a introduit du bruit gaussien ν d'amplitude 10^{-5} dans l'argument de la $\tanh$ pour empêcher le surapprentissage par le réservoir (voir § 4.2.2). On a donc :

$$x(t+1) = (1 - 0.044 \cdot 0.9)x(n) + 0.044 \tanh\left(0.9W_{in}u(t+1) + 0.79\tilde{W}x(t) + \nu(t)\right)$$

- ▷ **Apprentissage forcé** : en ce qui concerne l'entraînement, l'idée est de reproduire ce que faisait Bob avec son ancien système à retard. Concrètement, ce dernier va entrer $u(t) = a(t)$ dans son ordinateur réservoir et l'entraîner avec la fonction d'apprentissage $\hat{y}(t) = b(t)$. Nous rappelons que les signaux $a(t)$ et $b(t)$ sont générés par intégration numérique (§ B.3.3) des équations (5.2) ramenées au Mackey - Glass habituel (§ 2.3.1) avec, pour $m(t)$, un exemple de message de la même forme que le futur message caché (voir paragraphe suivant).

Dès que le réservoir est entraîné, Alice injecte son message caché dans (5.2a) et elle envoie son signal à l'entrée du réservoir de Bob. Celui-ci fait les mêmes manipulations que lorsqu'il disposait d'une copie du système d'Alice : il collecte $z(t) = a(t) - y(t)$ en sortie et obtient une approximation du message avec $m(t) \approx \varepsilon_0 \dot{z}(t) + z(t)$.

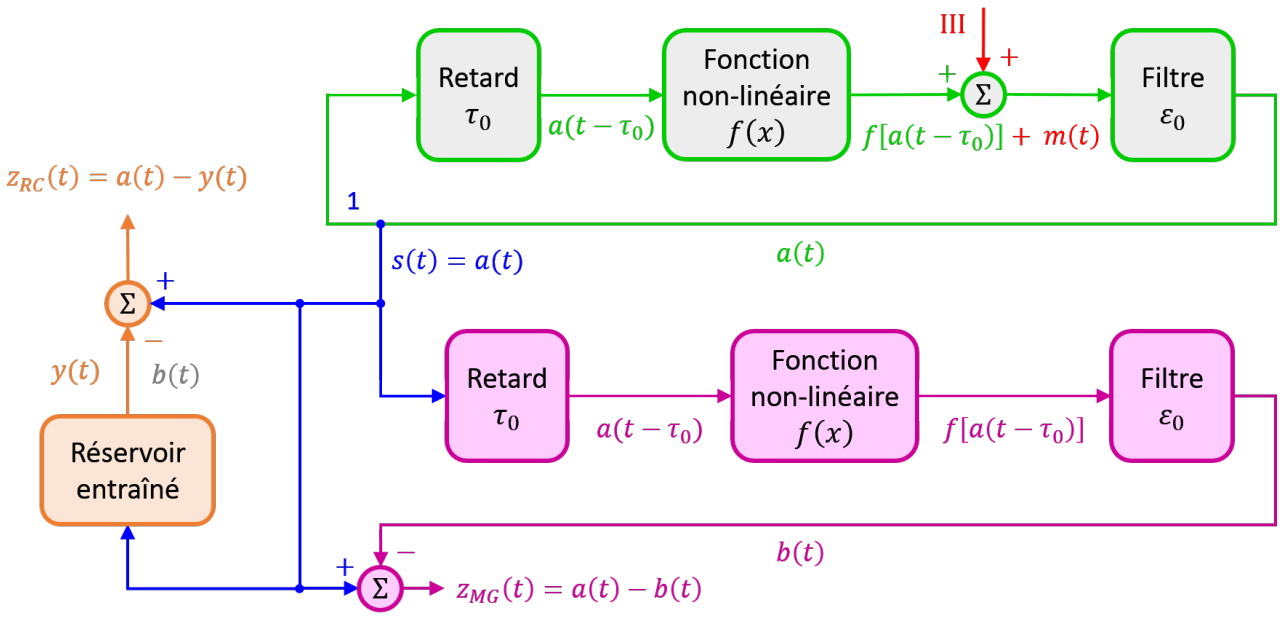


FIGURE 5.7 – Configuration $III/1$ pour la communication chaotique avec et sans réservoir en orange et magenta respectivement.

5.2.2 Résultats

On demande à Alice d'injecter un message sinusoïdal : $m(t) = A_m sin(\omega t)$. La figure 5.8 nous montre la réussite du décryptage avec $A_m = 0.05$ et $\omega = 1 (\Rightarrow f = 1/2\pi \approx 0.16\ Hz)$ par comparaison des spectres des messages original et reconstruit.

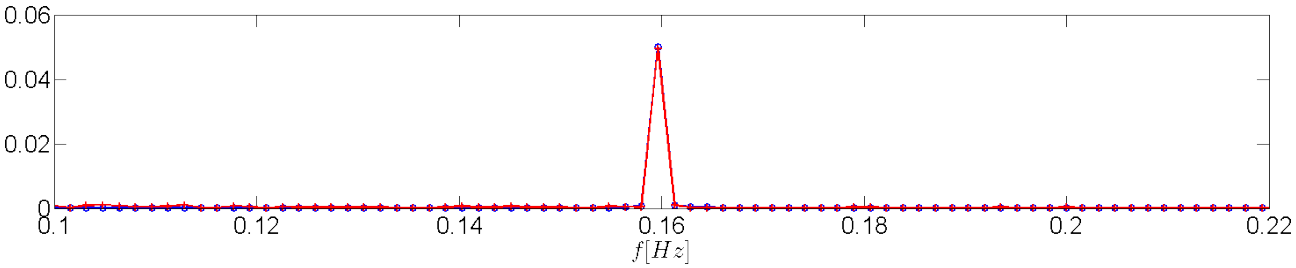


FIGURE 5.8 – Résultats de la communication non-bruitée par mélange non-linéaire où Bob utilise un ordinateur réservoir entraîné plutôt qu'une copie du système d'Alice. Le spectre du message original est en bleu tandis que celui reconstruit est en rouge.

Dans ce cas simple où le message est monochromatique, le spectre de celui-ci est correctement décrypté.

Malheureusement, nous n'avons pas réussi à transmettre un message similaire à celui utilisé à la section 2.3.4. Ce dernier était peut-être trop complexe à récupérer car modulé en basses fréquences. Il y a là toute une recherche à faire pour comprendre ce qui n'a pas fonctionné : étaient-ce les fréquences trop basses, ou bien la présence d'une modulation fréquentielle qui complique le spectre, ou encore les paramètres du réservoir mal choisis, autre chose ?

5.2.3 Sensibilité au bruit : modification de l'entraînement

Pour terminer, nous refaisons le test précédent en incorporant du bruit $\nu(t)$ uniformément généré avec une amplitude 10^{-2} afin que Bob reçoive $s(t) = a(t) + \nu(t)$. Notons que ce bruit n'est pas ajouté de manière interne au signal d'Alice mais bien de manière externe au signal transmis par elle. On applique le protocole 2.2 et cela ne fonctionne pas : le réservoir est instable.

Plutôt que de chercher à améliorer le jeu de paramètre, nous avons opté pour une approche moins fastidieuse : modifier l'entraînement en autorisant une rétro-action du réservoir en reprenant des poids de retour W_{fb} générés uniformément entre -1 et 1. On ajuste alors le gain de retour β à 0.8 et le niveau de bruit gaussien ν à 10^{-4} de sorte que la mise à jour du réservoir est donnée par :

$$x(t+1) = (1 - 0.44 \cdot 0.9)x(n) + 0.44 \tanh\left(0.9W_{in}u(t+1) + 0.79\tilde{W}x(t) + 0.8W_{fb}y(t) + \nu(t)\right),$$

où on a repris $h = 1$ pour le pas du réservoir. On applique alors le protocole 2.2 où, cette fois, l'étape (4) n'est plus tout à fait identique. En effet, en modifiant l'entraînement comme nous l'avons explicité ci-dessus, la production de $s'(t)$ se fait à partir de $s(t)$ *et* de $s'(t-1)$ plutôt que simplement à partir de $s(t)$. Cette rétro-action (représentée à la figure 5.9) va stabiliser le réservoir.

Enfin, la figure 5.10 présente les résultats obtenus en appliquant le protocole 2.2 mais en adaptant l'entraînement comme discuté ci-dessus. On y observe quelques parasites mais le pic représentant le message est correctement décrypté.

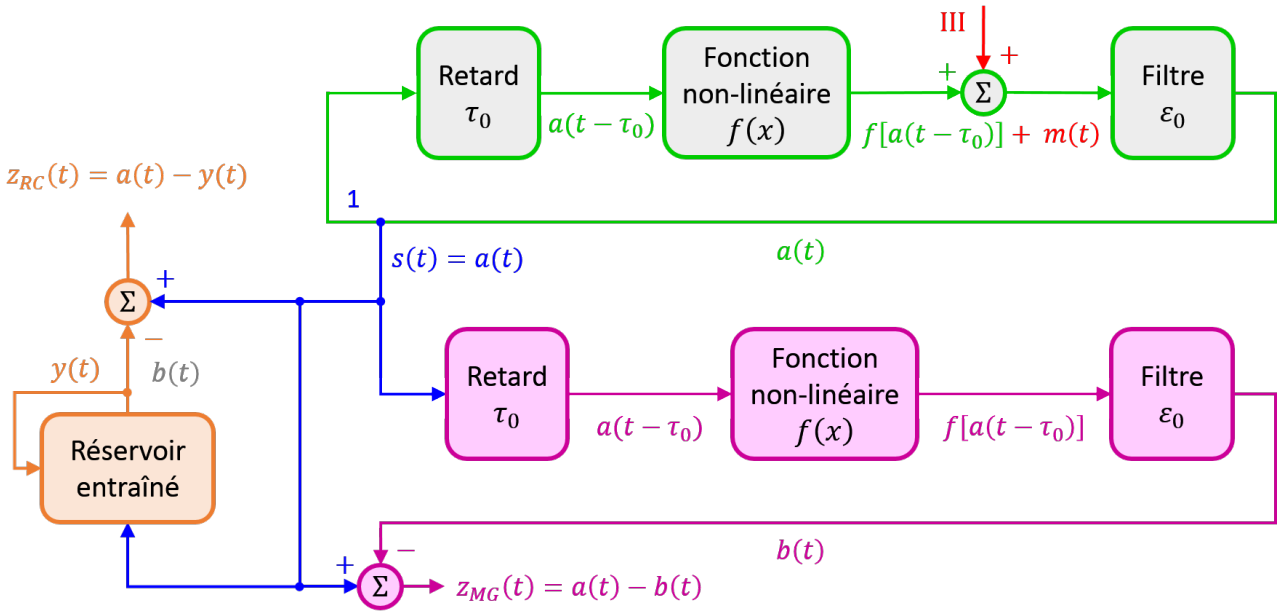


FIGURE 5.9 – Configuration $III/1$ pour la communication chaotique avec et sans réservoir.

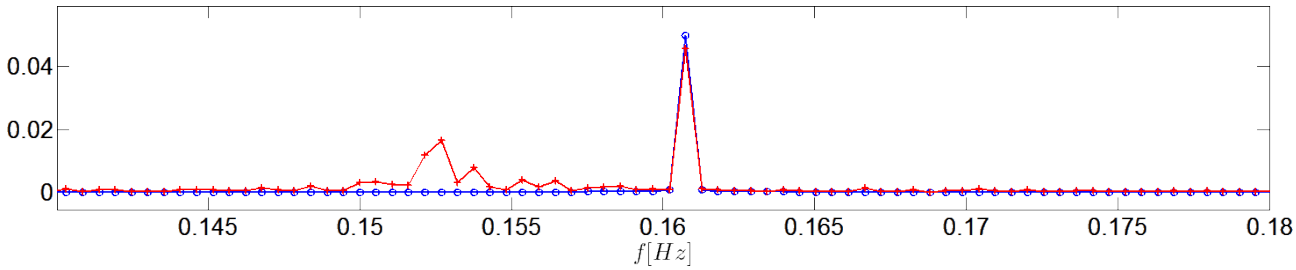


FIGURE 5.10 – Résultats de la communication bruitée par mélange non-linéaire où Bob utilise un ordinateur réservoir entraîné plutôt qu'une copie du système d'Alice. Le spectre du message original est en bleu tandis que celui reconstruit est en rouge.

5.2.4 Remarques sur la méthode d'entraînement proposée

Avant de conclure, nous attirons l'attention du lecteur sur la méthode d'entraînement du réservoir explicitée en 5.2.1. Il peut sembler que nous la proposions comme une évidence mais la réalité fut tout autre. En effet, nous avions d'abord essayé, sans succès, une autre méthode. Celle-ci voulait reproduire, à la sortie du réservoir, le signal $b(t)$ que Bob obtenait avec sa copie du système d'Alice. Cependant, nous injections un biais $u \equiv cste$ comme celui utilisé à la section 4.4 et le signal d'apprentissage $\hat{y} = b(t)$. Bien que l'entraînement se passait à merveille (fig. 5.11), le réservoir finissait par produire un signal oscillant de manière erratique dès qu'il était livré à lui-même. Compte tenu de la réussite de l'entraînement, c'était probablement le choix des paramètres du réservoir qui n'était pas optimal.

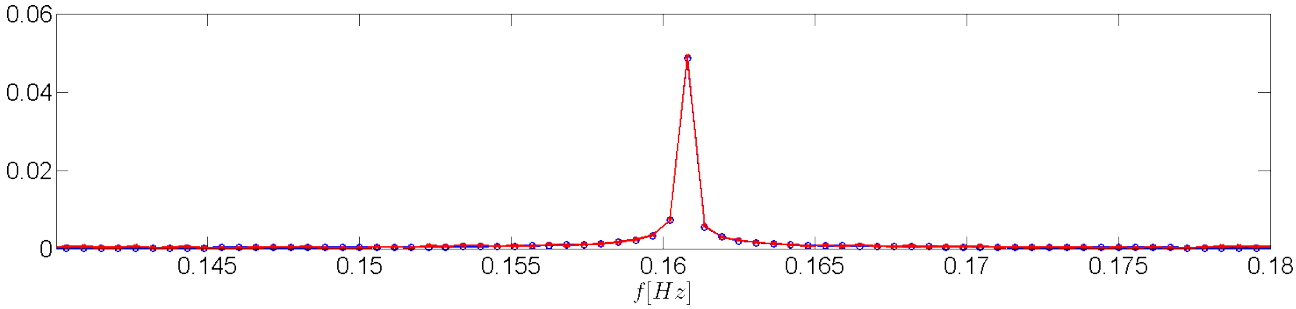


FIGURE 5.11 – Résultat de l'entraînement pour la communication par mélange non-linéaire où Bob utilise un ordinateur réservoir entraîné avec un biais plutôt que le signal d'Alice. Le spectre de l'exemple est en bleu tandis que celui reconstruit est en rouge.

Quoi qu'il en soit, plutôt que de continuer à tester de nouveaux paramètres, nous avons essayé de remplacer le biais d'entrée par $u(t) = a(t)$. Outre le fait que le protocole 2.2 devenait applicable, le signal erratique disparut pour laisser sa place au message enfin décrypté ...

Conclusions et perspectives

Nous arrivons au terme de ce travail où nous avons, pour commencer, introduit un système dynamique retardé : le système de Mackey - Glass.
Le comportement chaotique de ce dernier a été utilisé pour transmettre un message en utilisant des méthodes de communication tirées de la cryptographie par chaos. Plus précisément, nous avons introduit et appliqué les protocoles de superposition 2.1 et de mélange 2.2.
Finalement, après avoir laborieusement construit et entraîné un ordinateur réservoir, celui-ci est parvenu à décrypter un message simplifié transmis par les protocoles mentionnés ci-dessus : objectif principal atteint.

Bien entendu, tout ne s'arrête pas là : il reste des questions sans réponse et des problèmes à traiter.

▷ Tout d'abord, nous rappelons que lors de l'entraînement de l'ordinateur réservoir à l'émulation du système de Mackey - Glass (§ 4.4), nous avions observé une bonne reproduction de l'attracteur (fig. 1.5). On se demandait alors si le réservoir parvenait à imiter un système de Mackey - Glass avec un jeu de paramètres sensiblement différent que l'on avait noté $(\bar{\beta}, \bar{\gamma}, \bar{n}, \bar{\tau})$. Cette question encore ouverte mériterait une attention particulière.

▷ Ensuite, l'utilisation trop fréquente du verrouillage dans le protocole 2.1 présente deux désavantages [1] :

- **Débit** : On a montré que l'ordinateur réservoir conservait une erreur assez importante lors de l'entraînement à émuler le système de Mackey - Glass. Ceci entraînait une erreur dans le verrouillage qui désynchronisait les systèmes d'Alice et Bob. Pour pallier cela, Alice devait enlever son message régulièrement pour que Bob puisse se verrouiller à nouveau : le débit de transmission se voit diminuer.
- **Sécurité** : Si Alice envoie fréquemment son signal chaotique sans message, alors un espion pourrait peut-être s'en servir afin d'intercepter la communication.

En outre, le message que l'on a fait passer via l'utilisation du réservoir dans le protocole 2.1 avait une amplitude beaucoup plus importante que celui transmis à la section 2.2. En effet, nous avons vu que notre *reservoir computer* n'était pas plus précis que 10^{-4} dans son émulation du système de Mackey - Glass. Dès lors, décrypter un message d'amplitude 10^{-5} n'est pas envisageable pour l'instant. Cette augmentation de l'amplitude du message que requière l'utilisation du réservoir peut également constituer une faille de sécurité à prendre en compte.

Nous avons proposé une méthode alternative (protocole 5.1) n'utilisant pas le verrouillage. Celle-ci a l'avantage de ne plus demander à Alice d'envoyer son signal chaotique sans message et d'être plus robuste au bruit. Cependant, le prix à payer est la nécessité de répéter chaque bit un grand nombre de fois. Nous sommes alors confrontés à des problèmes similaires quant

1. Notons que ces deux inconvénients sont intrinsèques à la méthode et non à l'utilisation du réservoir.

au débit et à la sécurité potentiellement compromise. En effet, il est clair que si chaque bit est répété r fois, le débit sera diminué d'un facteur r. De plus, cette répétition importante pourrait peut-être permettre à un tiers d'étudier les caractéristiques du signal transmis et d'en déduire le message.

Ce serait intéressant de continuer l'investigation afin d'aboutir à une meilleure méthode en termes de débit et de sécurité. En effet, le protocole 5.1 utilisant le réservoir sans verrouillage semble être capable de casser le protocole 2.1 dans le sens où il ne serait plus nécessaire à Bob de posséder une copie du système transmetteur pour décrypter un message.

Malheureusement, avec notre matériel nous n'avons pas pu comparer les performances de ces protocoles sur des messages plus longs qui surchargeaient la mémoire vive jusqu'à l'erreur « *Out of memory* ». C'est pourquoi nous nous étions limité à 1200 bits au paragraphe 5.1.2.

▷ Finalement, en ce qui concerne la seconde méthode fondée sur le protocole 2.2, nous n'avons pas réussi à transmettre un message aussi complexe que pendant le test, c'est-à-dire modulé en basses fréquences. Nous avions alors proposé une série de questions qui permettrait peut-être de solutionner le problème. Il y a certainement de l'espoir d'y parvenir puisque nous avons réussi à décrypter, via cette même méthode, un message monochromatique. La modification de l'entraînement est également une piste à envisager. En effet, l'entraînement alternatif introduisant la rétro-action que nous avons proposé à déjà permis de pouvoir appliquer le protocole 2.2 même en présence de bruit. Peut-être qu'une autre adaptation rendrait le protocole 2.2 fonctionnel avec le message modulé en fréquence.

En conclusion, bien que les résultats que nous présentons soient assez prometteurs, nous n'affirmons pas que Bob soit capable de décrypter n'importe quel message en utilisant un ordinateur réservoir ou encore que la cryptographie par chaos soit complètement mise en défaut. En effet, il reste énormément de travail à investir dans ce domaine afin d'obtenir des réponses à ces questions.

Nous espérons tout de même, comme le dirait notre ancien professeur André FÜZFA, avoir créé un questionnement qui permettra, dans le futur, d'apporter des progrès dans ce domaine de recherche.

In fine, l'ordinateur réservoir est un système dynamique très capricieux qui pourrait bien devenir un pilier important de la cryptographie par chaos.

Annexes et compléments

Annexe A

Famille d'applications logistique

A.1 Généralités

Le choix de l'expression des T_μ a été fait de telle sorte que le paramètre μ puisse être associé à celui des Q_μ. Sans plus attendre, la forme analytique de la famille logistique :

$$\begin{aligned} Q_\mu : \ I &\rightarrow I \\ x &\mapsto Q_\mu(x) = \mu x(1-x) \end{aligned} \tag{A.1}$$

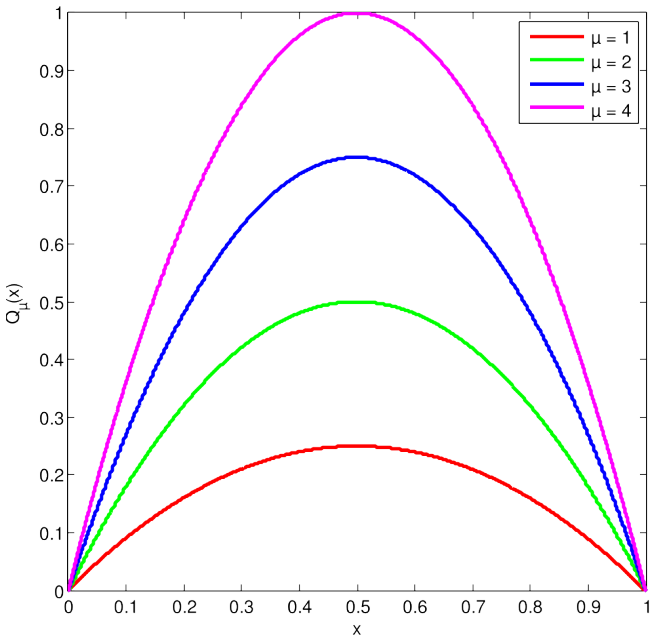


FIGURE A.1 – Quelques exemples de la famille d'applications logistique.

On retrouve ensuite les trois propriétés des T_μ :

(1) **Symétrique** : $Q_\mu(\frac{1}{2}+x) = Q_\mu(\frac{1}{2}-x)$, $\forall x \in [0, \frac{1}{2}]$.

(2) **Concave** : Q_μ est strictement croissante sur $[0, \frac{1}{2}]$ et strictement décroissante sur $[\frac{1}{2}, 1]$.

(3) **Unimodale** : Q_μ est concave et atteint son unique maximum de $\mu/4$ en $x = \frac{1}{2}$.

On se demande alors s'il existe un « changement de cordonnées » entre les familles tente et logistique. La réponse est donnée ci-après.

A.2 Conjugaison topologique et chaos

Commençons par donner la définition d'applications topologiquement conjuguées :

Définition A.1 (Conjugaison topologique)**.**

L'application $f : A \to A$ est **(topologiquement) conjuguée** à $g : B \to B$ si et seulement s'il existe un homéomorphisme $h : A \to B$ tel que :

$$h \circ f = g \circ h \tag{A.2}$$

L'homéomorphisme est appelé **conjugaison** et on note $f \overset{h}{\sim} g$ la relation de conjugaison. Si h n'est pas surjective, on parle de **semi-conjugaison**.

Notons que la réciproque d'une bijection continue entre ensembles compacts (tel que $I = [0\,,1]$, fermé borné dans $\mathbb{R}$) est toujours continue. Il suffira donc de vérifier que h est bijectif et continu pour avoir la[1] conjugaison.

De la définition A.1 découle une propriété intéressante : les orbites (voir définition 1.3) de f sont envoyées sur celles de g via h. En effet, de (A.2) on a directement $f = h^{-1} \circ g \circ h$. Il suffit alors d'itérer f pour arriver à $f^{\circ n} = h^{-1} \circ g^{\circ n} \circ h$, d'où $f^{\circ n} \overset{h}{\sim} g^{\circ n}$.

Sans surprise, on montre maintenant que T_4 est conjuguée à Q_4 :

Résultat A.1 (Conjugaison topologique entre T_4 et Q_4)**.**

Soient T_4 et Q_4 définies par (1.1) et (A.1), alors $T_4 \overset{h}{\sim} Q_4$ avec

$$h : I \to I : x \mapsto h(x) = \sin^2(\pi x/2) \tag{A.3}$$

Preuve

▷ Soit $x \in [0, \frac{1}{2}] \Longrightarrow T_4(x) = 2x$:

$$(Q_4 \circ h)(x) = 4h(x)\,(1 - h(x)) = 4\sin^2\left(\frac{\pi}{2}x\right)\cos^2\left(\frac{\pi}{2}x\right) = \sin^2(\pi x) = (h \circ T_4)(x)$$

▷ Soit $x \in [\frac{1}{2}, 1] \Longrightarrow T_4(x) = 2(1 - x)$:

$$(h \circ T_4)(x) = h(2(1 - x)) = \sin^2(\pi - \pi x) = \sin^2(\pi x) = (Q_4 \circ h)(x)$$

□

1. On peut montrer que si la conjugaison existe, alors elle est unique.

En fait, la conjugaison préserve bien plus que les orbites, comme l'affirme le résultat suivant :

Théorème A.1 (Conservation du chaos dans la conjugaison)**.**

Soit $f : A \to A$ et $g : B \to B$, deux applications (semi-)conjuguées via $h : A \to B$.
Si f est chaotique sur A (voir définition 1.8), alors g est chaotique sur B.

Preuve

(1) **Dépendance sensible aux conditions initiales** :
Soit $\delta > 0$, la constante de sensibilité de f telle que $\delta < \beta - \alpha$ avec $A \overset{def}{=} [\alpha, \beta]$.

$\forall y \in [\alpha, \beta - \delta] : |h(y + \delta) - h(y)|$ est une fonction positive et continue sur un compact.

Si on note δ^* son minimum global on a donc que h envoie des intervalles de longueur δ vers des intervalles de longueur au moins égale à δ^* (affirmation $\circledast$). Montrons que ce δ^* est la constante de sensibilité de g que l'on recherche.
Soient $y_0 \in B$ et U un intervalle ouvert contenant y_0. Il s'en suit que $h^{-1}(U)$ est un voisinage de $h^{-1}(y_0)$ dans A. Et comme f est sensible aux conditions initiales, il vient :

$$\exists x_0 \in h^{-1}(U),\ \exists n \in \mathbb{N} \text{ tels que } \left| f^{\circ n}\left(h^{-1}(y_0)\right) - f^{\circ n}(x_0) \right| > \delta$$

On arrive alors à la dépendance sensible aux conditions initiales pour g :

$$|g^{\circ n}(y_0) - g^{\circ n}(h(x_0))| \overset{(A.2)}{=} \left| h\left(f^{\circ n}\left(h^{-1}(y_0)\right)\right) - h\left(f^{\circ n}(x_0)\right) \right| \overset{\circledast}{>} \delta^*.$$

(2) **Topologiquement transitive** :
Soient U_1 et U_2, deux intervalles ouverts de B. Par continuité, $h^{-1}(U_1)$ et $h^{-1}(U_2)$ sont deux intervalles ouverts de A. Exploitons le fait que f est topologiquement transitive :

$$\exists x \in h^{-1}(U_1) \text{ et } \exists n \in \mathbb{N} \text{ tels que } f^{\circ n}(x) \in h^{-1}(U_2).$$

D'où, $h(x) \in U_1$ et par (A.2) : $g^{\circ n} \circ h(x) = h \circ f^{\circ n}(x) \in U_2 \Longrightarrow g^{\circ n}(U_1) \cap U_2 \neq \varnothing$.

(3) **Points périodiques denses** :
Soit U un intervalle ouvert de B. Par continuité, $h^{-1}(U)$ est un intervalle ouvert de A. Comme f possède des points périodiques denses dans A : $\exists x \in h^{-1}(U)$ un point périodique de période n. En appliquant la conjugaison (A.2) il vient :

$$g^{\circ n} \circ h(x) = h \circ f^{\circ n}(x) = h(x)$$

Donc $h(x)$ est un point périodique de période n dans U et on a la densité dans B. □

On termine ce chapitre annexe en combinant le résultat A.1 et le théorème A.1 pour en déduire que Q_4 est également chaotique.

Annexe B

Outils d'intégration numérique

Il arrive souvent en physique que des équations ne peuvent pas être solutionnées analytiquement. Cette annexe présente brièvement les outils numériques essentiels pour l'intégration d'équations différentielles. Nous commençons par la méthode des différence finies et, ensuite, nous donnons l'algorithme très connu de Runge et Kutta d'ordre 4. Nous terminons avec un exemple : l'intégration du système de Mackey - Glass, pour lequel nous explicitons également une approximation de la solution formelle.

Le lecteur intéressé pourra parcourir [20], [21] ou encore [24] pour découvrir ce vaste sujet.

B.1 Méthode des différences finies

La méthode des différences finies est très intuitive. Elle est basée sur la définition de la dérivée d'une fonction. Soit $y(t)$ une application suffisamment lisse, sa dérivée s'écrit :

$$\dot{y}(t) = \lim_{\Delta t \to 0} \frac{y(t+\Delta t) - y(t)}{\Delta t},$$

où l'on a introduit la notation pointée pour la dérivée temporelle. On construit une grille de pas $h > 0$ pour discrétiser le temps : $t_n = t_0 + nh$ avec $n \in \mathbb{N}$ la variable temporelle discrète. On approxime alors $\dot{y}$ au nœud n à l'aide du nœud suivant $n+1$ et on écrit :

$$\dot{y}(t_n) = \frac{y(t_{n+1}) - y(t_n)}{h} + \mathcal{O}(h) \tag{B.1}$$

C'est la **formule d'Euler progressive**. Si on exploite le nœud précédent, on obtient la **formule d'Euler rétrograde** :

$$\dot{y}(t_n) = \frac{y(t_n) - y(t_{n-1})}{h} + \mathcal{O}(h)$$

Afin de gagner en précision, on peut utiliser simultanément les informations aux nœuds suivant et précédant en moyennant les formules progressive et rétrograde. On obtient la **différence centrée** :

$$\dot{y}(t_n) = \frac{y(t_{n+1}) - y(t_{n-1})}{2h} + \mathcal{O}(h^2)$$

L'article de Fornberg [23] fournit les coefficients à utiliser pour les dérivées d'ordre supérieur (jusque 4) ainsi que pour des niveaux plus fins de précision.

Afin d'illustrer notre propos, nous appliquons cette méthode sur un exemple au paragraphe B.3.1.

B.2 Méthode de Runge et Kutta

Considérons un problème de Cauchy pour $y(t)$:

$$\begin{cases} \dot{y}(t) = f(t, y(t)) \\ y(t_0) = y_0 \end{cases} \tag{B.2}$$

De nouveau, on suppose la fonction y suffisamment lisse sur l'intervalle $I \subset \mathbb{R}$ que parcourt t. Dès lors, on peut montrer l'existence et l'unicité d'une solution locale. Celle-ci s'écrira formellement comme :

$$y(t) = y_0 + \int_{t_0}^{t} f(s, y(s))ds$$

Maintenant, on discrétise l'intervalle d'intégration en $N+1$ nœuds et on note le pas h :

$$t_n = t_0 + nh, \quad n \in \{0, 1, \cdots, N\}$$

Pour simplifier les notations, on définit pour chacun de ces nœuds :

$$y_n = y(t_n) \; ; \; f_n = f(t_n, y_n)$$

Ceci permet de construire un schéma d'intégration générique calculant successivement y_n en chaque nœud depuis la condition initiale connue :

$$y_{n+1} = y_n + hF(t_i, y_i, f_i; h) + \mathcal{O}(h^{\theta+1}),$$

où F est la **fonction d'incrément** caractérisant le schéma d'intégration d'ordre θ utilisé. Le plus fréquent est celui d'ordre 4, donné ci-après :

$$y_{n+1} = y_n + h\left(\frac{K_1 + 2K_2 + 2K_3 + K_4}{6}\right) + \mathcal{O}(h^5) \tag{B.3}$$

avec

$$\begin{cases} K_1 = f_n \\ K_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}K_1) \\ K_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}K_2) \\ K_4 = f(t_{n+1}, y_n + hK_3) \end{cases}$$

On note généralement RK4 cet algorithme d'intégration d'ordre 4 en l'honneur des mathématiciens C. Runge et M. W. Kutta.

Remarquons que si l'on prend $F(t_i, y_i, f_i; h) = f_n$ comme fonction d'incrément, c'est-à-dire :

$$y_{n+1} = y_n + hf_n,$$

on retrouve la formule d'Euler progressive (B.1).

B.3 Intégration du système de Mackey - Glass

Cette section est dédiée à la résolution numérique de l'équation (1.9). Il faut savoir que toutes les méthodes d'intégration ne donnent pas exactement les mêmes résultats. Nous illustrons cette affirmation par un exemple d'utilisation des outils que nous avons présentés, fournissant de ce fait un complément pour le texte principal qui fait usage à maintes reprises du système de Mackey - Glass. Nous rappelons donc l'équation à résoudre :

$$\dot{y}(t) = -\gamma y(t) + \beta \frac{y_\tau(t)}{1 + y_\tau^n(t)}, \ \beta, \gamma, n > 0 \ ,$$

où l'on a posé $y_\tau(t) = y(t - \tau)$ avec le retard $\tau \in \mathbb{R}_0^+$.

La solution la plus simple est de demander gentiment à MATLAB® de résoudre le problème en utilisant la fonction dde23. Le résultat ainsi obtenu se trouve à la figure B.1, en rouge. Nous renvoyons le lecteur vers [22] pour plus de détails sur cet intégrateur. Voyons plutôt comment appliquer les notions que l'on a présentées aux sections B.1 et B.2.

B.3.1 Différences finies

Nous précisons au lecteur que cette méthode est la seule qui n'a pas été utilisée dans nos codes pour le système de Mackey - Glass. Dès ce fait, ce paragraphe n'a d'autre but que d'illustrer la méthode des différences finies sur un exemple concret.

On commence par exprimer (1.9) comme suit :

$$\frac{y(t+h) - y(t)}{h} = -\gamma y(t) + \beta \frac{y_\tau(t)}{1 + y_\tau^n(t)} + \mathcal{O}(h),$$

où $h > 0$ sera le pas de discrétisation du temps. En réarrangeant les termes, on arrive aisément à l'expression suivante :

$$y(t+h) = (1 - h\gamma)y(t) + h\beta \frac{y(t-\tau)}{1 + y^n(t-\tau)} + \mathcal{O}(h)$$

Si on note n le temps discret [1], alors il vient la récurrence suivante :

$$y_{n+1} = (1 - h\gamma)y_n + h\beta \frac{y_{n-\frac{\tau}{h}}}{1 + y_{n-\frac{\tau}{h}}^n},$$

où l'on a laissé tomber les termes d'ordre supérieur. Finalement, on a refait le raisonnement qui nous avait conduit à (B.1). Il ne reste plus qu'à implémenter cette relation pour obtenir la courbe magenta à la figure B.1.

1. La notation est un peu dangereuse lorsque l'on est pas habitué : ne pas confondre le temps discret avec le paramètre n apparaissant dans l'équation (1.9) !

B.3.2 Runge - Kutta 4

On identifie le champ de vitesse présent dans (B.2) à celui de (1.9) :

$$f(t,y) = -\gamma y(t) + \beta \frac{y_\tau(t)}{1+y_\tau^n(t)} \equiv f(y)$$

Notons que la dépendance en t n'est pas explicite : cela simplifiera l'utilisation de (B.3). Comme pour les différences finies[2], on discrétise avec $h > 0$ et il vient :

$$f_n = -\gamma y_n + \beta \frac{y_{n-\frac{\tau}{h}}}{1+y_{n-\frac{\tau}{h}}^n}$$

Nous sommes également en mesure de calculer les K_i apparaissant dans (B.3). Comme la dépendance en t n'est pas explicite dans f, on a plus simplement :

$$\begin{cases} K_1 = f(y_n) \\ K_2 = f(y_n + \frac{h}{2}K_1) \\ K_3 = f(y_n + \frac{h}{2}K_2) \\ K_4 = f(y_n + hK_3) \end{cases}$$

En injectant le tout dans l'équation (B.3), cette dernière devient bien, *in fine*, une relation de récurrence que l'on peut implémenter pour obtenir la courbe bleue à la figure B.1.

B.3.3 Approximation de la solution formelle

Le raisonnement suivant présente une approximation de la solution pour le système de Mackey - Glassque nous avons exploitée pour intégrer numériquement le système (2.8).

Afin de simplifier les calculs, nous considérons l'expression (2.6) plutôt que (1.9). Ainsi, ce sera seulement à la fin que l'on choisira ε_0 et $f[y(t-\tau_0)]$ pour retrouver (1.9) (voir paragraphe 2.3.1). En outre, on écrit simplement $f(t)$ plutôt que $f[y(t-\tau_0])$ pour ne pas alourdir les équations. Fort de ces détails pré-calculatoires, on doit donc résoudre :

$$\varepsilon_0 \dot{y}(t) + y(t) = f(t),$$

entre $t_0 \in \mathbb{R}$ et $t \geq t_0$ sachant que $y(t_0) = y_0$. La solution homogène est donnée par :

$$y_H(t) = y_0 e^{-\frac{t-t_0}{\varepsilon_0}}$$

La variation des constantes aboutit à une solution particulière de la forme :

$$y_P(t) = e^{-\frac{t-t_0}{\varepsilon_0}} \int_0^{t-t_0} \frac{e^{\frac{s}{\varepsilon_0}}}{\varepsilon_0} f(s+t_0)\, ds$$

2. La notation est un peu dangereuse lorsque l'on est pas habitué : ne pas confondre le temps discret avec le paramètre n apparaissant dans l'équation (1.9) !

La combinaison de ces résultats nous fournit la solution de notre problème :

$$y(t) = \left(y_0 + \int_0^{t-t_0} \frac{e^{\frac{s}{\varepsilon_0}}}{\varepsilon_0} f(s+t_0)\ ds \right) e^{-\frac{t-t_0}{\varepsilon_0}}$$

Si maintenant on discrétise le temps avec un pas $h > 0$, entre le nœud $t-h$ et le nœud t on peut écrire la relation suivante à partir de notre solution pour approximer la valeur de $y(t)$:

$$y(t) \approx \left(y(t-h) + \int_0^{h} \frac{e^{\frac{s}{\varepsilon_0}}}{\varepsilon_0} f(s+t-h)\ ds \right) e^{-\frac{h}{\varepsilon_0}}$$

On peut supposer $f(s+t-h)$ constante et valant $f(t-h)$ sur le petit intervalle $[0\,,h]$. Ceci nous permet de sortir $f(t-h)$ de l'intégrale et d'arriver à :

$$y(t) \approx \left(y(t-h) + f(t-h) \int_0^{h} \frac{e^{\frac{s}{\varepsilon_0}}}{\varepsilon_0}\ ds \right) e^{-\frac{h}{\varepsilon_0}}$$

En calculant analytiquement l'intégrale et en distribuant l'exponentielle on termine avec notre approximation de la solution formelle :

$$y(t) \approx y(t-h)e^{-\frac{h}{\varepsilon_0}} + f(t-h)\left(1 - e^{-\frac{h}{\varepsilon_0}}\right)$$

Finalement, en introduisant la variable temporelle discrète propice à l'implémentation, il vient :

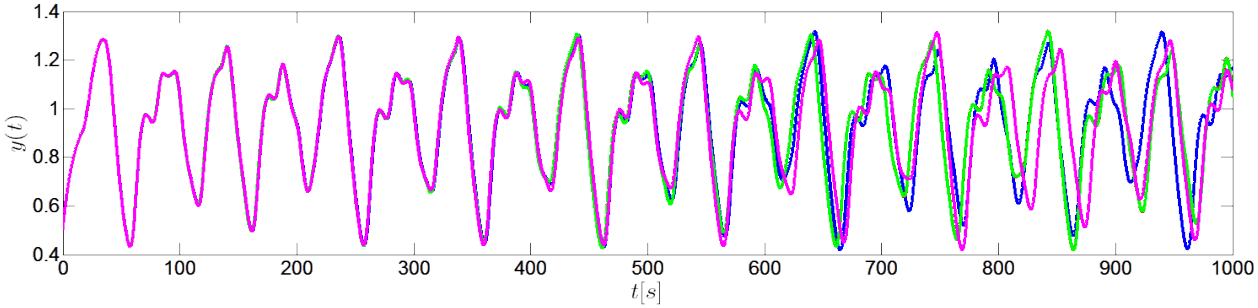
$$y_{n+1} = y_n e^{-\frac{h}{\varepsilon_0}} + f(t_n)\left(1 - e^{-\frac{h}{\varepsilon_0}}\right)$$

En se ramenant à l'équation (1.9), on obtient la courbe verte de la figure B.1.

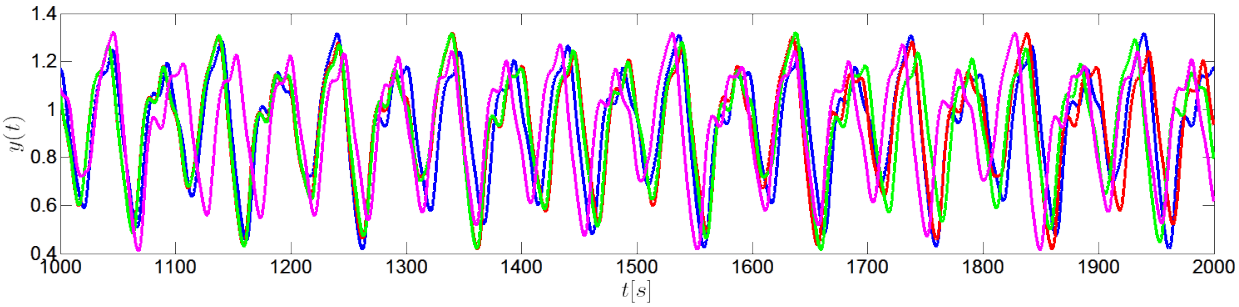
Les courbes qui sont données à la figure B.1 ont été intégrées avec $h = 0.1$, le jeu de paramètres « habituel » $(0.2, 0.1, 10, 17)$ et l'histoire $\mathbb{S} \equiv 0.5$. Elles sont identifiées comme suit :

- ▷ Runge et Kutta d'ordre 4 en bleu ;
- ▷ Fonction `dde23` fournie par MATLAB® en rouge ;
- ▷ Approximation de la solution formelle en vert ;
- ▷ Différences finies en magenta.

Les résultats obtenus pour chaque méthode ne coïncident pas parfaitement. Au début les courbes sont bien superposées (figure B.1a) tandis qu'à la fin les différentes solutions numériques diffèrent l'une de l'autre (figure B.1b). En effet, chaque méthode correspond à une approximation différente de la solution. De plus, comme la valeur de cette dernière au nœud n dépend de sa valeur aux nœuds précédents, l'accumulation des erreurs n'est pas la même pour chaque cas et les courbes divergent.



(a) $t \in [0\,,1000]$



(b) $t \in [1000\,,2000]$

FIGURE B.1 – Comparaison des méthodes d'intégration sur le système de Mackey - Glass. Voir texte pour le code couleur.

Annexe C

Neurone formel et intégrateur à fuite

Cette annexe vient compléter la notion de neurone formel introduite implicitement en 3.1 et la discussion sur les réseaux de neurones intégrateurs à fuite faite à la section 3.2.

C.1 Neurone formel

Le paradigme du neurone formel a été proposé par W. S. McCulloch, W. Pitts en 1943 [9]. Le modèle de l'époque était un modèle binaire simple : le neurone reçoit une entrée $u \in \mathbb{R}^K$ et répond par $x = 0$ ou $x = 1$ de la manière suivante :

$$x = H(W_{in}u - \omega_0),$$

où H est la fonction en escalier de Heaviside, $W_{in} \in \mathbb{R}^{1\times K}$ les **poids synaptiques** et $\omega_0 \in \mathbb{R}$ le **seuil d'activation**.

L'utilisation de la **fonction d'activation** H, caractéristique du modèle binaire, permet de justifier la terminologie du paramètre ω_0. En effet, l'image par H vaut 0 tant que son argument est strictement négatif et 1 sinon. En d'autres termes, le neurone n'est activé que lorsque l'entrée, pondérée par les poids synaptiques, franchit le seuil d'activation :

$$x = \begin{cases} 1 \text{ si } W_{in}u \geq \omega_0 \\ 0 \text{ sinon} \end{cases}$$

Dans le cadre de notre travail, nous avons utilisé ce modèle en modifiant le seuil et la fonction d'activation. Le seuil a été fixé à 0 partout et les fonctions d'activation pour les neurones d'entrée et de sortie prennent différentes formes qui sont données au chapitre 3. Notons que la modification de celles-ci a enlevé le caractère binaire des neurones.

C.2 Intégrateur à fuite

Le modèle de neurone intégrateur à fuite permet d'apporter une dynamique supplémentaire. En effet, le neurone formel décrit plus haut reçoit une entrée et répond indépendamment de sa valeur précédente. Nous expliquons brièvement comment introduire cette « mémoire ».

Considérons le problème de Cauchy suivant pour la fonction réelle $x(t)$:

$$\begin{cases} \dot{x}(t) = -ax(t) + k \\ x(0) = x_0 \end{cases} \tag{C.1}$$

où $a > 0$ est le **taux de fuite** et $k \in \mathbb{R}$ une constante.

On trouve la solution générale du problème homogène, notée x_H, en annulant k :

$$\dot{x}_H(t) = -ax_H(t) \Longrightarrow x_H(t) = x_0 e^{-at}$$

Utilisons la méthode de variation des constantes pour trouver une solution particulière x_P, soit :

$$x_P(t) = X_0(t)e^{-at} \Longrightarrow \dot{x}_P(t) = \dot{X}_0(t)e^{-at} - aX_0(t)e^{-at}$$

En l'injectant dans l'équation de départ on obtient une équation à résoudre pour $X_0(t)$:

$$\dot{X}_0(t)e^{-at} = k \Longrightarrow X_0(t) = \frac{k}{a}\left(e^{at} - 1\right) + x_0$$

Et on a la solution particulière :

$$x_P(t) = \frac{k}{a}\left(1 - e^{-at}\right)$$

Finalement, la solution du problème de Cauchy (C.1) est donnée par $x = x_H + x_P$, c'est-à-dire :

$$x(t) = x_0 e^{-at} + \frac{k}{a}\left(1 - e^{-at}\right)$$

De manière générale, cette solution (dé)croît exponentiellement au taux a depuis $x(0)$ vers k/a, où il y a stabilisation. Ce comportement est illustré à la figure C.1 dans les deux cas.

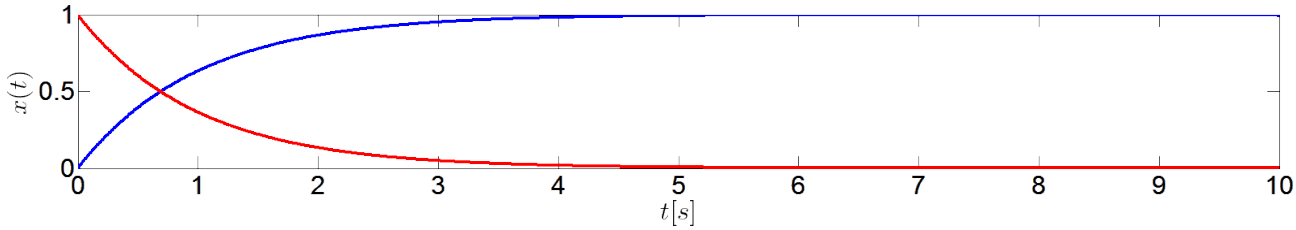


FIGURE C.1 – Solutions de l'intégrateur à fuite avec $a = 1$. En bleu $k = 1$ et $x(0) = 0$, en rouge $k = 0$ et $x(0) = 1$.

Si maintenant k prenait une valeur non-nulle jusque $t = T$ et tombait à 0 ensuite, alors on observerait le comportement suivant :

- ▷ $t \leq T$: le système va progressivement oublier son état initial pour tendre vers $k/a \overset{def}{=} \tilde{k}$;
- ▷ $t \geq T$: le système va progressivement oublier $\tilde{k}$ pour tendre vers 0.

Ce nouveau système dispose ainsi d'une certaine « mémoire » régulée par le taux de fuite a.

Au paragraphe 3.2 nous adaptons l'équation (C.1) pour obtenir (3.3) décrivant l'évolution du réseau de neurones intégrateurs à fuite.

Annexe D

Description des fichiers de codes

Le lecteur peut télécharger l'ensemble des codes en utilisant l'hyperlien suivant :
https://github.com/mgulina/ReservoirComputing/raw/master/RC_11052017.zip
qui vous conduira vers l'archive `RC_11052017.zip` d'environ 40 Mo.

D.1 Répartition

L'ensemble de nos fichiers de codes MATLAB® se scinde en cinq dossiers :

(1) **`Prédiction`** : contient trois exemples de systèmes émulés. Nous avons présenté la fonction périodique quelconque (§ 4.3.2) et le système de Mackey - Glass (§ 4.4) mais pas le système de Lorenz sur lequel nous avions également travaillé quelques temps.

(2) **`Verrouillage`** : contient simplement les programmes utilisés pour le verrouillage ;

(3) **`Cryptographie`** : contient les deux méthodes discutées, la superposition et le mélange ;

(4) **`Autres`** : contient deux exemples de tâches. La première est la classification (§ 4.3.1) tandis que la seconde, passée sous silence ici, est l'égaliseur de canaux) ;

(5) **`Subs communs`** : regroupe un bon nombre de petits scripts appelés à plusieurs endroits (fichiers de paramétrisations, de calcul d'erreur, générateur de messages, etc ...) .

D.2 Fichiers de paramétrisation

Nous avons utilisé énormément de *scripts* afin de globaliser nos variables sans devoir les déclarer partout. Cette méthode de travail permet également à l'utilisateur de lancer les programmes principaux qui utiliseront nos scripts avec des valeurs par défaut.

- ▷ Si les variables suivantes sont fixées *avant* `cibleXXX.m`, celles-ci remplacent les valeurs par défaut présentes dans le script correspondant : `CI, h, T_tot`. En outre, `cibleMGXXX.m` proposent aussi la modification des variables `tau` et `ChangeScaleMG` qui leur sont propres.
- ▷ Si les variables suivantes sont fixées *avant* `GenResXXX.m`, celles-ci remplacent les valeurs par défaut présentes dans le script correspondant : `N, rho, gainIn, gainFb, delta, C, a, LvlNoise`.

D.3 Prédiction

L'équation d'évolution du réservoir est contenue dans le fichier `majRes.m`. Pour lancer un entraînement, faites appel aux fichiers `mainPredictXXX.m`. Ces derniers démarrent plusieurs sous-routines, dans l'ordre :

(1) La fonction d'apprentissage est construite dans les fichiers `cibleXXX.m` ;

(2) Les paramètres de l'équation d'évolution et du réservoir sont définis dans les fichiers `genResXXX.m` ;

(3) Le nombre de points rejeté (transient), d'entraînement ou pour l'évolution libre sont posés dans les fichiers `train(Version)XXX.m` où « Version » n'est pas toujours présent mais peut apparaître avec les valeurs « Simple » et « Avance » [1] ;

(4) L'évaluation des erreurs pendant l'entraînement et l'évolution libre est réalisée par le fichier `calcErreursTrain.m` ;

(5) Pour tous les systèmes `XXX`, un test du caractère chaotique peut être fait via l'activation de `testChaos.m` ;

(6) Pour Mackey - Glass, le fichier `testAttracteursMG.m` permet de comparer les attracteurs cible et prédit.

D.4 Verrouillage

Le fichier `mainLock.m` invite l'utilisateur à choisir parmi quatre scripts :

(1) Verrouillage d'un système de Mackey - Glass sur un autre (`MGLockMG.m`) ;

(2) Verrouillage d'un MG sur un réservoir entraîné pour imiter un système de Mackey - Glass (`MGLockRC.m`) ;

(3) Verrouillage d'un réservoir entraîné pour imiter un système de Mackey - Glass sur un système de Mackey - Glass (`MGLockRC.m`) ;

(4) Verrouillage d'un réservoir entraîné pour imiter un système de Mackey - Glass sur un autre réservoir (`RCLockRC.m`).

La variable $q \in [0\,,1]$ conditionnant la proportion de verrouillage est définie séparément *dans* ces fichiers. À noter que `MGLockMG.m` peut être lancé directement sans passer par le programme principal. Pour les autres cas, le fichier correspondant peut être lancé directement (même après modification de q par exemple) sans passer par le programme principal si ce dernier a déjà tourné une fois pour ce cas avant.

1. La méthode « Avance » est obsolète et doit être mise à jour avant d'être utilisable.

D.5 Cryptographie par chaos

L'interception de messages transmis par un algorithme basé sur le chaos se fait via l'appel de `mainCrypto.m`. Ce fichier propose à l'utilisateur nos deux méthodes de communication : la superposition (`superposition.m`) et le mélange non-linéaire (`melange.m`). Ci-dessous, les paramètres pouvant être modifiés directement *dans* ces deux fichiers :

- ▷ Superposition : `h, bitRepete, nbrBitLock, nbrBit, nbrBitTrain, A, A_eps` ;
- ▷ Mélange : `nbrSinus, omega, h, B, fc_ex, fm_ex, fc_msg, fm_msg, inputFactor, tau, A_filtre, A_eps`.

En outre, ces deux routines font appel aux programmes précédents : leurs paramètres peuvent alors également être modifiés.

Chacune des méthodes est utilisée par Bob et par Eve. Dans les codes, Bob dispose d'une copie du système d'Alice alors que Ève est munie d'un ordinateur réservoir.

Bibliographie

Avant de parler des ouvrages sur lesquels notre travail est basé, nous nous devons de *rendre à César ce qui est à César* (verset Matthieu 22:21 de [25]). En effet, l'image ornant notre page de couverture a été prise chez *Caroline Davis2010* à l'adresse www.flickr.com/photos/53416677@N08/4972916707. Bien que le titre original parle d'une molécule, nous avons adapté cette photo pour représenter, de manière un peu abstraite, l'ordinateur réservoir.

Systèmes dynamiques

Cette première section de la bibliographie référence les différents documents à la base du chapitre 1 et de l'annexe A.

▷ Articles

[1] J. GUO, *Analysis of chaotic systems*, The University of Chicago Mathematics REU, 2014.
Nous nous sommes basés sur ce document pour établir le résultat 1.2 et le théorème A.1.

[2] M. MACKEY & L. GLASS, *Mackey-Glass equation*, Scholarpedia, 2010.
DOI : 10.4249/scholarpedia.6908.
Cet article représente sans doute la meilleure introduction au système de Mackey - Glass. Nous recommandons de commencer par celle-ci et de poursuivre avec les références qui s'y trouvent.

▷ Notes de cours

[3] T. CARLETTI, *Chaos et fractales*, Université de Namur, 2016.
Syllabus d'un cours dispensé à l'Université de Namur que l'auteur a suivi, ces notes ont tout à fait contribué à la réalisation du chapitre 1 et de l'annexe A.

[4] J. BRICMONT, *Introduction à la dynamique non linéaire*, Université Catholique de Louvain, 2009.
Ce document-ci contient une excellente bibliographie pour le lecteur curieux et, bien qu'il soit rigoureux, il donne une bonne intuition du chaos tel que donné par la définition 1.8.

[5] N. CHEVALLIER, *Introduction aux systèmes dynamiques*, Université de Haute Alsace, 2011.
Moins agréable à lire pour un physicien que [4], il n'en est pas moins bien construit. Le chapitre sur la dynamique topologique permet de mieux appréhender la notion de transitivité introduite dans la définition 1.8 du chaos.

▷ Livre

[6] C. MISBAH, *Dynamiques complexes et morphogénèse : Introduction aux sciences non lineaires* (chapitre 8), Springer, 2011.
Cet ouvrage traite du chaos et de l'exposant de Liapounov au chapitre 8.

Cryptographie par chaos

Les articles donnés ci-dessous nous ont permis de développer le chapitre 2.

▷ Articles

[7] L. M. PECORA & T. L. CARROLL, *Synchronization in chaotic systems*, Phys Rev Lett 1990 ;64 :821–4. DOI : 10.1103/PhysRevLett.64.821.
Article qui introduit la notion de synchronisation entre deux systèmes de Lorenz. Inspirés par cette idée, nous avons pu développer le verrouillage à la section 2.2.

[8] M. D. PROKHOROV & V. I. PONOMARENKO, *Encryption and decryption of information in chaotic communication systems governed by delay-differential equations*, Chaos, Solitons and Fractals 35 (2008) 871–877. DOI : 10.1016/j.chaos.2006.05.081.
Ces deux messieurs nous présentent un moyen de communication basé sur le chaos engendré par un système dynamique à retard. En outre, ils montrent comment ils parviennent à récupérer un message caché. La section 2.3 résume une partie de cet article.

Ordinateur réservoir

Nous avons rassemblé ici les documents sous-jacents aux chapitres 3 et 4 ainsi qu'à l'annexe C.

▷ Articles

[9] W. S. MCCULLOCH, W. PITTS, *A Logical Calculus of the Ideas Immanent in Nervous Activity*, Bulletin of Mathematical Biophysics, vol. 5, pp. 115-133, 1943. DOI : 10.1007/BF02478259.
Article important dans la théorie des réseaux neuronaux, il est celui qui a introduit le paradigme biologique du neurone formel.

[10] S. GROSSBERG, *Recurrent neural networks*, Scholarpedia, 2013. DOI : 10.4249/scholarpedia.1888.
Chercheur en sciences cognitives de la Boston University, Grossberg propose ici un article d'introduction aux *Recurrent neural networks* et offre toute une zoologie de références à parcourir.

[11] H. JAEGER, *Echo state network*, Scholarpedia, 2007. DOI : 10.4249/scholarpedia.2330.
Professeur à la Jacobs University en Allemagne, Jaeger est spécialisé dans l'étude des systèmes non-linéaires stochastiques. L'article ci-dessus est une bonne introduction aux *echo state networks* référençant en particulier tous les articles qui suivent, sauf le dernier qui est plus particulier.

[12] H. JAEGER & H. HAAS, *Harnessing nonlinearity : predicting chaotic systems and saving energy in wireless communication*. Science **304**, 78-80 (2004). DOI : 10.1126/science.1091277.
Cet article (accompagné de son supplément) nous a permis de développer nos propres codes MATLAB® pour la conception et l'entraînement d'un ordinateur réservoir. En particulier, le lecteur y retrouvera la classification de signaux et l'émulation du système de Mackey - Glass.

[13] H. JAEGER, M. LUKOSEVICIUS, D. POPOVICI, et U. SIEWERT, *Optimization and applications of echo state networks with leaky-integrator neurons*. Neural Networks, 20(3) :335 – 352, 2007.
Dédié à l'étude des *echo state networks* basés sur les réseaux de neurones intégrateurs à fuite.

[14] H. JAEGER, *The "echo state" approach to analysing and training recurrent neural networks - with an Erratum note*. GMD Report 148 (2010).
Ce document, encore de Jaeger, est le corrigé d'un rapport publié en 2001. Il définit formellement les réseaux d'état d'écho et montre également comment s'en servir en émulant le système de Mackey - Glass. Outre cela, les mélomanes trouveront aussi leur bonheur ...

[15] G. MANJUNATH & H. JAEGER, *Echo State Property Linked to an Input : Exploring a Fundamental Characteristic of Recurrent Neural Networks*, Neural Computation 2013 25 :3, 671-696.
DOI : 10.1162/NECO_a_00411.
Article plus récent sur les ordinateurs réservoirs duquel nous avons repris la définition 3.7.

[16] Y. PAQUOT et al., *Optoelectronic Reservoir Computing*. Scientific Reports **2** :287 (2012).
DOI : 10.1038/srep00287.
Cet article co-écrit par notre directeur de mémoire propose une implémentation physique d'un *reservoir computer* en combinant l'optique et l'électronique. Les suppléments qui accompagnent ce document nous ont permis de créer rapidement notre propre réservoir numérique.

▷ Notes de cours

[17] H. JAEGER, *Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the "echo state network" approach*, GMD Report 159 (2002).
Ce document constitue les notes d'un cours de 5h dispensé au *Fraunhofer Institute for Autonomous Intelligent Systems* et destiné, entre autre, à introduire l'approche d'*echo state network* sur les *recurrent neural networks*.

▷ Livres

[18] C. ELIASMITH, C. H. ANDERSON, *Neural Engineering Computation, Representation, and Dynamics in Neurobiological Systems*, MIT Press, 2003.
Ouvrage assez imposant plutôt orienté dans l'implémentation physique. Il contient, en particulier, les descriptions du neurone avançant et à taux de fuite.

[19] M. LUKOSEVICIUS, *A Practical Guide to Applying Echo State Networks*.
DOI : 10.1007/978-3-642-35289-8_36.
Chapitre 27 de la seconde édition du livre *Neural Networks : Tricks of the Trade* édité chez Springer en 2012, le lecteur y trouvera un excellent guide sur la paramétrisation d'un *echo state network* dont nous avons repris le strict minimum au paragraphe 3.4. En outre, Lukosevicius introduit la régularisation de Tikhonov aussi appelée régression d'arête (*ridge regression*).

Intégration numérique

Cette dernière section de la bibliographie regroupe les quelques documents qui nous ont permis d'élaborer l'annexe B.

▷ Notes de cours

[20] A. FUZFA, *Introduction aux algorithmes mathématiques et au calcul scientifique*, Presses Universitaires de Namur, 2012.
Tout comme [3], ce syllabus a servi de support lorsque l'auteur a suivi le cours dispensé à l'Université de Namur. Comme le laisse sous-entendre le titre, les algorithmes d'intégration numérique y sont présentés, en particulier les formules d'Euler et de Runge - Kutta .

[21] E. GONCALVÈS, *Résolution numérique, discrétisation des EDP et EDO*, Institut National Polytechnique De Grenoble, 2005.
Ces notes sont assez agréables à lire. Après une introduction historique, l'auteur nous expliquera comment résoudre numériquement des équations différentielles à l'aide, entre autre, des différences finies et de l'algorithme de Runge - Kutta.

[22] L.F. SHAMPINE, S. THOMPSON, J. KIERZENKA, *Solving Delay Differential Equations with* `dde23`, 2002.
Bien que ce ne soit pas vraiment un cours, ce tutoriel trouve sa place ici pour sa pédagogie.

▷ Article

[23] B. FORNBERG, *Generation of Finite Difference Formulas on Arbitrarily Spaced Grids*, Mathematics of Computation, vol. 51 num. 184, 1988. DOI : 10.1090/S0025-5718-1988-0935077-0.
Le *must have* lorsque l'on fait des différences finies, ce petit article a fait (et fera encore) beaucoup d'heureux en fournissant les coefficients à utiliser dans les formules d'approximation de dérivées à une dimension.

▷ Livre

[24] J. STOER & R. BULIRSCH, *Introduction to Numerical Analysis Second Edition*, Springer (1992).
Cette brique de papier raconte beaucoup d'histoires concernant le calcul numérique. Nous avons pu trouver au chapitre 7 celle des approximations d'Euler et de Runge - Kutta.

Autre

[25] *La Sainte Bible*, version Louis Segond, 1910. Ouvrage consulté sur internet le 15 mai 2017 à l'adresse : www.enseignemoi.com/bible/lire-la-bible.html