Optimisation des cinématiques

Le modèle tel qu'il a été défini dans le chapitre précédent prend en entrée les différents angles repérant la position des ailes en fonction du temps. Dans le cas le plus général, le vecteur d'entrée aura donc huit composantes indépendantes :

$$U(t) = {}^t(\xi_1(t)\ \xi_2(t)\ \lambda_1(t)\ \lambda_2(t)\ \mu_1(t)\ \mu_2(t)\ \nu_1(t)\ \nu_2(t)) \tag{3.1}$$

On a vu qu'il était toutefois possible de réduire ce nombre en supposant le mouvement des ailes symétrique, et en négligeant certains déplacements pour des configurations de vol particulières. Ainsi, le cas de vol le plus souvent considéré lors de cette étude est celui du vol vibrant longitudinal (incluant le cas particulier du vol stationnaire). L'hypothèse de vol vibrant signifie d'une part que $\mu \equiv 0$ (cf. éq. (2.3) p. 67), et d'autre part que ξ reste voisin de $\pi/2$. Le vol longitudinal est quant à lui défini en mécanique du vol par des déplacements confinés dans un plan vertical, qui tient également lieu de plan de symétrie pour l'engin [Boiffier 1998]. Sous ces deux hypothèses, les degrés de liberté des ailes du microdrone ne sont alors plus définis que par les variables $\lambda_1 = -\lambda_2 \triangleq \lambda$ et $\nu_1 = \nu_2 \triangleq \nu$.

Pour la validation d'*OSCAB* par comparaison avec les résultats expérimentaux publiés par l'université de Berkeley, les cinématiques des ailes ont naturellement été choisies de façon à reproduire les mouvements de *Robofly*. Mais en dehors de ces essais, le choix de la forme de ces signaux reste totalement libre, la seule exigence étant bien évidemment que ceux-ci soient périodiques et que leur amplitude ne dépasse pas un certain seuil : l'amplitude du battement en particulier ne saurait excéder 180 ° pour des raisons physiques.

Si la morphologie d'*OSCAB* était calquée sur un animal volant précis, il serait légitime d'adopter également dans la simulation les cinématiques de l'espèce en question. On est en effet en droit de supposer que, suite à l'évolution et l'adaptation fonctionnelle qu'ont connues ces animaux[1], leurs mouvements d'ailes sont définis « au mieux », c'est-à-dire qu'ils maximisent l'efficacité en termes d'efforts et de manœvrabilité tout en réduisant les besoins énergétiques nécessaires.

Or les dimensions du microdrone virtuel représenté par *OSCAB* ne sont pas définies de manière exacte : comme on l'a vu, l'objet du PRF *REMANTA* tient davantage en une étude ouverte dans le domaine des micro-engins, et il est donc prématuré à ce stade de l'étude de souhaiter travailler sur une configuration figée. Par conséquent, on ignore si

[1] À ce sujet, rappelons que les insectes font partie des plus anciennes espèces vivantes présentes sur Terre, et ont ainsi prouvé leur résistance et leur faculté d'adaptation à leur environnement [Grassé *et al.* 1961, Clarkson 1986].

les cinématiques arbitrairement définies jusqu'à présent constituent une manière optimale pour l'engin de battre des ailes.

La problématique posée peut alors s'énoncer de la manière suivante : étant donné une configuration particulière, comment faire battre les ailes du microdrone dans le but d'obtenir les meilleures performances ? Cette question se ramène intuitivement à un problème d'optimisation, lequel suppose la définition d'un critère à maximiser ou à minimiser. De manière triviale, on pourra poser dans un premier temps celui-ci comme égal à la portance moyenne en vol stationnaire. En effet, l'observation est l'une des applications visées avec ce nouveau concept. Il sera donc intéressant de maximiser la charge utile (capteurs, caméras visible et infrarouge, etc. [Szmelter et Żbikowski 2000]) que peut soulever l'engin en vol stationnaire. En remarquant que le poids total du microdrone (poids de la charge utile + poids propre) est équilibré par la portance dans le cas du vol stationnaire, on pourra donc se ramener à maximiser celle-ci.

3.1 Position du problème - Modélisation des signaux d'entrée

Ce problème d'optimisation est singulier car il nécessite *a priori* la recherche d'une fonction optimale : en appelant J le critère à minimiser[2] et $\mathcal{C}_T^\infty$ l'ensemble des fonctions réelles lisses périodiques de période T, on cherche :

$$(\lambda^*, \nu^*) = \arg \min_{(\lambda,\nu)\in(\mathcal{C}_T^\infty)^2} J(\lambda, \nu) \tag{3.2}$$

Les outils permettant de résoudre de manière analytique ce type de problème sont relativement rares en mathématiques, et souvent applicables uniquement à des « cas d'école » très simples et peu représentatifs de véritables modèles physiques [Cherruault 1999]. On ne s'attardera pas non plus sur les méthodes basées sur le principe du maximum de Pontryagin, dont le domaine d'application reste la commande optimale des processus sous contraintes [Boudarel *et al.* 1969]. Il semble donc nécessaire de devoir se ramener d'un problème d'*optimisation fonctionnelle* à un problème plus classique d'*optimisation paramétrique* :

$$(\Theta_1^*, \Theta_2^*) = \arg \min_{(\Theta_1,\Theta_2)\in(\mathbb{R}^n)^2} J(\lambda_{\Theta_1}, \nu_{\Theta_2}) \tag{3.3}$$

$$\text{avec} \quad \lambda_{\Theta_1}(t) = \lambda(\Theta_1(1), \ldots, \Theta_1(n), t) \tag{3.4}$$

$$\nu_{\Theta_2}(t) = \nu(\Theta_2(1), \ldots, \Theta_2(n), t) \tag{3.5}$$

Chaque fonction est définie par un vecteur de paramètres scalaires Θ, et c'est sur les composantes de celui-ci que porte l'optimisation. Les méthodes d'optimisation paramétrique sont quant à elles plus répandues, et différents algorithmes de résolution numérique sont disponibles, comme nous le verrons plus en détail.

3.1.1 Transformée de Fourier

La première étape consiste à rechercher une paramétrisation satisfaisante pour les fonctions λ et ν. On peut songer naturellement à prendre en premier lieu les amplitudes respectives λ_m et ν_m ainsi que le déphasage relatif Φ, tels que définis en (2.64) et (2.65).

[2] La définition exacte de ce critère sera précisée par la suite. Notons par ailleurs que le fait de choisir un critère à minimiser n'est absolument pas restrictif pour la suite des calculs, étant donné que la minimisation de J est équivalente à la maximisation de $-J$.

Toutefois, ce choix impose l'allure des signaux $\frac{\lambda}{\lambda_m}(t)$ et $\frac{\nu}{\nu_m}(t)$, et restreint donc l'espace solution. C'est pourquoi on cherchera non pas à optimiser ces paramètres usuels mais plutôt la forme globale des signaux.

Considérons donc une fonction $f \in \mathcal{C}_T^\infty$. Les coefficients de Fourier de f sont définis par [Papoulis 1962] :

$$a_0 = \frac{1}{T}\int_{-T/2}^{T/2} f(u)du \tag{3.6a}$$

$$a_p = \frac{2}{T}\int_{-T/2}^{T/2} f(u)\cos\frac{2\pi pu}{T}du \quad \forall p \in \mathbb{N}^* \tag{3.6b}$$

$$b_p = \frac{2}{T}\int_{-T/2}^{T/2} f(u)\sin\frac{2\pi pu}{T}du \quad \forall p \in \mathbb{N}^* \tag{3.6c}$$

et la série de Fourier de f est égale à la fonction de $\mathcal{C}_T^\infty$:

$$S_f(t) = a_0 + \sum_{p=1}^{\infty} a_p \cos\frac{2\pi pt}{T} + b_p \sin\frac{2\pi pt}{T} \tag{3.7}$$

On montre alors que la série de Fourier de f converge uniformément vers f sur $\mathbb{R}$:

$$\lim_{p\to\infty} ||S_f - f|| = 0 \tag{3.8}$$

En d'autres termes, en définissant la somme partielle :

$$S_f^n(t) = a_0 + \sum_{p=1}^{n} a_p \cos\frac{2\pi pt}{T} + b_p \sin\frac{2\pi pt}{T} \quad \forall p \in \mathbb{N}^* \tag{3.9}$$

on peut considérer que S_f^n représente une bonne approximation de f sur $\mathbb{R}$ pour un ordre n suffisamment grand. La décomposition en série de Fourier constitue donc un moyen rapide d'approcher une fonction périodique — tels les angles de battement et de rotation $\lambda(t)$ et $\nu(t)$ — à l'aide d'une représentation paramétrique de la forme (a_n, b_n), n fini. Le principal inconvénient de cette méthode réside dans les oscillations rencontrées au voisinage de « pseudo-discontinuités » (par exemple lors de la commutation de $-\nu_m$ à $+\nu_m$ pour la rotation[3]), effet connu sous le nom de phénomène de Gibbs [Labarrère *et al.* 1993]. Cette méthode a toutefois été appliquée par Th. Le Moing et B. Dang Vu pour la recherche de cinématiques optimales dans le cadre du PRF *REMANTA* [Luc-Bouhali *et al.* 2005].

3.1.2 Réseaux de neurones

L'approche développée durant ce travail de thèse est quelque peu différente, puisqu'elle se base sur des réseaux de neurones pour modéliser les cinématiques des ailes. Nous allons présenter rapidement cet outil et les méthodes associées, avant de revenir sur son application au problème qui nous intéresse.

[3] Les fonctions λ et ν sont des angles physiques, donc continus par nature, et les discontinuités ne seront qu'apparentes : le passage de $-\nu_m$ à $+\nu_m$ correspondra en réalité à un front montant de forte pente.

3.1.2.1 Présentation générale

Les premières recherches sur les réseaux de neurones artificiels datent des années soixante [Minsky et Papert 1969], et furent motivées par le constat que les organismes vivants les plus simples, tels les invertébrés, réalisent aisément des tâches que les ordinateurs de l'époque n'accomplissaient qu'au prix de lourds calculs, telles la reconnaissance d'objets indépendamment de leur taille ou de leur position. L'idée de s'inspirer du fonctionnement des systèmes nerveux pour concevoir des machines plus habiles que les ordinateurs conventionnels a donc émergé. Ces techniques ont connu un regain d'intérêt dans les années quatre-vingts [Hopfield et Tank 1985, Kawamoto et Anderson 1985], avec le développement d'outils plus puissants en informatique aussi bien qu'en mathématiques. Les réseaux de neurones constituent aujourd'hui un pan très actif de la recherche dans le domaine de l'informatique et de l'intelligence artificielle.

Un neurone artificiel peut être simplement défini comme une fonction non linéaire, paramétrée et à valeurs bornées [Dreyfus *et al.* 2004] :

$$y = f(x_1, \ldots, x_n; w_1, \ldots, w_p) \tag{3.10}$$

Les variables $\{x_i\}_{i=1\ldots n}$ sur lesquelles opère le neurone sont habituellement désignées sous le terme d'entrées du neurone, et la valeur y de la fonction sous celui de sortie. Deux types de paramétrages sont fréquemment utilisés pour f :

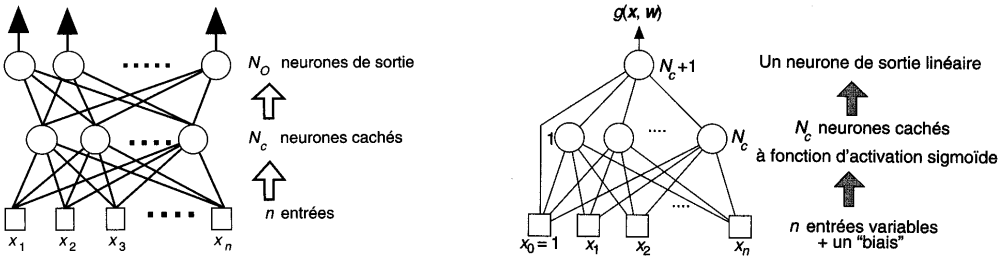
- les paramètres sont attachés aux entrées du neurone : la sortie de celui-ci est une fonction non linéaire d'une combinaison des entrées $\{x_i\}$ pondérées par les paramètres $\{w_i\}$, qui sont alors désignés sous le terme de poids. Cette combinaison linéaire est quant à elle appelée potentiel, en raison de l'inspiration biologique sous-jacente. La fonction f est appelée fonction d'activation, et est souvent prise égale à une fonction arctangente ou sigmoïde :
$$f(u) = \frac{1}{1 + e^{-u}} \tag{3.11}$$
- les paramètres sont attachés à la non-linéarité du neurone : ils interviennent directement dans la fonction d'activation f. Celle-ci peut être une ondelette, ou bien encore une fonction à base radiale (RBF, *Radial Basis Function*), par exemple de la forme :
$$y = \exp\left[-\frac{\sum_{i=1}^{n}(x_i - w_i)^2}{2w_{n+1}^2}\right] \tag{3.12}$$

La différence pratique essentielle entre ces deux types de neurones est que les derniers ont des non-linéarités locales, qui tendent vers zéro dans toutes les directions de l'espace des entrées : leur zone d'influence est donc limitée, ce qui n'est pas le cas des neurones à potentiel et fonction d'activation sigmoïde.

L'intérêt des neurones réside dans les propriétés qui résultent de leur association en réseau, c'est-à-dire de la composition des fonctions non linéaires réalisées par chacun des neurones. Là encore on distingue deux types de réseaux : les réseaux non bouclés et les réseaux bouclés. Un réseau non bouclé est représenté graphiquement par un ensemble de neurones connectés entre eux, l'information circulant des entrées vers les sorties, comme illustré sur la figure 3.1(a). Le graphe d'un tel réseau est acyclique, c'est-à-dire qu'on ne peut pas revenir à son point de départ en suivant le trajet des connexions depuis un neurone quelconque. Les neurones qui effectuent le dernier calcul de la composition de fonctions sont appelés neurones de sortie, ceux qui effectuent les calculs intermédiaires sont les neurones cachés (cf. fig. 3.1(a)). Les réseaux non bouclés à couches dont les neurones cachés ont une

fonction d'activation sigmoïde sont souvent appelés perceptrons multicouches, ou MLP (*Multi-Layer Perceptron*). Un cas particulier est celui des réseaux à une couche cachée et un seul neurone de sortie linéaire : un tel réseau réalise ainsi une fonction non linéaire des paramètres de la première couche de connexions (cf. fig. 3.1(b)).

Les réseaux bouclés, quant à eux, présentent une structure cyclique, ce qui implique la prise en compte du temps afin que l'entrée d'un neurone ne dépende pas de sa sortie au même instant. Ainsi, à chaque connexion d'un tel réseau est attaché, outre un poids comme pour les réseaux non bouclés, un retard, sous forme de multiple entier de l'unité de temps choisie : le réseau ainsi défini réalise une ou plusieurs équations aux différences non linéaires. Toutefois, on peut montrer que tout réseau de neurones bouclé peut être mis sous une forme canonique comprenant un réseau non bouclé dont certaines sorties sont ramenées aux entrées par des bouclages de retard unité.



(a) Un réseau à n entrées, une couche de N_c neurones cachés et N_O neurones de sortie

(b) Un réseau à $n+1$ entrées, une couche de N_c neurones cachés et un neurone de sortie linéaire

FIG. 3.1 – Schéma de réseaux de neurones non bouclés

Une notion fondamentale dans l'étude et l'utilisation de réseaux de neurones est celle d'apprentissage. On désigne sous ce terme, toujours en référence à la biologie, la procédure qui consiste à estimer les paramètres des différents neurones du réseau afin que celui-ci remplisse au mieux la tâche qui lui est affectée. Deux types d'apprentissage sont à distinguer : l'apprentissage supervisé et l'apprentissage non supervisé. Dans le premier cas, on suppose que l'on connaît les valeurs que doit avoir la sortie du réseau pour des valeurs données des entrées : il s'agit alors pour un « professeur » (en l'occurrence un algorithme d'optimisation) d'apprendre au réseau quelles sont les bonnes valeurs, sous-entendu celles à reproduire. L'apprentissage non supervisé est quant à lui utilisé principalement pour des tâches d'agrégation, c'est-à-dire de regroupement de données, représentées par des vecteurs de grande dimension, dans des espaces de dimension beaucoup plus petite, selon des critères de ressemblance inconnus *a priori*. Il n'y a donc pas là de professeur, puisque c'est au réseau de découvrir par lui-même les similitudes entre les données. Les réseaux à apprentissage non supervisé les plus utilisés dans ce but sont les cartes auto-organisatrices de Kohonen.

Pour ce qui est des domaines d'application des réseaux de neurones, ceux-ci sont également très variés. On rencontre ainsi, comme on l'a vu, la classification de données, suivant des critères qui peuvent être connus à l'avance (reconnaissances de formes, *data mining*), ou non. Ils peuvent également être utilisés pour la modélisation de systèmes non linéaires : dans le cas où tout ou partie de ceux-ci peut être difficilement décrit par des équations physiques (modèle de connaissance), ou bien en présence d'incertitudes importantes sur les paramètres du modèle, un réseau de neurones pourra fournir une bonne modélisation du comportement du système, pourvu que l'apprentissage ait été effectué sur un domaine de

fonctionnement suffisamment grand. L'autre intérêt d'utiliser des réseaux de neurones pour la modélisation des processus est que ceux-ci pourront ensuite être commandés en boucle fermée toujours grâce à des réseaux de neurones, soit par inversion directe du modèle, soit par utilisation d'un modèle de référence comme en commande adaptative.

Toutefois, l'une des utilisations les plus efficaces des réseaux de neurones — qui nous préoccupe directement dans le cadre de cette thèse — est celle de la modélisation statique et de l'approximation de fonction. On montre en effet que toute fonction bornée suffisamment régulière peut être approchée uniformément, avec une précision arbitraire, dans un domaine fini de l'espace de ses variables, par un réseau de neurones comportant une couche cachée de neurones possédant tous la même fonction d'activation, et un neurone de sortie linéaire [Hornik *et al.* 1989]. Il sera donc possible de représenter chaque fonction $\lambda(t)$ et $\nu(t)$ à l'aide d'un réseau ayant une telle structure, dont les paramètres auront été ajustés par un apprentissage de la fonction à reproduire. Mais l'avantage de cette méthode par rapport à d'autres méthodes d'approximation est que les réseaux de neurones sont des approximateurs parcimonieux [Dreyfus *et al.* 2004] : en d'autres termes ils nécessitent, pour une précision d'approximation donnée, moins de paramètres qu'une autre méthode (en particulier les méthodes linéaires).

L'intérêt de cette propriété des réseaux de neurones est loin d'être négligeable : dans le cas général, un nombre plus réduit de paramètres allégera les temps de calcul ; dans le cadre d'un problème d'optimisation comme c'est le cas ici, cela entraînera de meilleures performances des algorithmes en termes de rapidité et de convergence, et augmentera par conséquent les chances de trouver la solution optimale. En ce qui concerne le choix du type de neurones, la sortie des réseaux de RBF à centres et écarts-types fixés est linéaire par rapport aux poids, tandis que celle des réseaux de neurones à fonction d'activation sigmoïde est non linéaire. Or il a été montré que si une approximation dépend des paramètres ajustables de manière non linéaire, elle est plus parcimonieuse que si elle dépend linéairement de ceux-ci [Barron 1993]. On peut donc en déduire qu'un réseau de neurones cachés à fonction d'activation sigmoïde est un meilleur approximateur qu'un réseau à RBF.

Toutes ces considérations permettent de conclure qu'un réseau de type perceptron multicouches est un bon choix pour la modélisation des cinématiques d'entrée, étant donné qu'il garantit un compromis entre le nombre de paramètres requis et la variété des allures représentées.

3.1.2.2 Application à la modélisation de signaux

La démarche va donc consister à modéliser les fonctions $\lambda(t)$ et $\nu(t)$ par des réseaux de neurones à une entrée (t), dont les paramètres (poids des connexions et biais des neurones) seront ensuite optimisés vis-à-vis d'un critère donné. Dans le cas du vol stationnaire, on peut supposer que l'équilibre est atteint et donc supprimer les degrés de liberté en translation et rotation : toutes les grandeurs sont alors périodiques, ce qui nous permet de réduire l'étude du problème à une seule période $t \in [0; T]$. Le critère à optimiser peut donc être défini comme la valeur moyenne de l'effort vertical R_z :

$$\bar{R}_z = \frac{1}{T} \int_0^T R_z dt \tag{3.13}$$

L'axe $\vec{z}$ étant dirigé vers le bas, la portance moyenne sur un cycle est égale à $-\bar{R}_z$ (cf. note 6 p. 88). En posant dans un premier temps $J = \bar{R}_z$, maximiser la portance moyenne revient donc à minimiser le critère J.

Il s'agit ensuite de déterminer une structure minimale pour chacun des réseaux. Pour ce faire, on considère un réseau de base comportant une couche cachée et une couche de sortie, cette dernière se résumant à un neurone linéaire (qui agit en tant que simple sommateur), et on cherche dans chacun des cas le nombre minimal de neurones permettant de représenter la plus grande variété d'allures de fonctions. On soumet pour cela un certain nombre de fonctions « types » (c'est-à-dire que l'on souhaitera voir éventuellement reproduire par la suite) au réseau, et on vérifie si celui-ci est capable de les apprendre correctement, ou en d'autres termes s'il existe une réalisation du vecteur des paramètres telle que la sortie du réseau constitue une bonne approximation de la fonction modèle. La création du réseau et son apprentissage ont été réalisées sous MATLAB, à l'aide de la boîte à outils dédiée *Neural Network*. Compte-tenu de sa rapidité et son efficacité, l'algorithme choisi pour l'apprentissage fut celui de Levenberg-Marquardt [Bonnans *et al.* 2003, Dreyfus *et al.* 2004], qui consiste à calculer itérativement le vecteur des poids Θ par la formule :

$$\Theta(i) = \Theta(i-1) - [H(\Theta(i-1)) + \mu_i I]^{-1} \nabla C(\Theta(i-1)) \tag{3.14}$$

où ∇ représente l'opérateur gradient. Il s'agit en fait d'un algorithme de minimisation d'une fonction de coût quadratique C associée à l'erreur e entre la sortie obtenue et la sortie désirée. L'expression de la matrice hessienne de cette fonction de coût est :

$$H(\Theta(i)) = \sum_{k=1}^{N} \left(\frac{\partial e^k}{\partial \Theta(i)} \right)^t \left(\frac{\partial e^k}{\partial \Theta(i)} \right) + \sum_{k=1}^{N} \frac{\partial^2 e^k}{\partial \Theta(i)^t \partial \Theta(i)} e^k \tag{3.15a}$$

$$\approx \sum_{k=1}^{N} \left(\frac{\partial e^k}{\partial \Theta(i)} \right)^t \left(\frac{\partial e^k}{\partial \Theta(i)} \right) \tag{3.15b}$$

en négligeant le deuxième terme, étant donné que celui-ci est proportionnel à l'erreur e^k au pas de temps k.

Considérons en premier lieu la modélisation de l'angle de rotation de l'aile $\nu(t)$. Comme on l'a vu, l'observation du vol stationnaire animal, chez les insectes et le colibri, montre que cet angle est à peu près constant sur une demi-période : une famille de solutions possible sera donc la base de fonctions carrées ν_{car} définies en (2.64) p. 83. On ne souhaite toutefois pas écarter les fonctions plus classiques de type trigonométrique par exemple : le réseau choisi devra donc être capable de reproduire par exemple les fonctions carré $\nu_{\text{car}}(t) = \nu_m \tanh(k_{\text{car}} \cos \omega t)$ et cosinus $\nu_{\cos}(t) = \nu_m \cos \omega t$ (le paramètre k_{car} définissant la raideur des fronts montants et descendants).

Après plusieurs essais, il s'avère que le réseau minimal capable de reproduire ces formes de signaux comporte deux neurones sur sa couche cachée (voir schéma 3.2). Les poids $\{w_i\}_{i=1\ldots4}$ sur chaque connexion plus les biais $\{b_j\}_{j=1\ldots3}$ que l'on peut affecter à chaque fonction de transfert donnent un total de $4+3=7$ paramètres à optimiser. Étant donné la simplicité de ce réseau, la fonction de transfert globale de celui-ci peut s'écrire directement de manière analytique :

$$f(t) = \sum_{i=1}^{2} \frac{w_{i+2}}{1 + e^{-(w_i t + b_i)}} + b_s \tag{3.16}$$

L'entrée du réseau est naturellement le vecteur temps $t \in [0; T]$, b_1, b_2 sont les biais des neurones de la couche cachée, b_s est le biais de la couche de sortie.

On a représenté sur les figures 3.3(a) à 3.3(e) le résultat de l'apprentissage des fonctions ν_{car} et $\nu_{\cos}$ sus-citées, ainsi que d'autres allures de signaux : ν_{rot1} et ν_{rot2}, fonctions

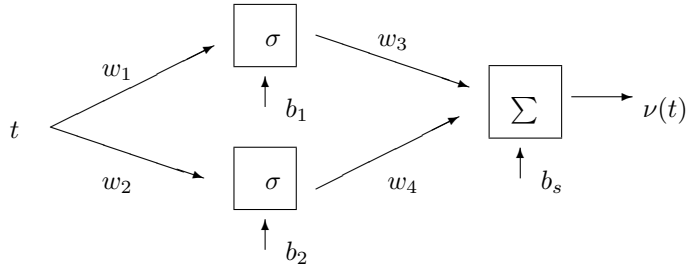


FIG. 3.2 – Structure du réseau choisi pour modéliser $\nu(t)$

trapézoïdales, où la jonction entre les paliers se fait à l'aide d'arcs de sinusoïdes (allures inspirées des cinématiques du dispositif *Robofly*), et enfin ν_{comp}, composition de fonctions carrée et cosinus :

$$\nu_{\text{comp}}(t) = \begin{cases} \nu_{\text{car}}(t), \ t \in [0; T/2[\\ \nu_{\cos}(t), \ t \in [T/2; T[\end{cases} \tag{3.17}$$

On constate que cette structure de réseau est capable de reproduire parfaitement les différentes fonctions proposées, ce qui suggère donc que l'espace des solutions atteignables sera suffisamment vaste.

Une démarche exactement similaire a été employée pour l'angle de battement de l'aile $\lambda(t)$, dont l'allure est susceptible d'évoluer entre la sinusoïde pure $\lambda(t) = \lambda_m \sin \omega t$ et le signal trianglaire $\lambda_{\text{tri}}(t) = \lambda_m \int_0^t \tanh(k_{\text{tri}} \cos \omega u) du$. Une couche cachée comportant cette fois-ci 4 neurones s'est avérée nécessaire (étant donné par la plus grande complexité des fonctions à apprendre), ce qui correspond à $4 \times 3 + 1 = 13$ paramètres.

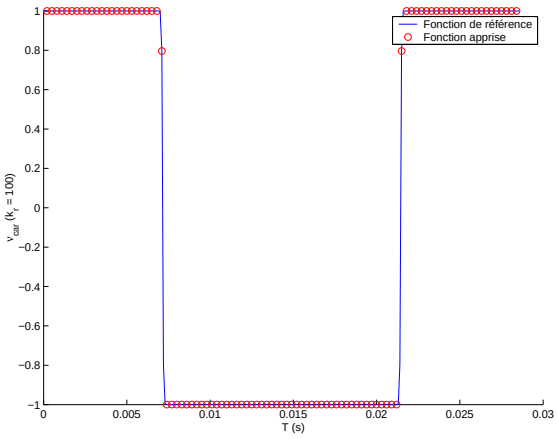
3.2 Optimisation des paramètres du réseau

Une fois la structure minimale des réseaux de neurones déterminée, la recherche des formes de cinématiques maximisant le critère proposé se fait en optimisant le vecteur des paramètres (poids et biais) des réseaux modélisant respectivement λ et ν. Différentes méthodes de résolution ont été employées : on a choisi en premier lieu une méthode numérique, basée sur une estimation des dérivées successives (gradient et hessien) tout au long de la progression vers l'optimum. Toutefois, nous verrons que cette méthode ne s'est pas avérée concluante, du fait de la présence de plusieurs optimums locaux. C'est pourquoi nous nous sommes tournés par la suite vers des méthodes dites métaheuristiques, qui présentent l'avantage de converger en théorie vers l'optimum absolu (ou *optimum optimorum*) du critère.

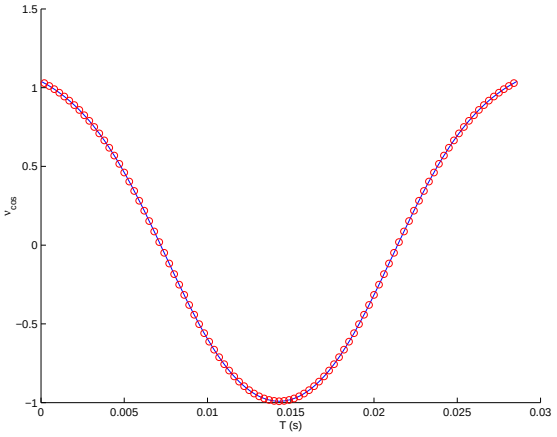
3.2.1 Méthode numérique

3.2.1.1 Principe et algorithme

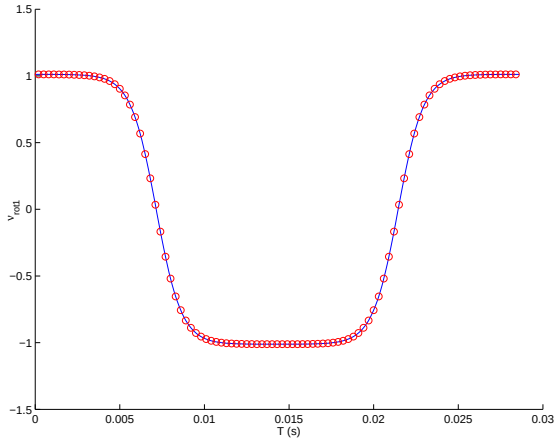
Soit $\Theta = (w_1 \ w_2 \ w_3 \ w_4 \ b_1 \ b_2 \ b_s)$ le vecteur contenant les paramètres à optimiser et f_Θ le transfert du réseau de neurones associé. La recherche d'une réalisation optimale Θ^* de Θ a été effectuée dans un premier temps à l'aide de la fonction d'optimisation sous contrainte `fmincon` disponible sous MATLAB. Pour ce type de problème, cette fonction utilise un algorithme de programmation quadratique itérative ou SQP (*Sequential Quadratic Programming*), que l'on décrit brièvement ci-après.



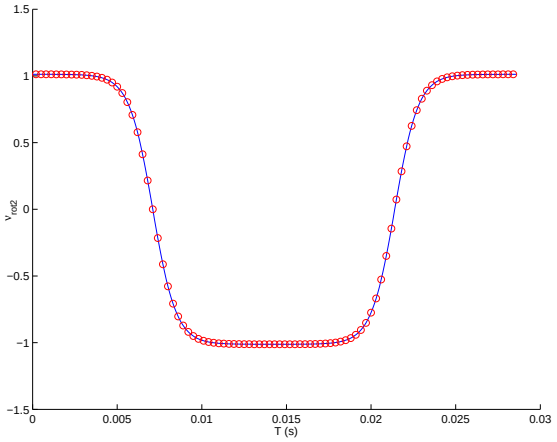
(a) ν_{car}



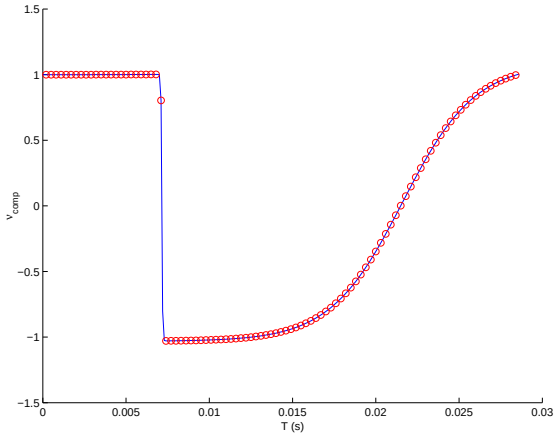
(b) ν_{cos}



(c) ν_{rot1}



(d) ν_{rot2}



(e) ν_{comp}

FIG. 3.3 – Apprentissage de différentes fonctions de base

Le principe est le suivant : soit le problème général de la minimisation sous contraintes d'une fonction J :

$$\begin{cases} \min\limits_{x\in\Omega} J(x) \\ c_E(x) = 0 \\ c_I(x) \leq 0 \end{cases} \tag{3.18}$$

Ω étant un ouvert de $\mathbb{R}^n$, c_E et c_I représentant respectivement les équations des contraintes d'égalité et d'inégalité. Pour une écriture plus concise, on peut supposer que ces contraintes peuvent être regroupées en une seule contrainte c, telle que $c(x) = 0$ si et seulement si $c_E(x) = 0$ et $c_I(x) \leq 0$. On peut ensuite définir le lagrangien associé à ce problème :

$$L(x, \ell) = J(x) + {}^t\ell c(x) \tag{3.19}$$

où ℓ est le vecteur des multiplicateurs de Lagrange. La solution de (3.18) est alors approchée de manière itérative [Bonnans *et al.* 2003] :

$$x_{k+1} = x_k + \alpha_k d_k \tag{3.20}$$

La direction de descente d_k est calculée à chaque itération par résolution d'un sous-problème quadratique (d'où le nom de la méthode) :

$$\begin{cases} \min\limits_{d\in\mathbb{R}^n} {}^t\nabla J(x_k)d + \frac{1}{2}{}^t d H_k d \\ {}^t\nabla c_E(x_k)d + c_E(x_k) = 0 \\ {}^t\nabla c_I(x_k)d + c_I(x_k) = 0 \end{cases} \tag{3.21}$$

dans lequel H_k est une approximation de la matrice hessienne du lagrangien, calculé par exemple à l'aide de la méthode de BFGS (Broyden, Fletcher, Goldfarb, Shanno) :

$$H_{k+1} = H_k + \frac{q_k {}^t q_k}{{}^t q_k s_k} - \frac{{}^t H_k H_k}{{}^t s_k H_k s_k} \tag{3.22}$$

$$\text{avec} \quad s_k = x_{k+1} - x_k \tag{3.23}$$

$$q_k = \nabla J(x_{k+1}) - \nabla J(x_k) + \sum_{i=1}^{n} \ell_i \left[\nabla c_i(x_{k+1}) - \nabla c_i(x_k)\right] \tag{3.24}$$

Le pas de descente α_k est quant à lui déterminé de manière à obtenir une diminution suffisante d'une fonction de mérite donnée [Powell 1978].

3.2.1.2 Résultats

L'optimisation a concerné dans un premier temps l'angle de rotation seulement, le battement ayant une forme fixe. La périodicité des entrées permet de restreindre le problème à une seule période $[0;T]$, et l'algorithme d'optimisation aura à prendre en compte la contrainte exprimant la continuité des signaux en fin de période, c'est-à-dire :

$$f_\Theta(0) - f_\Theta(T) = 0 \tag{3.25}$$

Dans un premier temps le réseau est initialisé de manière arbitraire, et la fonction d'optimisation n'atteint pas son but (non-convergence de l'algorithme, ou contraintes non respectées). On choisit alors d'initialiser les poids du réseau de façon à reproduire une fonction connue, par exemple la fonction carrée représentée sur la figure 3.3(a) page 103. Cette fois-ci l'optimisation converge et une solution est trouvée. Mais la répétition du

	Θ^0 vecteur des poids d'initialisation du réseau						
k_{car}	w_1^0	w_2^0	w_3^0	w_4^0	b_1^0	b_2^0	b_s^0
1	541,47	541,79	1,6352	-1,6347	-11,575	-3,8988	0,80696
3	-1312,3	1312,2	-1,9847	-1,9848	28,12	-9,373	2,9771
5	-2178	-2178	-1,9979	1,9979	46,672	15,557	0,99894
10	-4366,5	-4367	-1,9996	1,9996	93,568	31,193	0,99982
20	-8727,6	8727,5	-1,9998	-1,9998	187,02	-62,34	2,9998
30	-13169	13169	-2	-2	282,2	-94,067	3
40	17416	-17422	1,9999	1,9999	-373,2	124,44	-0,99995
50	21377	21431	1,9998	-1,9997	-458,09	-153,08	0,99986
75	-32637	32829	-2	-2	699,37	-234,49	3
100	41846	-43016	1,9999	1,9999	-896,75	307,28	-0,99997

	Θ^* vecteur des poids optimisés								
k_{car}	w_1^*	w_2^*	w_3^*	w_4^*	b_1^*	b_2^*	b_s^*	J^*	$\lvert\Theta^* - \Theta^0\rvert$
1	541,5	541,8	2,0013	-2,0892	-10,611	-2,9557	1,0534	-0,23732	1,4904
3	-1312,3	1312,2	-1,7355	-1,7365	26,228	-7,4804	2,6039	-0,26736	2,7252
5	-2178,1	-2178	-1,6881	1,6881	43,87	12,767	0,84375	-0,2768	3,9826
10	-4366,6	-4367	-1,6501	1,6501	88,373	26,018	0,82482	-0,27981	7,3522
20	-8727,8	8727,6	-1,6327	-1,6327	176,51	-51,853	2,4488	-0,27906	14,868
30	-13170	13170	-1,6285	-1,6285	266,14	-78,127	2,4425	-0,2786	22,644
40	17416	-17422	1,6266	1,6266	-351,87	103,44	-0,81375	-0,27839	29,947
50	21378	21431	1,6272	-1,6272	-431,01	-126,83	0,81294	-0,27825	37,724
75	-32638	32829	-1,6265	-1,6265	657,34	-196,97	2,439	-0,27812	56,351
100	41848	-43017	1,6554	1,6554	-804,56	235,27	-0,83126	-0,26742	117,01

TAB. 3.1 – Influence de l'initialisation du réseau sur l'optimisation

processus pour différentes fonctions d'initialisation, en l'occurrence la même fonction carrée, mais avec différentes valeurs du paramètre de forme k_{car}, conduit à des résultats différents, qu'il s'agisse du vecteur optimal Θ^* ou du critère J^*. On présente dans le tableau 3.1 les résultats d'optimisations pour diverses valeurs du paramètre k_{car} de la fonction ayant servi à initialiser le réseau.

Le premier tableau détaille les composantes du vecteur d'initialisation Θ^0. Ces poids et biais ont été obtenus *via* l'apprentissage de fonctions carrées de paramètre de forme k_{car}, et constituent le point de départ de l'optimisation. Le second tableau montre respectivement les composantes du vecteur optimal Θ^* obtenu à l'issue de l'optimisation pour chaque paramètre caractérisant la fonction initialement apprise, le critère J^* à l'optimum ainsi que la « distance » entre les points de départ et d'arrivée, exprimée à l'aide de la norme $|\Theta^* - \Theta^0|$. On constate donc que, si la procédure d'optimisation converge à chaque fois, l'allure de la rotation optimale f_{Θ^*} tout comme la valeur du critère J^* sont différents, selon le paramètre k_{car} caractérisant la forme de la fonction d'initialisation du réseau. En d'autres termes, l'optimum trouvé dépend du point d'initialisation. On peut donc supposer que tous les optimums trouvés sont en fait des optimums locaux, qui n'ont pu être franchis par la fonction d'optimisation. Il apparaît que certains poids varient moins que d'autres durant l'optimisation, tels w_1 et w_2. Mais d'un autre côté les valeurs de ces mêmes poids sont très différentes suivant l'initialisation : la modélisation d'un signal par réseau de neurones est liée en effet à un processus d'apprentissage qui reste par nature opaque pour l'utilisateur ; un même signal pourra être reproduit de la même façon avec des jeux de poids complètement différents, c'est ce qui rend très délicate l'interprétation de ces poids comme des paramètres « classiques » caractérisant une fonction.

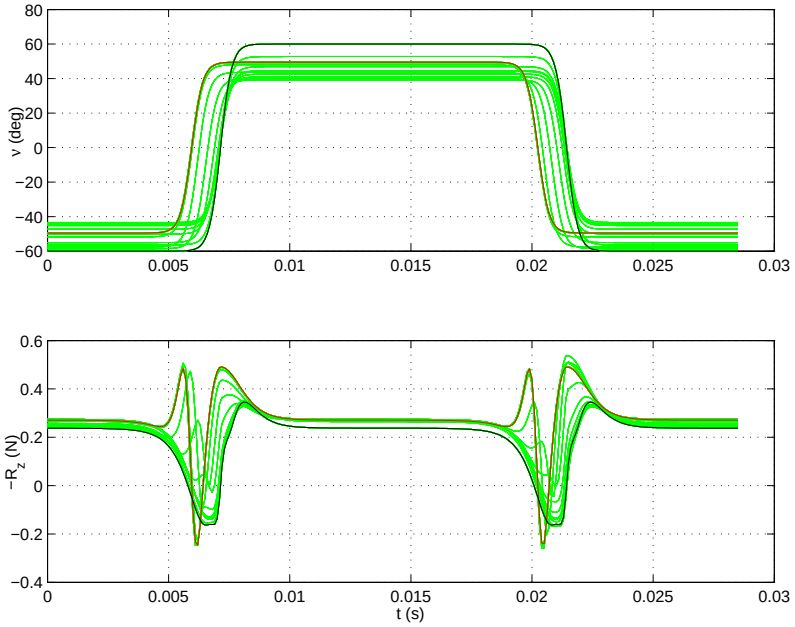


FIG. 3.4 – Évolution de l'entrée ν et de la portance

Les résultats obtenus sont néanmoins significatifs : on a reproduit sur la figure 3.4 p. 106 l'évolution de l'entrée ν et de la portance correspondante au fur et à mesure des itérations de l'algorithme d'optimisation. En noir est représentée la courbe initiale, soit une fonction carrée avec un paramètre de forme $k_{\text{car}} = 10$, en vert les courbes correspondant aux itérations successives, et en rouge le résultat final de l'optimisation. On note alors que,

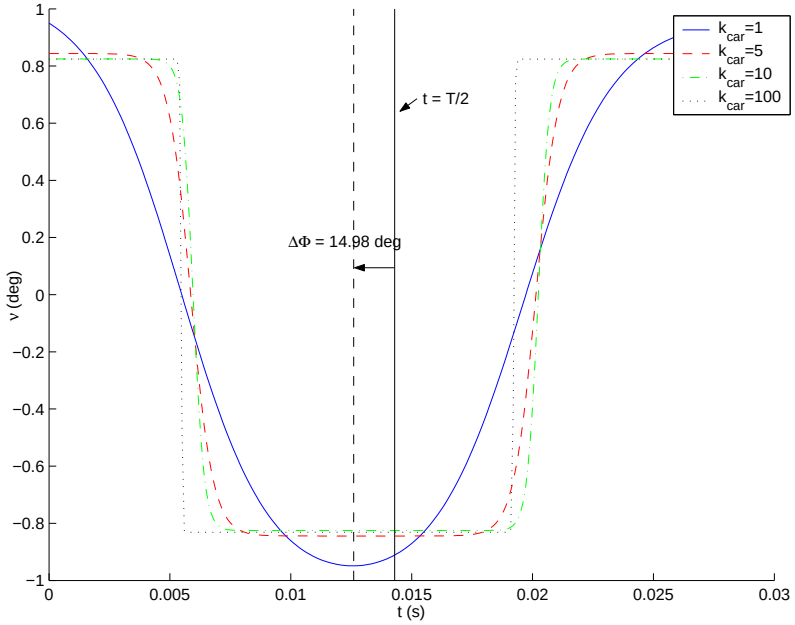


FIG. 3.5 – Rotations optimales obtenues par SQP, en fonction du paramètre de forme de la fonction d'initialisation

si l'allure en elle-même de la fonction n'est pas modifiée, l'optimisation des poids du réseau de neurones a tout de même permis de retrouver des résultats intuitifs ou bien issus d'optimisations plus empiriques. En effet, l'amplitude de la fonction optimale est plus faible que celle de la fonction initiale ($\pm 49{,}5\,^\circ$ contre $\pm 60\,^\circ$), ce qui correspond, du fait du plan de battement incliné à l'horizontale, à une incidence aérodynamique (et donc une portance) plus importante. En outre, cet optimum est en avance de phase par rapport à la courbe initiale (déphasage $\Delta\Phi = +14{,}98\,^\circ$) sans que l'on ait pour autant joué explicitement sur un déphasage global, ce qui corrobore le fait qu'une rotation en avance de phase par rapport au battement engendre un supplément de portance, résultat apparaissant dans la littérature (montage *Robofly*) et retrouvé par le biais d'autres simulations en jouant sur le déphasage comme seul paramètre. On retrouve ce résultat sur la figure 3.5, qui représente les rotations optimales obtenues pour différentes valeurs du paramètre de forme k_{car} utilisé pour l'initialisation. Il est clair que, si toutes les fonctions présentent des allures différentes (et similaires dans chaque cas à la fonction d'initialisation), elles possèdent toutes un déphasage $\Delta\Phi$ identique. La modélisation de la rotation de l'aile sous forme de réseau de neurones a donc permis, malgré la sensibilité de l'optimisation à la fonction initialisant l'apprentissage, de retrouver des résultats connus par des méthodes expérimentales : le signal ainsi modélisé vient se placer « naturellement » en avance de phase durant l'optimisation, générant ainsi le supplément de portance que l'on connaît bien.

3.2.2 Étude du comportement au voisinage de l'optimum

Afin de corroborer l'hypothèse de l'existence d'optimums locaux, on peut visualiser l'allure du critère au voisinage des optimums trouvés. L'espace de recherche étant de dimension 7, on ne pourra donc visualiser l'influence des divers paramètres que 2 par 2, à l'aide de coupes de la surface $J(\Theta)$ par des sous-espaces plans.

La procédure choisie est la suivante : on attribue au paramètre de forme k_{car} de la fonction carré modélisant la rotation de l'aile 3 valeurs différentes : $k_{\text{car}} \in \{1; 10; 101\}$. On réalise alors un apprentissage de ces signaux par un réseau de neurones comme précédemment, et les poids du réseau ainsi obtenu constituent le vecteur d'initialisation de l'algorithme d'optimisation. Lorsque ce dernier a convergé et trouvé un optimum (*a priori* non global), on trace la surface $\mathcal{S}$ représentant le critère J en fonction des paramètres du réseau regroupés 2 par 2 selon leur position au sein du réseau, soit : (w_1, w_2) (poids d'entrée), (w_3, w_4) (poids de sortie) et (b_1, b_2) (biais). Le biais de sortie total b_s n'a pas été utilisé, car il correspond à une translation verticale pure du signal, et n'a donc pas d'effet sur sa forme. Les résultats sont présentés sur les figures 3.6 à 3.14.

Pour chaque figure, l'optimum calculé se trouve au centre, et l'intervalle choisi pour faire varier chaque paramètre Θ_i est défini arbitrairement de la façon suivante :

$$\Theta_i \in [\Theta_i^* - d_i; \Theta_i^* + d_i]$$

avec :

$$d_i = 10|\Theta_i^0 - \Theta_i^*|$$

où Θ^0 est le vecteur initial et Θ^* le vecteur optimisé. On constate sur ces figures que la sensibilité du critère n'est pas la même pour les différents paramètres : celui-ci se montre très peu sensible aux variations des poids w_1 et w_2, et ce pour les différentes valeurs du paramètre de forme k_{car}. À l'inverse, on note de nombreux vallonnements dans le plan $(w_3; w_4)$ (figs. 3.9, 3.10, 3.11), traduisant ainsi la présence de plusieurs optimums locaux.

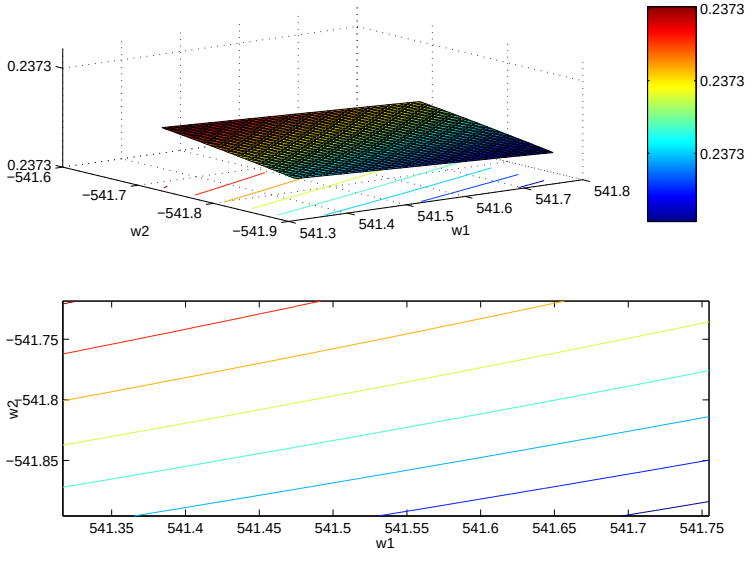


FIG. 3.6 – Plan $(w_1; w_2)$, $k_{\text{car}} = 1$

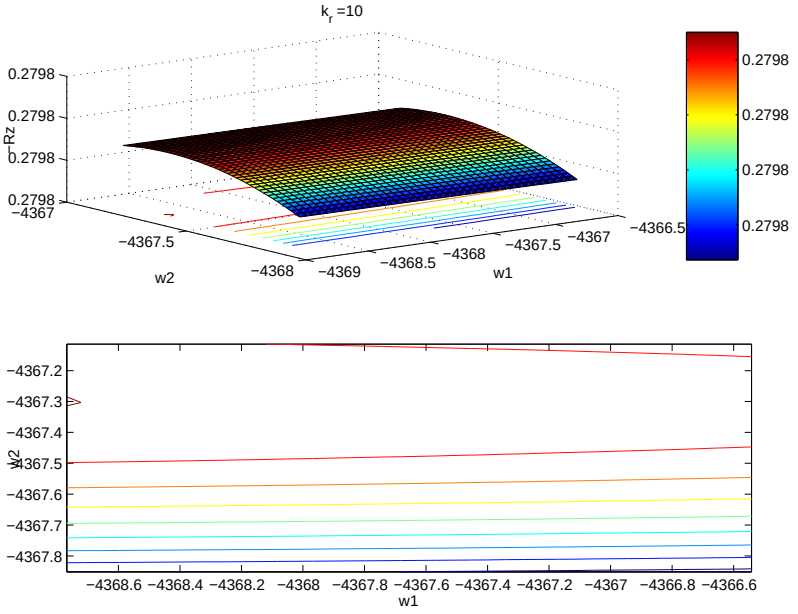


FIG. 3.7 – Plan $(w_1; w_2)$, $k_{\text{car}} = 10$

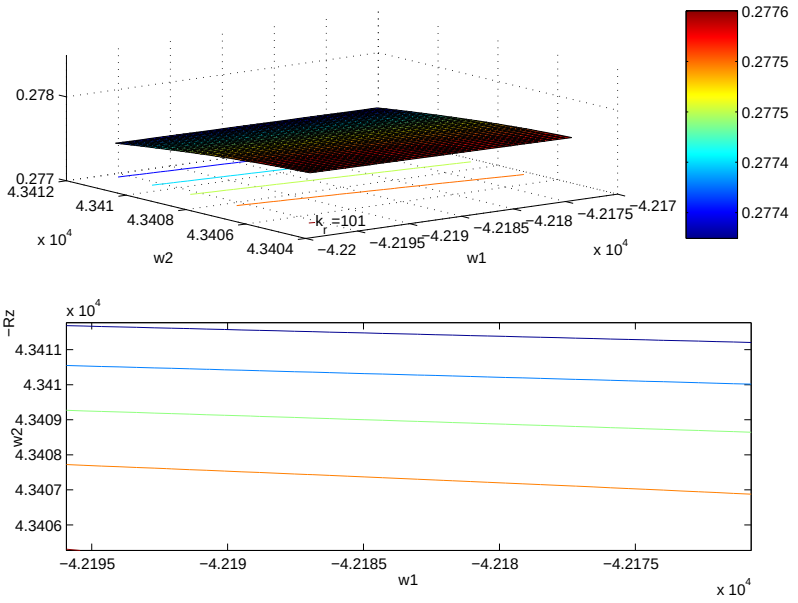


FIG. 3.8 – Plan $(w_1; w_2)$, $k_{\text{car}} = 101$

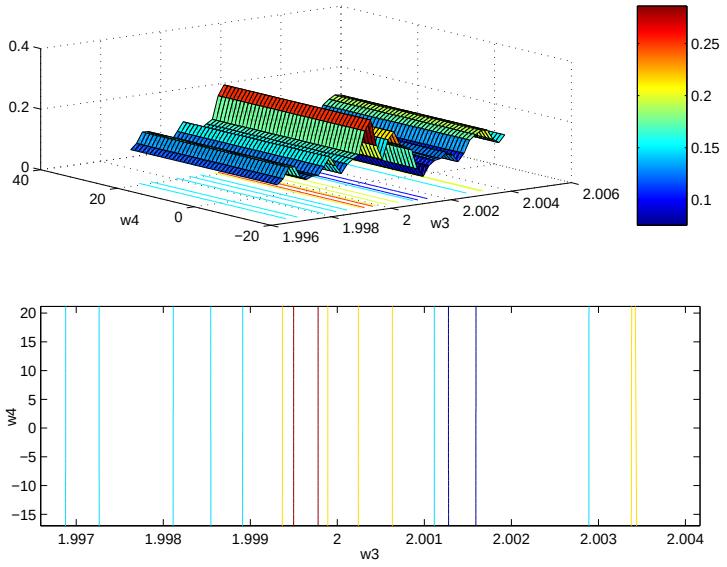


FIG. 3.9 – Plan $(w_3; w_4)$, $k_{\text{car}} = 1$

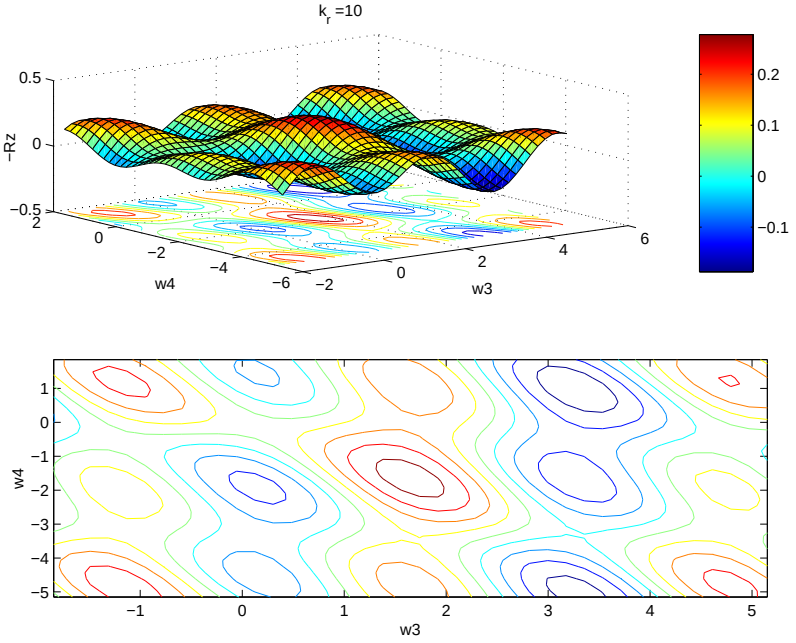


FIG. 3.10 – Plan $(w_3; w_4)$, $k_{\text{car}} = 10$

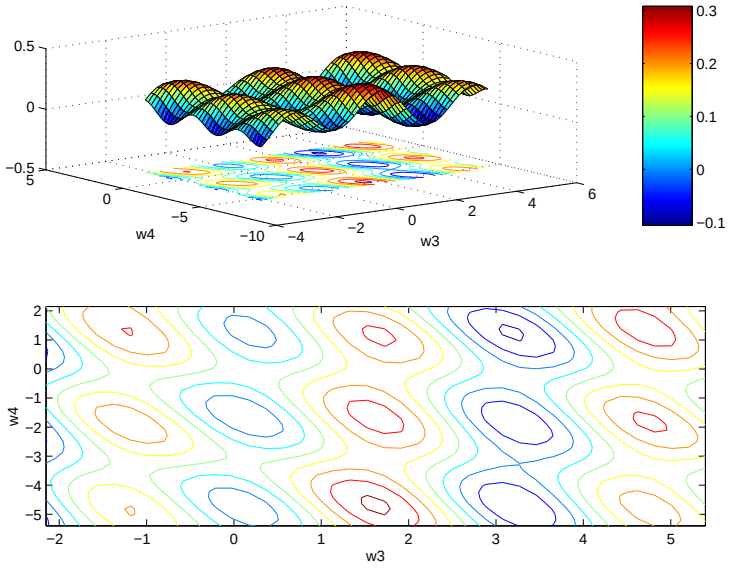


FIG. 3.11 – Plan $(w_3; w_4)$, $k_{\text{car}} = 101$

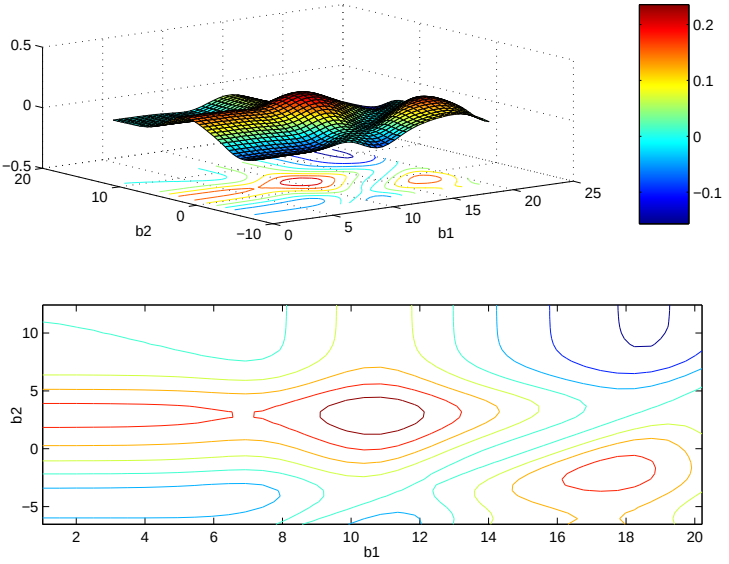


FIG. 3.12 – Plan $(b_1; b_2)$, $k_{\text{car}} = 1$

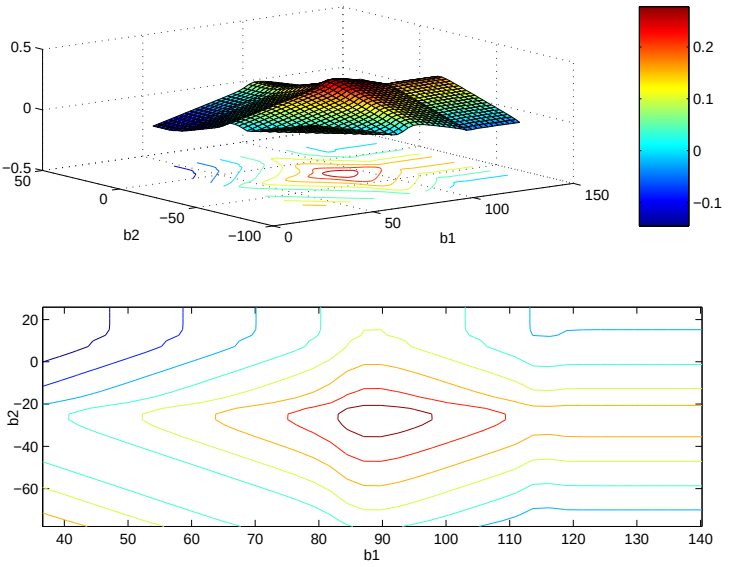


FIG. 3.13 – Plan $(b_1; b_2)$, $k_{\text{car}} = 10$

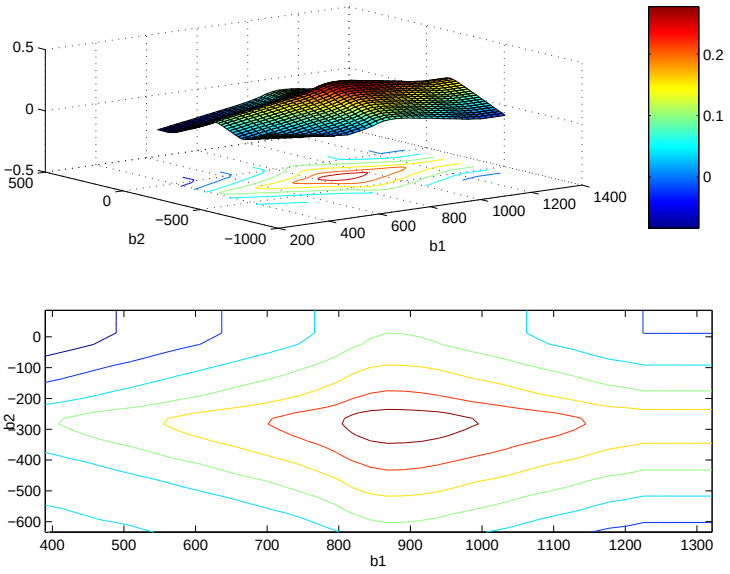


FIG. 3.14 – Plan $(b_1; b_2)$, $k_{\text{car}} = 101$

Ce relief dans les surfaces de critère $J(\Theta)$ confirme donc qu'une méthode d'optimisation numérique « classique » comme celles utilisées jusqu'alors risquerait de rester bloquée sur un optimum local, et donc de ne pas fournir l'*optimum optimorum* recherché.

Pour pallier cet inconvénient, il s'avérerait utile de considérer des algorithmes d'optimisation métaheuristiques, c'est-à-dire des méthodes qui ne s'appuient pas sur une connaissance *a priori* des caractéristiques du problème à résoudre (par exemple en nécessitant le calcul des dérivées du critère), mais plutôt sur une expérience acquise au cours du cheminement vers l'optimum.

3.2.3 Méthodes heuristiques

Les méthodes d'optimisation heuristiques s'avèrent être des outils particulièrement efficaces dans la résolution de problèmes d'optimisation combinatoire, tels ceux du voyageur de commerce ou bien du sac à dos, problèmes de classe NP pour lesquels l'exploration de tout l'espace des solutions possibles prendrait un temps considérable au-delà d'une certaine dimension [Johnson et Mc Geoch 1997]. Un autre avantage de ces méthodes est qu'elles ne nécessitent aucune condition sur la régularité du critère, ni même l'expression exacte de celui-ci [Renders 1995], ce qui se révèle très adapté au problème posé ici, pour lequel on a vu qu'une expression analytique du modèle (et donc du critère) est extrêmement délicate. Toutefois, elles présentent l'inconvénient d'un grand nombre de paramètres de convergence à régler, ce qui peut influer grandement sur les performances : il n'existe pour cela pas de règle générale, ni même en ce qui concerne le choix d'un algorithme plutôt qu'un autre pour un problème donné[4].

[4]De manière plus générale encore, on montre qu'il n'existe pas de méthode supérieure aux autres pour résoudre l'ensemble des problèmes existants (théorème *"No free lunch"* [Wolpert et Macready 1997]).

3.2.3.1 Description des algorithmes

Les méthodes employées ici sont au nombre de trois : deux méthodes dites de recherche aléatoire adaptative et un algorithme génétique, que nous allons rapidement présenter dans un premier temps. L'une des caractéristiques principales de celles-ci est l'introduction d'éléments stochastiques dans le processus de recherche : la « remontée » vers des valeurs apparemment moins intéressantes est ainsi autorisée de temps en temps, ce qui permet éventuellement le franchissement de cuvettes locales.

3.2.3.1.1 Recherche aléatoire adaptative

La première version de recherche aléatoire adaptative (que nous appellerons RAA1) est basée sur des travaux de Ye et Kalyanaraman [Ye et Kalyanaraman 2001], qui ont conçu un algorithme pour l'optimisation de paramètres d'un réseau de télécommunications. La recherche aléatoire adaptative est basée sur l'alternance de phases d'exploration (phase globale) et d'exploitation (phase locale). L'exploration consiste comme son nom l'indique à déterminer des régions encore inconnues, tandis que l'exploitation a pour but de converger vers un optimum local au sein d'une région sélectionnée. Le principe de l'algorithme de Ye et Kalyanaraman est d'explorer l'ensemble de l'espace des paramètres D, et d'exploiter seulement les régions définies par :

$$A_D(r) = \{\Theta \in D | J(\Theta) \leq y_r\} \tag{3.26}$$

pour un r donné, avec

$$\phi_D(y_r) = r \tag{3.27}$$

où $\phi_D(y_r)$ est la mesure définie par :

$$\phi_D(y_r) = \frac{m(\{\Theta \in D | J(\Theta) \leq y_r\})}{m(D)} \tag{3.28}$$

m étant la mesure de Lebesgue. La phase d'exploitation est quand à elle basée sur un tirage aléatoire des valeurs de Θ, associée à une réduction progressive de l'espace atteignable, de manière à converger vers un optimum. Cet algorithme a donné de bonnes performances sur les fonctions tests de Rastrigin et de Schwefel en dimension 2. Un autre algorithme de recherche aléatoire adaptative (RAA2) a été développé en parallèle par Th. Le Moing à l'ONERA, suivant une démarche qui alterne globalement des phases d'exploration aléatoire et des phases de recherche plus fine dans un voisinage des points sélectionnés.

3.2.3.1.2 Algorithme génétique

Les premières apparitions des stratégies dites évolutionnistes en tant qu'outils d'optimisation datent des années soixante, avec les travaux de Rechenberg [Rechenberg 1973] notamment. L'idée sous-jacente était de faire évoluer une population de solutions candidates à la résolution d'un problème donné, en utilisant des opérateurs inspirés par les altérations génétiques et la sélection naturelle. Dans le même temps, les algorithmes génétiques proprement dits furent inventés par Holland [Holland 1975]. Contrairement aux stratégies évolutionnistes, ceux-ci ne visaient pas au départ à la résolution de problèmes spécifiques, mais étaient plutôt envisagés comme un moyen d'étude formelle des mécanismes d'évolution et d'adaptation tels qu'ils apparaissent dans la nature. Ces deux approches se sont rapprochées et complétées, pour donner naissance au concept générique d'algorithmes génétiques que l'on rencontre aujourd'hui [Mitchell 1996].

Tout comme pour les méthodes de recherche aléatoire adaptative, il existe une infinie variété d'implémentations d'algorithmes génétiques. Celles-ci présentent néanmoins des bases communes : on considère une population initiale composée d'un certain nombre n d'*individus*. Chacun de ces individus est une réalisation du vecteur des paramètres (les *gènes*), en d'autres termes une solution candidate. À chaque itération ou *génération*, la population est en partie reconstituée grâce au croisement (ou *crossover*) de deux individus *parents*, résultant en un *enfant* dont les gènes sont hérités de ses parents. De plus, afin d'élargir l'espace de recherche, des *mutations* interviennent aléatoirement, correspondant à l'altération de tout ou partie des gènes d'un individu donné. Enfin, à chaque génération intervient le processus de *sélection naturelle*, qui ne conserve que les n meilleurs individus, c'est-à-dire ceux dont l'adéquation vis-à-vis du problème considéré est la plus forte : l'effectif initial est ainsi renouvelé, et la sélection naturelle garantit la convergence de la population vers des individus de plus en plus « aptes » à résoudre le problème posé.

Étant donnée la nature continue du problème ($\Theta \in \mathbb{R}^n$), différentes stratégies sont envisageables pour définir l'opération de croisement. En effet, dans le cas d'un problème binaire, c'est-à-dire si $\Theta \in \{0;1\}^n$, les gènes transmis à un enfant peuvent être déterminés simplement à l'aide de règles d'échange entre les gènes des deux parents. Mais si l'espace de recherche est continu, d'autres possibilités sont envisageables : on peut encore procéder par échanges stricts de portions du génotype, mais on peut également imaginer des règles plus complexes, basées par exemple sur une combinaison convexe des gènes :

$$\Theta'(i) = \Theta^1(i) + r.[\Theta^2(i) - \Theta^1(i)] \tag{3.29}$$

où Θ' désigne le génotype de l'enfant, Θ^1 et Θ^2 ceux de ses parents respectifs et r un coefficient aléatoire compris entre 0 et 1. C'est justement cette dernière méthode, dont les performances pour l'optimisation dans le domaine continu ont été prouvées [Ali et Törn 2004], qui a été employée au sein de l'algorithme génétique utilisé ici. Ce dernier est dérivé d'un algorithme initialement utilisé par M. Ouladsine *et al.* pour l'identification de systèmes [Ouladsine *et al.* 1995]. Le détail de l'algorithme est présenté page 115, et l'organigramme correspondant page 116. Précisons enfin que l'association d'algorithmes génétiques et de réseaux de neurones avait déjà été mise en œuvre par le passé, mais pour la résolution de problèmes plus formels, tels que l'amélioration des méthodes d'apprentissage [Branke 1995] ou bien de la topologie de réseaux [Radcliffe 1993].

3.2.3.2 Prise en compte de la contrainte

Les méthodes heuristiques employées ici ne permettent pas de considérer une contrainte sur les paramètres telle que l'équation de continuité (3.25). Il faut donc prendre en compte celle-ci de manière implicite dans le problème d'optimisation. Pour ce faire, deux solutions se présentent : on peut exprimer l'un des paramètres en fonction des autres *via* l'expression de f_Θ, et de réduire ce faisant d'une dimension la taille de l'espace de recherche. L'alternative consiste à modifier le critère en rajoutant un terme de pénalisation correspondant à cette contrainte :

$$\tilde{J} = J + \beta |f_\Theta(0) - f_\Theta(T)| \tag{3.30}$$

où β est une pondération ajustable. Compte-tenu que l'on dispose ici de l'expression exacte de la fonction de transfert du réseau, et qu'il est donc aisé d'exprimer un paramètre en fonction des autres, on choisira la première méthode, dont on donne le détail ci-après.

N : **taille de la population** ; *[paramètres de la recherche]*
$n_g$max : **nombre maximal de générations** ;
G : **probabilité de survie** ;
M : **probabilité de mutation** ;
Pour i **de** 1 **à** N **faire**
  $\Theta \leftarrow \Theta_{\min} + \text{rand.}(\Theta_{\max} - \Theta_{\min})$;
  $J \leftarrow \bar{R}_z(\Theta)$;
  $\mathbf{\Phi}(i) \leftarrow \Theta$; *[création de la population initiale]*
  $\mathbf{\Psi}(i) \leftarrow J$;
Fin Pour
$n_g \leftarrow 1$;
$\mathbf{\Phi} \leftarrow \{\arg(\min(\Psi)), \ldots, \arg(\max(\Psi))\}$;
[tri de la population par valeurs croissantes du critère]
$\mathbf{\Psi} \leftarrow \{\min(\Psi), \ldots, \max(\Psi)\}$;
$\epsilon \leftarrow \mathbf{\Psi}(N) - \mathbf{\Psi}(1)$;
Tant que ($\epsilon > \epsilon_{\text{stop}}$ **ET** $n_g < n_g$max) **faire**
  [début de la recherche]
  $k \leftarrow NG + 1$;
  $J^* \leftarrow \bar{R}_z(\mathbf{\Phi}(NG))$;
  Tant que ($k \leq N$) **faire**
    $k_1, k_2 \leftarrow \text{rand}(NG)$; *[sélection aléatoire de deux parents]*
    $\mathbf{\Phi}(k) \leftarrow \mathbf{\Phi}(k_1) + \text{rand.}(\mathbf{\Phi}(k_2) - \mathbf{\Phi}(k_1))$;
    $J \leftarrow \bar{R}_z(\mathbf{\Phi}(k))$;
    Si ($J \leq J^*$) **Alors**
      $k \leftarrow k + 1$;
    Sinon
      Si (rand $\leq M$) **Alors**
        $\Phi(k) \leftarrow \Theta_{\min} + \text{rand.}(\Theta_{\max} - \Theta_{\min})$;
        [mutation aléatoire]
        $J \leftarrow \bar{R}_z(\mathbf{\Phi}(k))$;
        Si ($J \leq J^*$) **Alors**
          $k \leftarrow k + 1$;
        Fin Si
      Fin Si
    Fin Si
  Fait
  $\mathbf{\Psi} \leftarrow \{\min(\mathbf{\Psi}), \ldots, \max(\mathbf{\Psi})\}$;
  $\mathbf{\Phi} \leftarrow \{\arg(\min(\mathbf{\Psi})), \ldots, \arg(\max(\mathbf{\Psi}))\}$;
  $\epsilon \leftarrow \mathbf{\Psi}(N) - \mathbf{\Psi}(1)$;
  $n_g \leftarrow n_g + 1$;
Fait
$\Theta^* = \mathbf{\Phi}(1)$; *[individu optimum, fin de la recherche]*

Algorithme 1: Détail de l'algorithme génétique employé

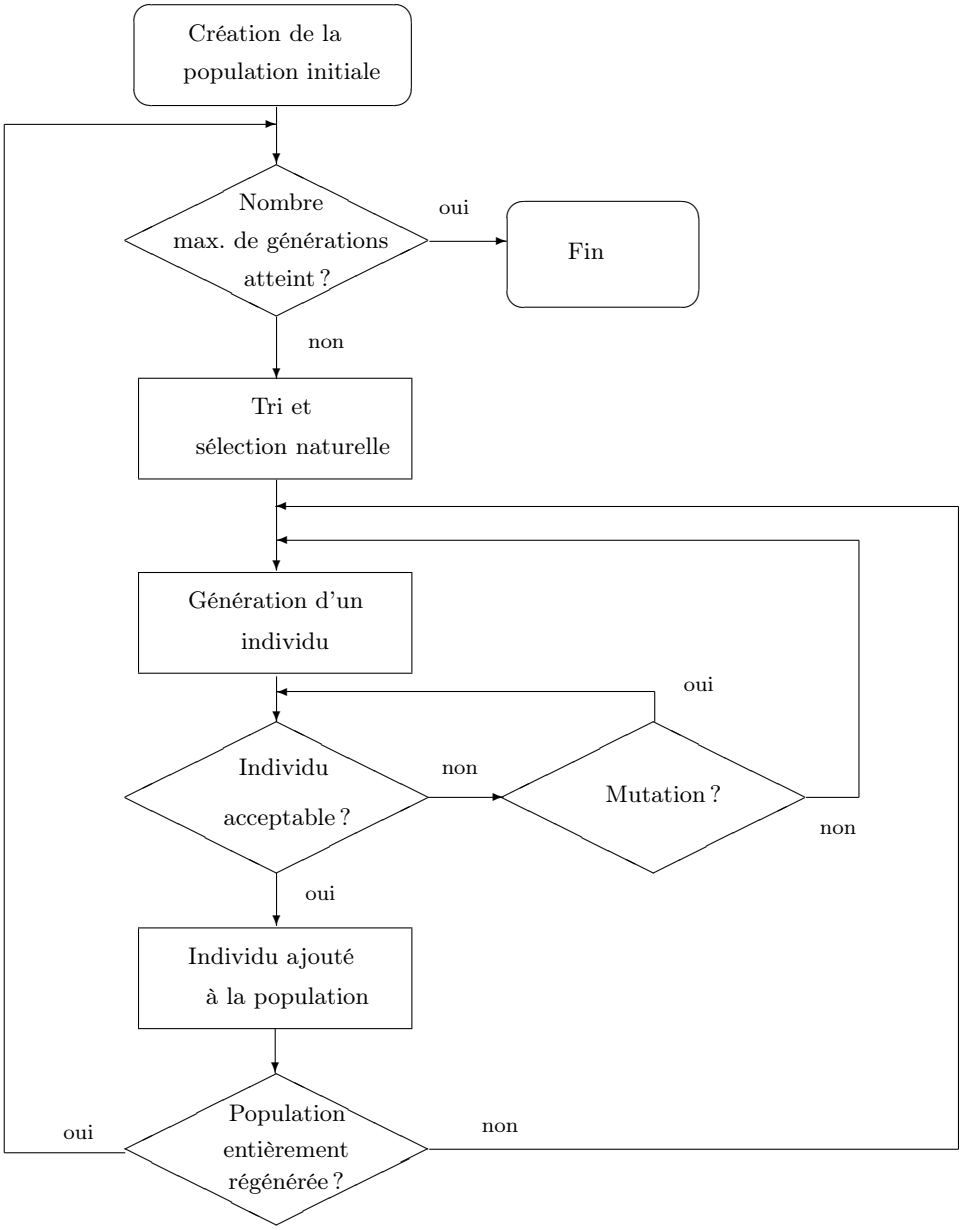


FIG. 3.15 – Organigramme de l'algorithme génétique

On a vu que le transfert associé à un réseau de type perceptron à une couche cachée tel que celui employé pour représenter $\lambda(t)$ et $\nu(t)$ pouvait se mettre sous la forme :

$$f_\Theta(t) = \sum_{i=1}^{n_n} \frac{w_{i+n_n}}{1+e^{-(w_i t+b_i)}} + b_s \tag{3.31}$$

n_n étant le nombre de neurones dans la couche cachée, $\{w_i\}_{i=1\ldots n}$ les poids d'entrée, $\{w_{i+n_n}\}_{i=1\ldots n}$ les poids de sortie, $\{b_i\}_{i=1\ldots n}$ les biais des neurones de la couche cachée et b_s le biais de sortie. L'expression de la contrainte de périodicité $f_\Theta(0) = f_\Theta(T)$ équivaut alors à :

$$\sum_{i=1}^{n_n} \frac{w_{i+n_n}}{1+e^{-(w_i T+b_i)}} - \sum_{i=1}^{n_n} \frac{w_{i+n_n}}{1+e^{-b_i}} = 0 \tag{3.32}$$

$$\Leftrightarrow \sum_{i=1}^{n_n} w_{i+n_n} \left(\frac{1}{1+e^{-(w_i T+b_i)}} - \frac{1}{1+e^{-b_i}} \right) = 0 \tag{3.33}$$

On peut alors exprimer un paramètre en fonction des autres, par exemple w_{2n_n} :

$$w_{2n_n} \left(\frac{1}{1+e^{-(w_{n_n} T+b_{n_n})}} - \frac{1}{1+e^{-b_{n_n}}} \right) + \sum_{i=1}^{n_n-1} w_{i+n_n} \left(\frac{1}{1+e^{-(w_i T+b_i)}} - \frac{1}{1+e^{-b_i}} \right) = 0 \tag{3.34}$$

d'où

$$w_{2n_n} = -\frac{\sum_{i=1}^{n_n-1} w_{i+n_n} \left(\frac{1}{1+e^{-(w_i T+b_i)}} - \frac{1}{1+e^{-b_i}} \right)}{\left(\frac{1}{1+e^{-(w_{n_n} T+b_{n_n})}} - \frac{1}{1+e^{-b_{n_n}}} \right)} \tag{3.35}$$

On s'est donc ramené d'une contrainte que l'algorithme d'optimisation ne pouvait pas prendre en compte à une relation directe entre les différentes composantes de Θ.

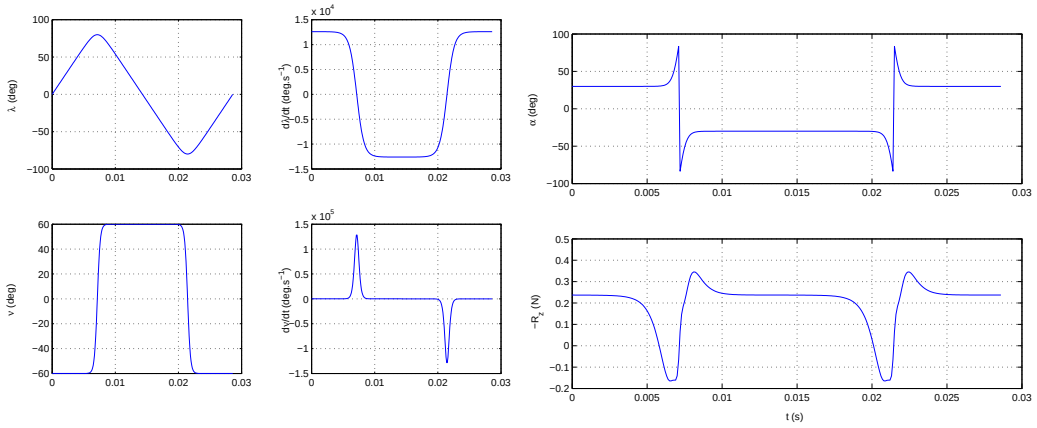
3.2.3.3 Cas d'étude

Différents cas d'étude ont été envisagés ; dans un premier temps, les deux entrées λ et ν ont été optimisées chacune séparément, l'autre entrée étant prise égale à l'une des fonctions de référence λ_{tri} ou ν_{car} définies respectivement en (2.65) et (2.64), avec les valeurs suivantes : $\lambda_m = 80\,^\circ$, $\nu_m = 60\,^\circ$, $k_{\text{tri}} = 4$, $k_{\text{car}} = 10$, $\Phi = 0\,^\circ$. On donne sur les figures 3.16(a) et 3.16(b) les tracés correspondant à ces cinématiques de référence. La portance moyenne correspondante est $-J = 0.1994$ N.

Les algorithmes métaheuristiques n'imposant pas de restrictions quand à la forme (quadratique, etc.) du critère à adopter, on pourra donc définir le critère de base comme étant la portance moyenne au cours d'un cycle, comme dans le cas précédent : $J = \bar{R}_z$ (on rappelle qu'il s'agit bien d'un critère à minimiser, étant donné que $R_z < 0$ lorsque la portance est dirigée vers le haut). Ce critère pourra toutefois être modifié, pour prendre en compte par exemple la nécessité de limiter les vitesses de rotations $\dot{\lambda}$ et $\dot{\nu}$, comme c'est le cas avec des actionneurs réels.

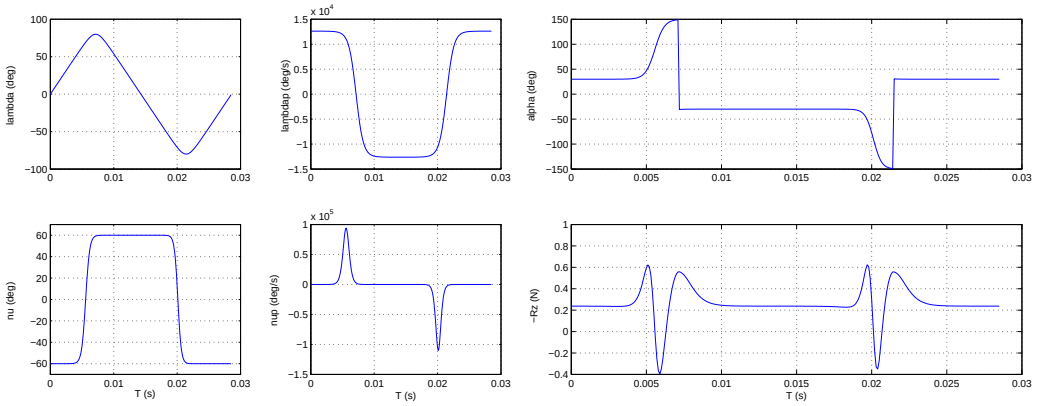
3.2.3.4 Résultats

Ces résultats ont été publiés à l'occasion du 16e congrès IFAC (*International Federation of Automatic Control*) [Rakotomamonjy *et al.* 2005].



(a) Entrées (b) Sorties

FIG. 3.16 – Entrées de référence λ_r et ν_r, incidence et portance correspondantes



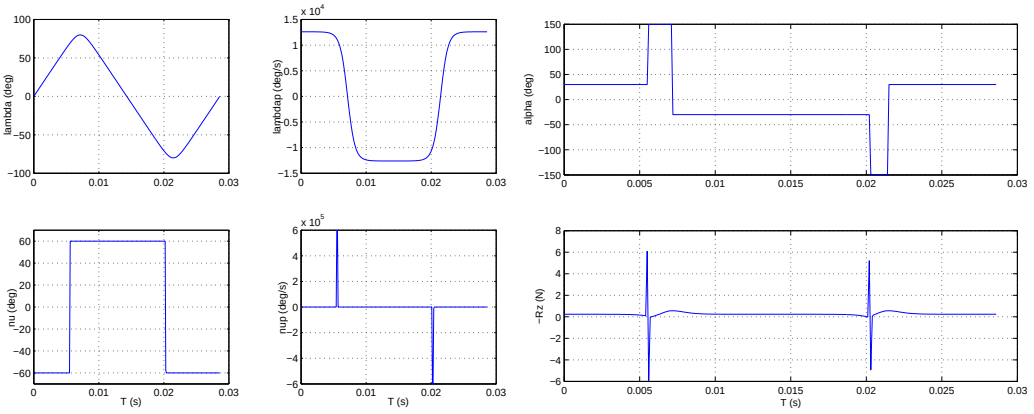
(a) Entrées (b) Sorties

FIG. 3.17 – Optimisation de ν par AG

3.2.3.4.1 Critère simple $J = \bar{R}_z$

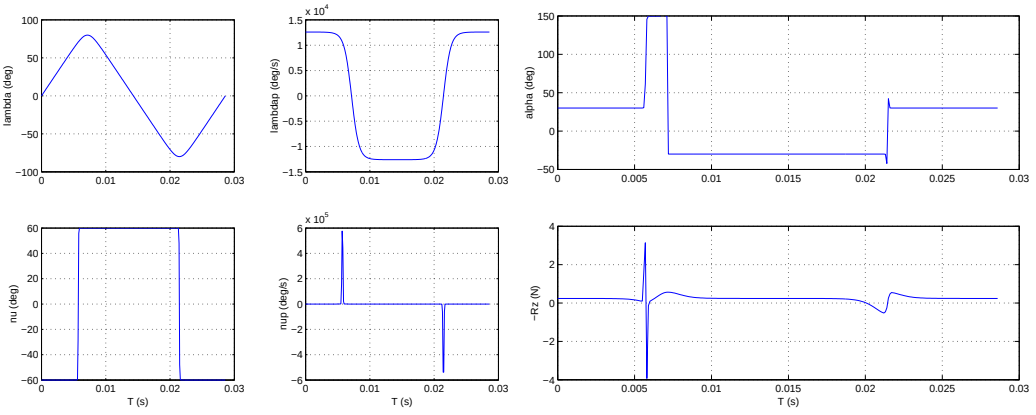
Les résultats obtenus avec l'algorithme génétique pour l'optimisation de la rotation de l'aile $\nu(t)$ sont représentés sur les figures 3.17(a) et 3.17(b). La portance optimale est de $-J^* = 0{,}2649$ N. On peut s'étonner que cette valeur soit inférieure à certains optimums trouvés précédemment par l'algorithme de SQP : il convient donc de préciser que les amplitudes des entrées ont été normalisées durant les optimisations heuristiques afin de faciliter la convergence des algorithmes (60 ° pour ν_m en l'occurrence). On perd le degré de liberté fourni par la variation d'amplitude de l'entrée, mais l'avantage est que l'on concentre la recherche sur la forme uniquement du signal, pour une amplitude donnée.

Les résultats menés avec la recherche aléatoire adaptative sont présentés respectivement sur les figures 3.18(a) et 3.18(b) pour RAA1, et les figures 3.19(a) et 3.19(b) pour RAA2. On constate que la forme optimale est dans les deux cas celle d'un signal carré, en avance de phase sur le battement, et induisant des portances de $-J^* = 0{,}2592$ N (RAA1) et de $-J^* = 0{,}2307$ N (RAA2). Ces portances sont légèrement inférieures à celle obtenue par algorithme génétique (figs. 3.17(a) et 3.17(b)), toutefois l'allure optimale reste dans tous



(a) Entrées (b) Sorties

FIG. 3.18 – Optimisation de ν par RAA1



(a) Entrées (b) Sorties

FIG. 3.19 – Optimisation de ν par RAA2

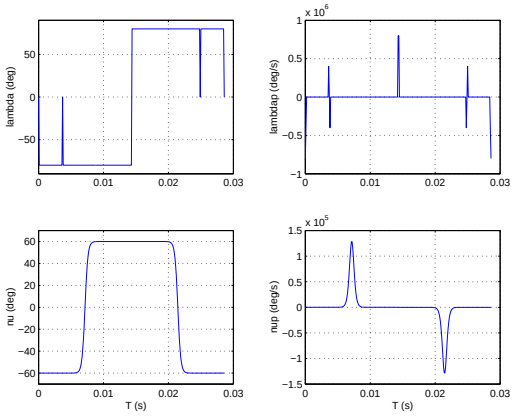


FIG. 3.20 – Optimisation de λ par AG (seules les entrées sont représentées, les sorties correspondantes étant de type non numérique)

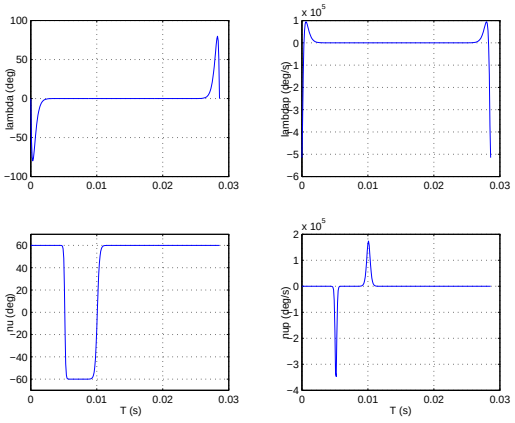
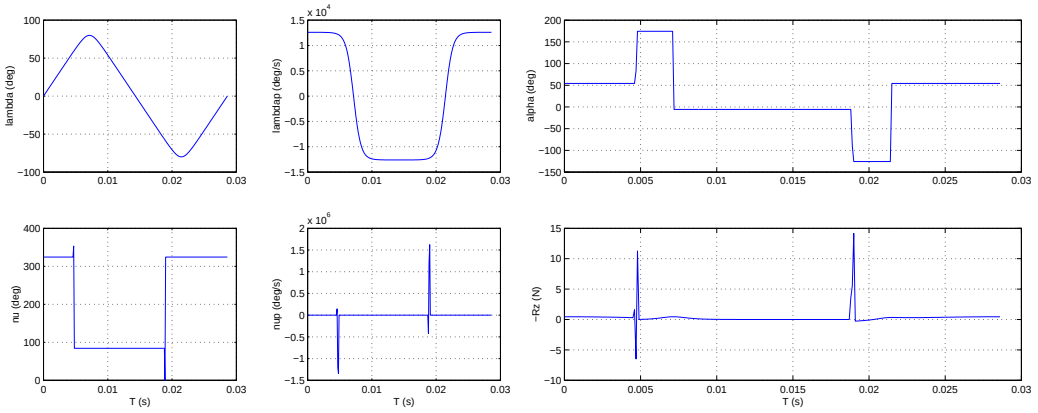


FIG. 3.21 – Optimisation simultanée de λ et ν par AG

les cas en avance de phase par rapport au battement, et le gain de portance par rapport au cas de référence est d'environ 33%.

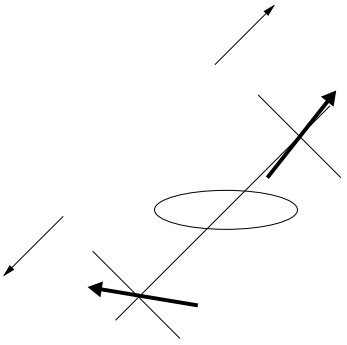
À l'opposé, pour ce qui est de la recherche d'une forme optimale pour le battement $\lambda(t)$, aucun des trois algorithmes n'a fourni de résultat satisfaisant, non pas parce qu'ils ne convergeaient pas, mais plutôt du fait que la solution proposée n'était pas physiquement acceptable : on voit sur la figure 3.20 que l'entrée λ optimale présente des variations très brusques, ce qui, couplé avec la très forte non-linéarité du modèle, entraîne au niveau des sorties la présence de nombreuses valeurs non numériques (`"Not-a-Number"` sous MATLAB). Nous verrons par la suite comment une modification du critère pourra fournir une solution acceptable.

Il a semblé également intéressant de rechercher à optimiser simultanément les deux entrées λ et ν. Pour ce faire, on a considéré comme vecteur des paramètres la concaténation des vecteurs définissant respectivement λ et ν : $\Theta = (\Theta_\lambda\ \Theta_\nu)$. Ici encore, on note (fig. 3.21) que les solutions trouvées ne sont pas acceptables, sans doute pour les mêmes raisons que précédemment (auxquelles on ajoutera un espace de recherche de dimension plus importante).



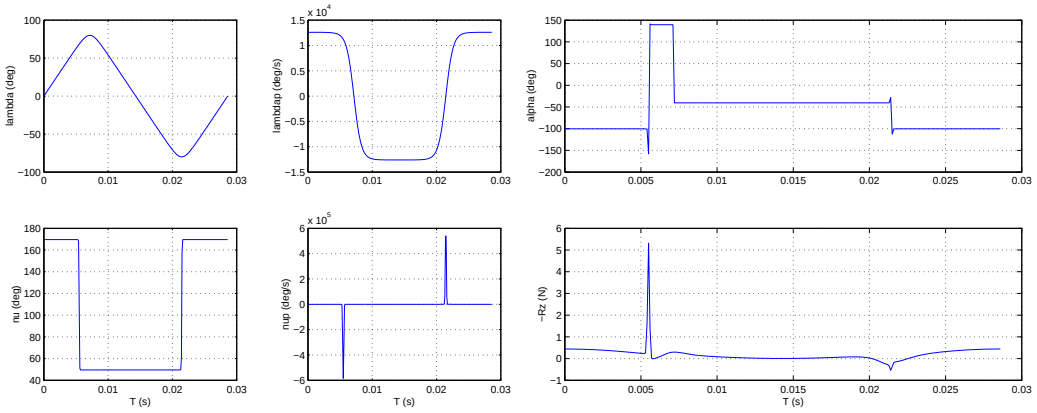
(a) Entrées

(b) Sorties



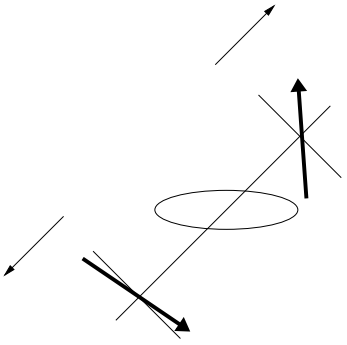
(c) Cinématique (vue latérale, flèche=bord d'attaque)

FIG. 3.22 – Optimisation de ν, $\xi = 45\,^{\circ}$ (AG)



(a) Entrées

(b) Sorties



(c) Cinématique (vue latérale, flèche=bord d'attaque)

FIG. 3.23 – Optimisation de ν, $\xi = 45\,^{\circ}$ (RAA2)

On a testé ces méthodes pour une configuration de vol souvent vue dans le monde animal, à savoir un vol avec un plan de battement incliné à 45 °. On s'attend à obtenir une rotation de l'aile dissymétrique ($\max(\nu) \neq -\min(\nu)$), afin de compenser l'inclinaison du plan de battement, et comme c'est le cas dans la nature pour le vol des libellules par exemple [Wakeling et Ellington 1997a, Savage *et al.* 1979]. On constate que c'est globalement le cas sur les figures 3.22(b) et 3.23(b), avec toutefois des résultats différents suivant la méthode choisie. On obtient en particulier une portance moyenne supérieure avec l'algorithme génétique ($-J^* = 0{,}2980$ N contre $-J^* = 0{,}1902$ N pour la recherche aléatoire adaptative). On a également représenté sur les figures 3.22(c) et 3.23(c) l'allure des cinématiques correspondantes : on constate notamment dans le cas de l'optimisation par algorithme génétique un retournement de l'aile très important, d'une amplitude de 240 °. Un tel mouvement ne se retrouve pas dans le vol naturel animal, étant donné qu'aucune espèce ne dispose d'articulations suffisamment souples pour permettre une telle excursion angulaire. Cette solution originale mériterait donc d'être étudiée plus en détail, de manière expérimentale par exemple.

3.2.3.4.2 Critère composé

Jusqu'à présent seule la portance moyenne avait fait l'objet d'une optimisation. Des situations plus complexes peuvent être envisagées, pour lesquelles on souhaitera optimiser non plus un seul, mais plusieurs critères. Ainsi, durant le vol stationnaire, il serait utile d'adopter des cinématiques qui réduisent les oscillations horizontales, dans le but de faciliter les diverses mesures et acquisitions que le microdrone aurait à accomplir : on passe alors d'un problème d'optimisation mono-objectif à un problème multi-objectif, les grandeurs à minimiser étant d'une part l'opposée de la portance et d'autre part la moyenne quadratique des déplacements horizontaux. Les approches les plus évoluées dans le domaine de l'optimisation multi-objectifs font intervenir la notion d'optimums au sens de Pareto : par définition il s'agit des points en lesquels on ne peut plus améliorer un critère sans en dégrader un autre [Meunier 2002]. Toutefois, la recherche de la frontière Pareto (constituée de l'ensemble des optimums au sens de Pareto) se révélerait dans notre cas assez coûteuse en temps de calcul, sans oublier les risques d'obtenir des résultats numériquement singuliers ou physiquement inacceptables. C'est la raison pour laquelle une méthode d'agrégation a été préférée, qui consiste de manière basique à combiner les différents critères en un seul. Pour ce cas d'étude, on a souhaité conserver la portance moyenne en tant que composante principale du critère final, c'est pourquoi le terme concernant les mouvements horizontaux a été incorporé sous forme d'une pénalisation exponentielle :

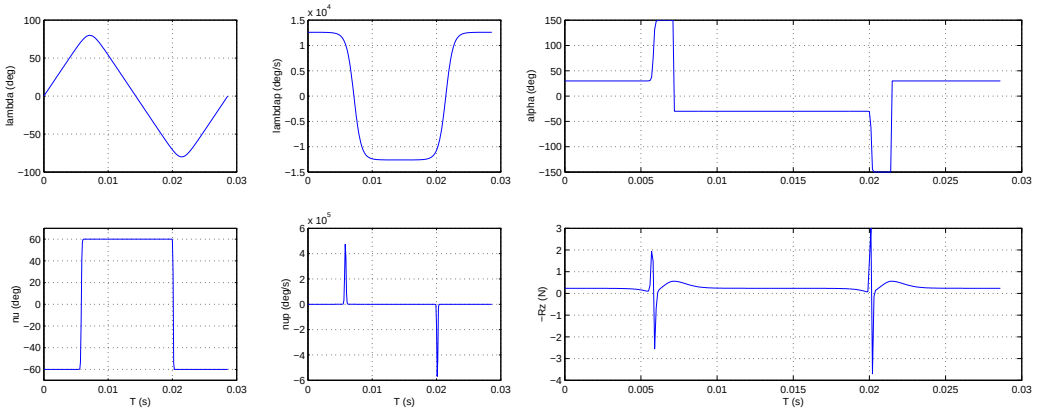
$$J' = J + e^{\beta \overline{u^2}} \tag{3.36}$$

β étant un coefficient de pondération. Les résultats sont donnés fig. 3.24 avec $\beta = 0{,}1$. On constate que la forme du signal est très peu différente de celle obtenue avec le critère standard, ce qui suggère d'ores et déjà qu'il existerait au niveau de la mécanique du vol un découplage entre les efforts verticaux et horizontaux.

On a vu par ailleurs que l'optimisation de la forme du battement λ ne conduisait pas à des résultats exploitables, du fait des trop brusques variations de l'entrée. Afin d'obtenir des résultats physiques, on a limité cette fois-ci l'amplitude de la dérivée de λ, en définissant le nouveau critère :

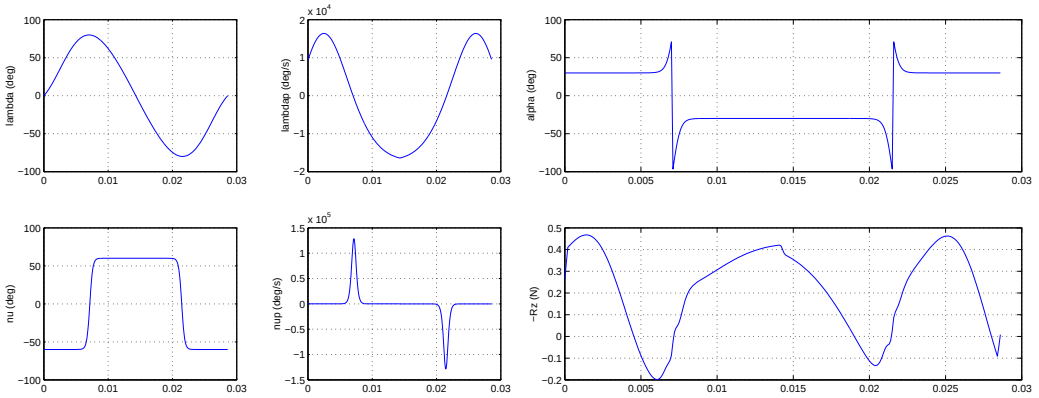
$$J'' = J + e^{\beta \max_{[0;T]}(\dot{\lambda})} \tag{3.37}$$

On constate sur la figure 3.25 que la forme optimale obtenue se rapproche davantage de la sinusoïde que du signal triangulaire initial, avec un gain de portance conséquent



(a) Entrées (b) Sorties

FIG. 3.24 – Optimisation de ν, minimisation de la vitesse quadratique horizontale u^2



(a) Entrées (b) Sorties

FIG. 3.25 – Optimisation de λ, minimisation de $\max(\dot{\lambda})$

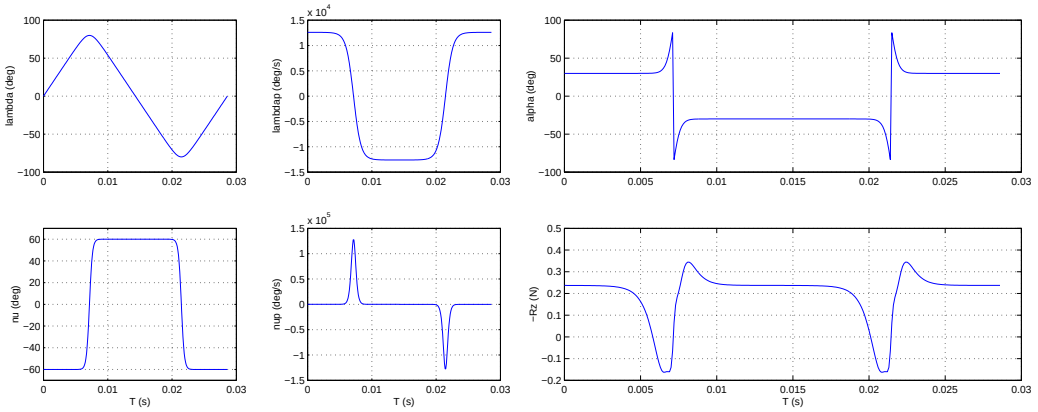
($-J^* = 0{,}2794$ N), soit un gain de 40% : cette allure sera donc à privilégier pour les essais ultérieurs.

On a procédé de même pour la rotation ν, en pénalisant le critère de départ par l'amplitude de la dérivée $\dot{\nu}$:

$$J''' = J + e^{\beta \max_{[0;T]}(\dot{\nu})} \tag{3.38}$$

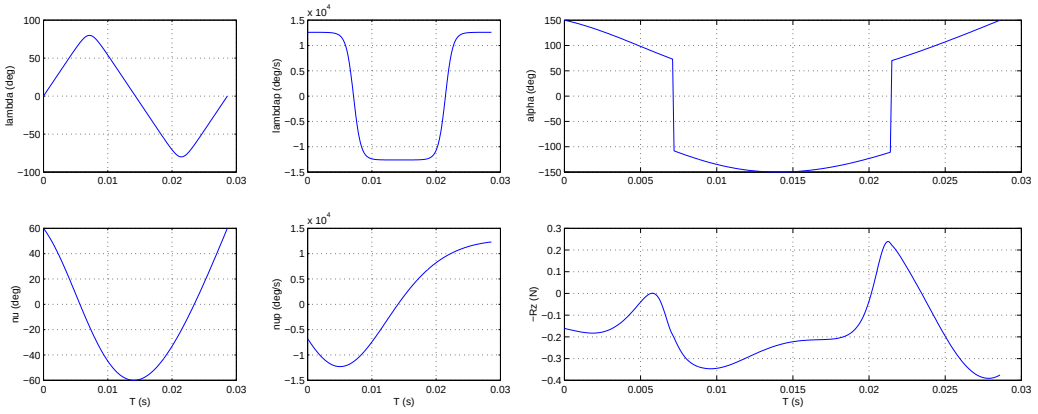
Cette fois-ci, on ne réussit pas à améliorer l'allure initiale (fig. 3.26), ce qui est sans doute dû au fait que c'est la rotation rapide de l'aile (et donc les pics de $\dot{\nu}$) qui est à l'origine d'une part non négligeable de la portance, *via* l'effet de circulation rotationnelle. On sera dans ce cas davantage limité par les capacités physiques (vitesse de rotation) des actionneurs utilisés pour accomplir ce mouvement. Il en est de même pour la figure 3.27, qui présente les résultats pour une pénalisation simultanée de la vitesse horizontale u^2 et de la vitesse de rotation $\max(\dot{\nu})$.

Enfin on a de nouveau considéré un plan de battement incliné à 45 ° , avec cette fois la contrainte supplémentaire de réduire la vitesse horizontale, toujours dans le même but que précédemment (fig. 3.28). On obtient une cinématique très proche de celle obtenue dans le



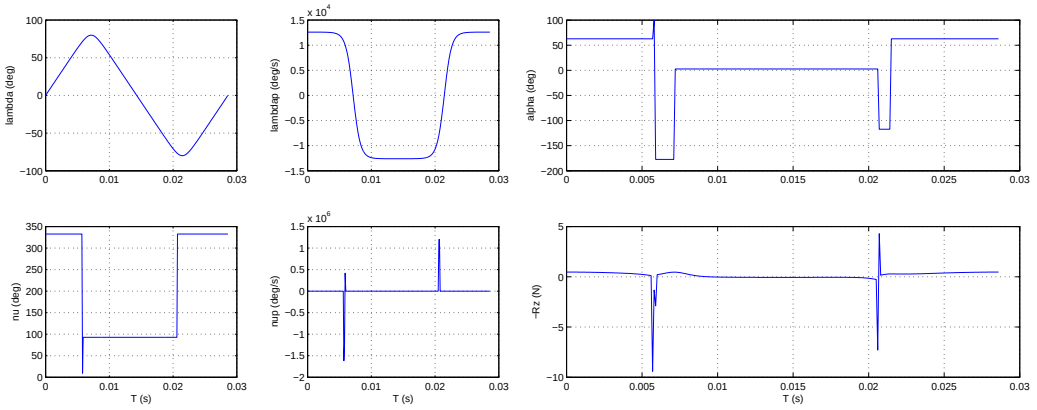
(a) Entrées (b) Sorties

FIG. 3.26 – Optimisation de ν, minimisation de $\max(\dot{\nu})$



(a) Entrées (b) Sorties

FIG. 3.27 – Optimisation de ν, minimisation de u^2 et de $\max(\dot{\nu})$



(a) Entrées (b) Sorties

FIG. 3.28 – Optimisation de ν, $\xi = 45^\circ$, minimisation de $||u||$

cas 3.22(c).

3.2.3.5 Aspects énergétiques

Pour finir, une étude comparative d'un point de vue énergétique a été envisagée. En effet, il a été montré qu'un angle de battement sinusoïdal permet d'améliorer sensiblement la portance, toutefois on ignore si ce gain s'accompagne ou non d'une consommation énergétique accrue. En d'autres termes, on souhaiterait étudier l'influence des différentes cinématiques de battement sur le rendement mécanique du microdrone. Plusieurs grandeurs peuvent être introduites à cet effet. On définira en premier lieu la portance efficace comme la moyenne quadratique de la portance :

$$R_{z\text{eff}} = \sqrt{\frac{1}{T}\int_0^T R_z^2(t)dt} \tag{3.39}$$

Cette grandeur permet de caractériser l'efficacité « absolue » du battement, sans tenir compte des compensations liées aux changements de signe. On peut également introduire la puissance instantanée des efforts aérodynamiques, égale au produit scalaire de la vitesse aérodynamique par l'effort engendré :

$$P = \vec{V}_a.\vec{R}_m \tag{3.40}$$

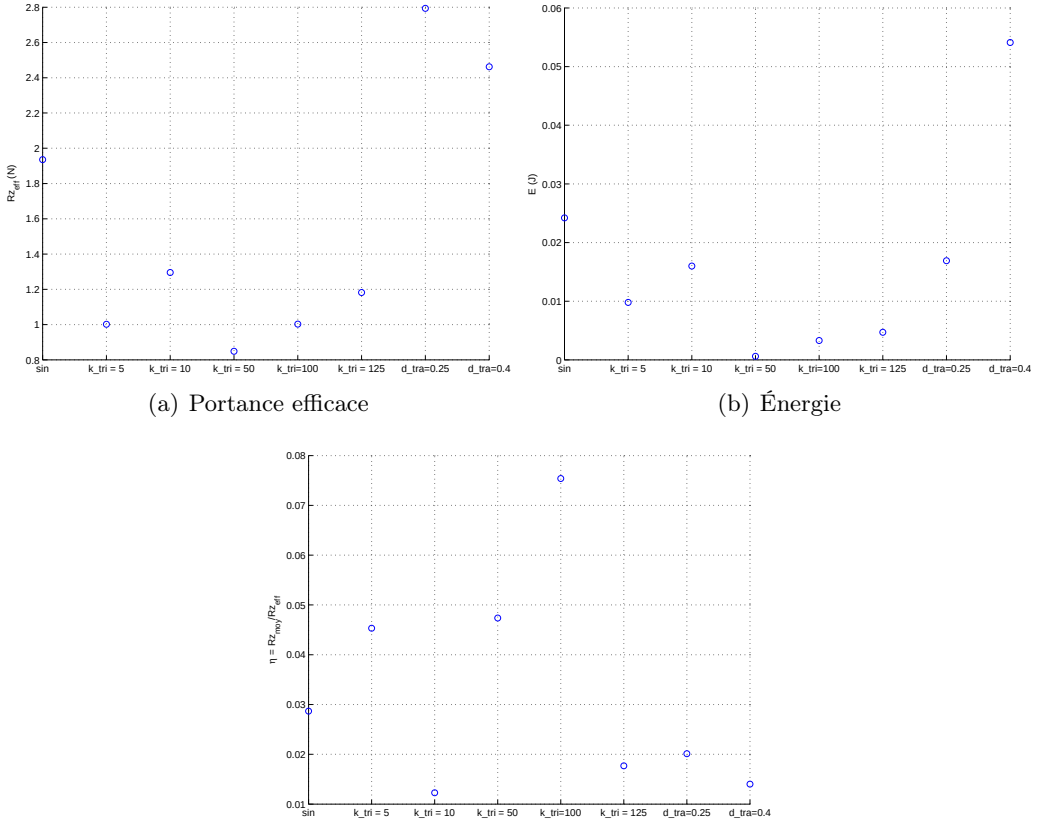
On en déduit alors la consommation énergétique par intégration :

$$E = \int_0^T P dt \tag{3.41}$$

On peut enfin définir un rendement aérodynamique par le rapport de la portance moyenne sur la portance efficace :

$$\eta = \frac{\bar{R}_z}{R_{z\text{eff}}} \tag{3.42}$$

On vérifie que ce rendement est bien inférieur à 1 par construction. On a ensuite calculé ce rendement pour différentes cinématiques de battement, la rotation étant à chaque fois



(a) Portance efficace

(b) Énergie

(c) Rendement

FIG. 3.29 – Influence de la forme du battement sur le rendement

optimisée par rapport à la portance efficace $R_{z\text{eff}}$. Les résultats sont reproduits sur la figure 3.29.

Des battements définis à l'aide d'une fonction trapézoïdale ont en outre été introduits. Il s'agit d'un signal composé de plateaux reliés par une sinusoïde, le paramètre d_{tra} représentant la durée des commutations. Ce type de battement ne se rencontre pas dans la nature, toutefois nous avons voulu juger de son efficacité en termes de rendement énergétique. La figure 3.29 montre clairement qu'un battement triangulaire pour une valeur du paramètre de forme $k_{\text{tri}} = 100$ offre un rendement supérieur aux autres cinématiques : une allure sinusoïdale, même si elle correspond à une portance moyenne plus élevée, ne sera donc peut-être pas à privilégier si l'on souhaite avant tout maximiser le rendement aérodynamique. Rappelons ici que ces résultats sont en toute rigueur fortement liés à la modélisation aérodynamique adoptée dans *OSCAB*, et pourront être confirmés ultérieurement par le biais de mesures expérimentales.

3.3 Conclusion

En l'absence à l'heure actuelle de données précises quant à la configuration d'un futur microdrone issu de *REMANTA*, il n'est pas possible de calquer les mouvements des ailes d'*OSCAB* sur ceux d'une espèce animale donnée. Pour pallier cet inconvénient, nous avons donc, au cours de cette partie de l'étude, trouvé des fonctions optimales pour les

cinématiques des ailes à l'aide d'une approche originale associant réseaux de neurones et métaheuristiques. Parmi les différentes méthodes d'optimisation employées ici, l'algorithme génétique a fourni des résultats légèrement supérieurs, en termes de convergence et de performance, tandis que la première méthode de recherche aléatoire adaptative (RAA1) n'a que peu souvent convergé. Précisons toutefois que l'utilisation de ces algorithmes est rendue délicate par le nombre important de paramètres de réglages, qui conditionnent le comportement de la méthode et qui nécessitent une bonne connaissance préalable de la fonction critère (probabilité d'atteindre un domaine d'une taille donnée, etc.). La complexité et le caractère fortement non-linéaire de notre problème rendent donc difficile cette connaissance *a priori*.

Les résultats ont toutefois confirmé la nécessité d'accomplir la rotation ν en avance de phase par rapport au battement λ. Quant à ce dernier, un choix devra être fait quant à la grandeur à privilégier : la portance seule ou bien le rendement énergétique. Dans le premier cas, c'est une forme sinusoïdale qui devra être employée tandis que dans le deuxième un battement plus raide donnera de meilleurs résultats.