Approximation de calculs éléments finis par un nouveau réseau de neurones

Application à l'optimisation

Jean-Luc Marcelin – Assad Kallassy

Laboratoire Sols, Solides, Structures 3S
UMR CNRS C5521
BP 53 F-38041 Grenoble Cedex 9
Jean-Luc.Marcelin@ujf-grenoble.fr

RÉSUMÉ. *Une stratégie d'approximation de calculs éléments finis par une nouvelle architecture de réseau de neurones est présentée avec des exemples montrant les performances de cette nouvelle architecture, et les perspectives qu'elle peut offrir dans le domaine de l'optimisation des structures mécaniques.*

ABSTRACT. *The research strategy consists in substituting, for finite element calculations in an optimization process, an approximate response of a new neural networks. Some tests show the accuracy of this new architecture.*

MOTS-CLÉS. *Optimisation - Eléments finis - Réseaux neuronaux.*

KEY WORDS. *Optimization - Finite elements - Neural networks.*

Revue européenne des éléments finis. Volume 8 – n° 2/1999, pages 201 à 227

1. Introduction

L'optimisation des structures mécaniques est souvent pénalisée par le coût des analyses par éléments finis, que l'on propose dans cet article de remplacer par des approximations neuronales dans le but de réduire ces coûts. L'objet de ce travail est de présenter un nouveau réseau de neurones permettant d'améliorer ainsi l'efficacité de l'optimisation. Dans deux précédentes publications [MAR 96b] et [MAR 99], on a montré comment on pouvait remplacer graduellement au début d'un processus stochastique très gourmand en temps de calcul (de type algorithme génétique) les analyses par éléments finis par des approximations utilisant un réseau de neurones standard. Dans ce processus, les premières étapes de l'optimisation permettent d'effectuer l'entraînement (ou apprentissage) du réseau de neurones qui remplace ensuite les calculs par éléments finis pour les étapes ultérieures, avec possibilité d'affinage ou de réactualisation du réseau de neurones en cours d'optimisation, comme l'indique la figure 1.

Néanmoins, dans les conclusions des articles précités, il est souligné que le réseau de neurones utilisé (qui est un réseau standard), à savoir le Perceptron Multicouche (PMC) présente un certain nombre de limitations dans son utilisation et ne donne pas entièrement satisfaction pour l'approximation des calculs par éléments finis, ce qui nous a conduit à développer une nouvelle architecture (à notre connaissance) plus adaptée à l'approximation de calculs par la MEF. C'est cette nouvelle architecture qui est présentée ici.

On verra même que ce nouveau réseau de neurones peut permettre d'effectuer directement l'optimisation, comme on le montrera dans la deuxième partie de cet article. En effet, le nouveau réseau de neurones présenté autorise dans certains cas l'explicitation des surfaces de réponses. Notre travail s'inscrit ainsi dans un domaine de recherche en pleine actualité qui est celui des méthodologies de construction des surfaces de réponse. Ces recherches sont actuellement motivées d'une part par l'extension du champ d'application des méthodes d'optimisation structurale et, de manière plus générale, par l'ensemble des problèmes où l'explicitation des surfaces de réponse s'avère pertinent. Il est à signaler que ce travail a fait l'objet d'une thèse [KAL 98], disponible sur demande, et dans laquelle on trouvera une liste importante de références bibliographiques. Rappelons que l'utilisation de réseaux de neurones pour la modélisation de structures mécaniques donne, depuis qu'ils sont utilisés dans cette optique, de bons résultats comme on peut le constater depuis quelque temps déjà dans la littérature [BER 92], [SZE 93], [HAJ 94] et [SZE 94]. De plus, cette thématique sur l'utilisation des réseaux de neurones en calcul des structures est d'actualité, comme en témoignent les articles récents sur ce thème [CAO 98], [GHA 98], [HEN 98] et [FUR 98], pour n'en citer que quelques-uns.

Après une description de la méthode utilisée, des exemples permettant de se faire une idée des performances de cette nouvelle architecture seront développés, pour l'optimisation notamment. Avant de rentrer dans le vif du sujet, on rappelle ci-après brièvement la terminologie utilisée pour les réseaux de neurones.

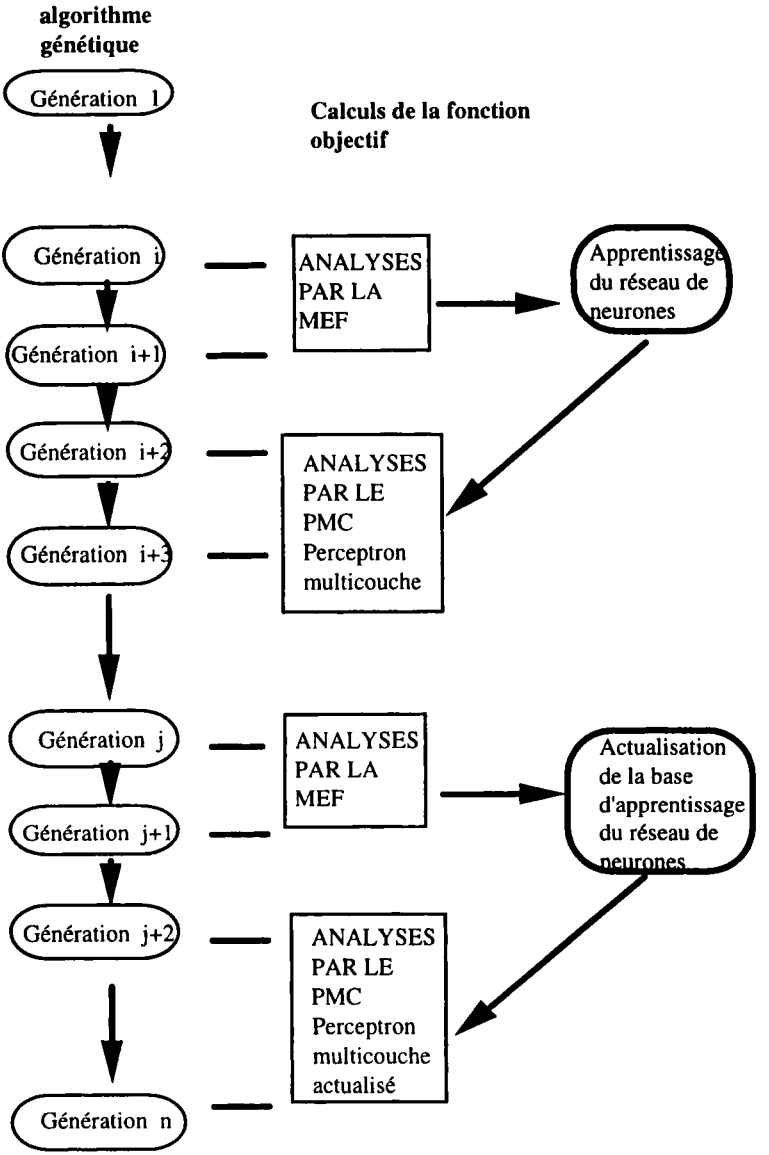

Figure 1. *La stratégie utilisée*

2. Présentation du nouveau réseau

2.1. *Les méthodes utilisées*

2.1.1. *Les réseaux de neurones*

Les réseaux de neurones artificiels s'inspirent dans leur fonctionnement des réseaux de neurones biologiques auxquels ils empruntent une grande partie du vocabulaire. On trouvera dans l'excellent ouvrage [JOD 94] le détail de la théorie. Nous n'en donnons ici que les principes. L'utilisation d'un réseau de neurones pour la simulation ou la modélisation se fait en deux temps : une phase dite d'apprentissage en utilisant des résultats de calculs par éléments finis, suivie d'une phase de calcul ou de généralisation. Dans le cas présent, le réseau de neurones doit pouvoir estimer une fonction objectif ou fonction coût en fonction des variables d'entrée ou de conception. Pour décrire un réseau de neurones, il suffit de connaître le modèle de neurone et l'organisation des connexions entre ces neurones.

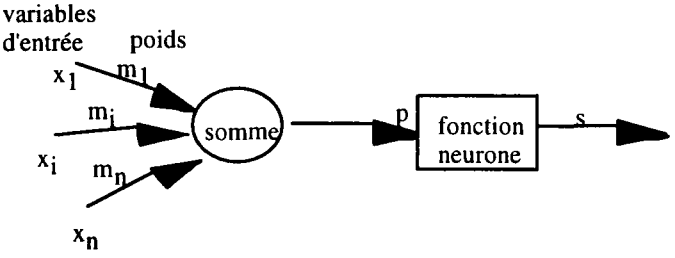

Figure 2. *Neurone élémentaire*

Un neurone est modélisé par deux opérateurs (figure 2). Un opérateur de sommation qui élabore un potentiel p égal à la somme pondérée des entrées x_i de la cellule (ce qui se traduit par l'optimisation des poids en phase d'apprentissage) et un opérateur qui calcule l'état de sortie s = f(p) du neurone en fonction de son potentiel (f est appelée la fonction neurone et peut être linéaire ou non, souvent de forme sigmoïde). Les entrées x_i sont des sorties de neurones du même réseau, ou d'une autre couche, éventuellement directement les entrées extérieures. Un réseau complexe peut être partagé en plusieurs couches et dans cette succession de couches, on peut définir un sens de transfert de l'information. Les connexions peuvent être alors directes ou récurrentes. Par ailleurs, à l'intérieur d'une même couche ou entre deux couches, les connexions peuvent être partielles ou totales. La phase d'apprentissage consiste à optimiser ou adapter au moyen d'une règle d'apprentissage les poids affectés à chaque liaison ; on utilise pour ce faire un échantillon d'apprentissage, c'est-à-dire des solutions qui sont déterminées au préalable par la méthode des éléments finis. Le principal critère est d'obtenir une erreur minimale pour l'évaluation de la fonction.

2.1.2. *La nouvelle architecture*

2.1.2.1. Les fondements mathématiques de la nouvelle architecture (NA) dans le cas d'une seule variable

Pour rendre la présentation plus claire, on donne ci-après les fondements mathématiques du nouveau réseau de neurones dans le cas d'une seule variable, puis les principes dans le cas de plusieurs variables. Dans le cas d'une fonction ou d'une réponse d'un système dépendant d'une seule variable, le problème consiste à déterminer une fonction d'approximation F entre la variable explicative X et la réponse correspondante Y (variable expliquée). L'information sur le modèle est donnée par un certain nombre n de points qui sont les vecteurs d'apprentissage (X1,Y1), (X2,Y2), ..., (Xn,Yn), et l'objectif est de déterminer une relation statistique entre Xi et Yi. Ce type d'analyse n'est autre qu'une analyse de régression et permet d'obtenir des estimations de valeurs moyennes ou des prévisions de valeurs individuelles de la variable Y à partir des valeurs de la variable X. Bien que la démarche présentée ici ressemble du point de vue calculatoire à celle de la régression mathématique, elle va en fait consister à être capable de déterminer dynamiquement l'ensemble des fonctions de base les plus pertinentes et la dimension de cet ensemble. Par conséquent, l'espace des fonctions pouvant être générées par cette méthode sera plus étendu que celui obtenu par utilisation simple d'une technique de régression mathématique.

La fonction élémentaire de base f utilisée n'est autre que la sigmoïde exponentielle : $f = 1 / (1 + e^{-x})$. Elle est bornée entre 0 et 1 et elle est strictement croissante. Elle présente deux paliers stables aux extrémités en lesquels la variation de la réponse de la fonction devient faible. Comme $f(-3) = 0{,}047$ et $f(3) = 0{,}953$, on considère que la fonction élémentaire f prend sa plus faible valeur au point $x = -3$ et sa plus grande valeur au point $x = 3$. On restreint ainsi la partie active de la fonction élémentaire entre -3 et 3. C'est la non-linéarité de la partie active de f qui permet d'établir une bonne approximation des parties non linéaires du domaine de recherche. L'objectif est de trouver la fonction qui minimise l'erreur Ey calculée par $Ey = \Sigma(Yd - Y)^2$. Yd est la valeur désirée et Y la réponse approximative du modèle $Y = F(X)$, cette sommation étant effectuée sur tout l'ensemble des vecteurs d'apprentissage. Dans un problème de régression mathématique, F est donnée comme étant la combinaison linéaire des fonctions de base (1, F1, F2, ..., Fm) : $F = k0 + k1F1 + k2F2 + ... + kmFm$.

Pour expliquer le principe qui a servi à construire le nouveau réseau de neurones on va commencer par le cas élémentaire d'une seule fonction de base, puis on passera à l'utilisation de plusieurs fonctions de base dans la fonction F et à l'ajustement des coefficients de F. Dans le cas de l'utilisation d'une seule fonction de base non constante ($m = 1$) égale à la fonction sigmoïde ($F1 = F$), le problème revient à déterminer deux paramètres $by = k0$ et $ky = k1$. Dans un repère cartésien illustrant le graphe de F, by représente la translation de f suivant l'axe Y et ky représente le coefficient de l'affinité orthogonale de f perpendiculairement à l'axe X. Ainsi, pour donner plus de souplesse à la fonction de base élémentaire, on ajoute deux coefficients bx et kx représentant respectivement la translation de la fonction f par rapport à l'axe X et l'affinité orthogonale perpendiculairement à l'axe Y telle

que : F1 = f ((X - bx) / kx). L'opération ci-dessus peut être traduite par le fonctionnement d'un seul neurone schématisé sur la figure 3 et dont la réponse Y est donnée par l'équation suivante :

$$Y = F(X) = ky\ f((X - bx) / kx) + by$$

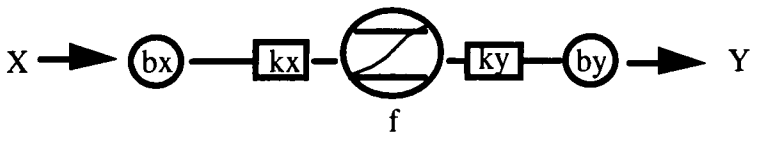

Figure 3. *Configuration d'un seul neurone.*

bx est appelé le seuil à l'amont et kx le poids à l'amont. La valeur (X - bx) / kx sert d'entrée à la fonction d'activation f dont la réponse F (appelée activation du neurone) est à son tour multipliée par le poids à l'aval ky et augmentée du seuil à l'aval by.

Cette structure de neurone est utilisée pour ajuster le nuage de points (X1,Y1), (X2,Y2), ..., (Xn,Yn) en optimisant les paramètres en amont et en aval du neurone afin de minimiser l'erreur de sortie Ey. Autrement dit, il s'agit de trouver les valeurs optimales de bx, by, kx, et ky qui minimisent la somme des distances verticales entre les différents points (Xi,Yi) et la courbe F(X). Pour cela, on propose de déterminer les paramètres du neurone en tenant compte de la minimisation de l'erreur à l'entrée Ex du neurone : $Ex = \Sigma(Xd - X)^2$. Xd est l'entrée désirée ou exacte Xi, et X est l'entrée du neurone qui donne comme sortie la valeur correspondante Yi : $X = kx\ f^{-1}((Y - by) / ky) + bx$. Ex n'est autre que la somme des distances horizontales entre les différents points (Xi,Yi) et la courbe F(X). Minimiser Ex revient à donner à F(X) la même allure que la distribution des points (Xi,Yi).

Cas de plusieurs fonctions de base :

La fonction F telle qu'elle a été définie ne peut approximer que des ensembles de points variant d'une façon monotone. Dans le cas général, on utilise plusieurs fonctions de base. La figure 4 montre par exemple le graphe d'une fonction F(X) admettant deux fonctions de base : F = K0 +k1F1 + k2F2. La fonction F possède alors 7 paramètres à optimiser : le seuil à l'aval by =k0, les poids à l'aval ky1 = k1 et ky2 = k2 de F1 et F2 respectivement, les seuils et les poids à l'amont bx1, kx1, bx2 et kx2 de F1 et F2 respectivement. Pour que F puisse approximer le nuage de points de la figure 4, il faut que le seuil à l'amont bx1 de F1 tombe dans la région ascendante du nuage des points et le seuil à l'amont bx2 de F2 dans la région descendante. Les poids et les seuils en amont vont ainsi obéir à certaines contraintes pour que la partie active de chaque fonction de base couvre un domaine spécifique au neurone appelé le domaine de spécialisation du neurone. Dans le cas général où

la fonction F contient m fonctions de base, la valeur de la fonction F peut être considérée comme la réponse d'un réseau de m neurones où l'activation F de chaque neurone représente la valeur d'une fonction de base.

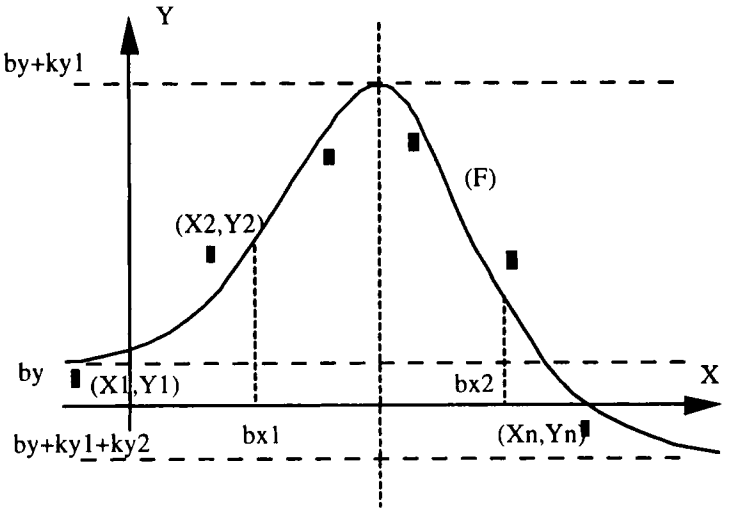

Figure 4. *Exemple à deux neurones.*

L'ajout de neurones n'a pas uniquement pour justification la présence de variations de pente dans la distribution des points, l'enrichissement du réseau par des neurones est aussi nécessaire pour augmenter la précision de l'approximation. Une fois les paramètres du neurone ajustés, on calcule les erreurs moyennes absolue et relative des neurones définies comme :

– l'erreur absolue moyenne du neurone i : c'est l'erreur moyenne à la sortie du réseau mais limitée aux ni points tombant dans le domaine de spécialisation Di du neurone i : $Ea^i = 1/ni\ \Sigma(Yd - Y)^2$

– l'erreur relative moyenne du neurone i : c'est la moyenne sur les ni points tombant dans le domaine de spécialisation du neurone i du carré de la différence de la réponse du réseau et de celle obtenue en éliminant le neurone i :

$Er^i = 1/ni\ \Sigma(Yei - Y)^2$, Yd étant la réponse désirée, Y la réponse du réseau et Yei celle obtenue en éliminant de la réponse du réseau l'activation Fi du neurone i. La configuration du réseau peut être définie en se basant sur les valeurs des erreurs moyennes absolue et relative des neurones. Si l'erreur absolue est jugée importante, le domaine de spécialisation du neurone est divisé en deux domaines et un neurone est affecté à chacun. En revanche, si l'erreur relative est jugée très faible, on élimine le neurone et on répartit le domaine de spécialisation correspondant aux neurones adjacents. C'est ainsi que l'on peut déterminer le nombre optimal de neurones nécessaire pour atteindre une précision donnée du réseau.

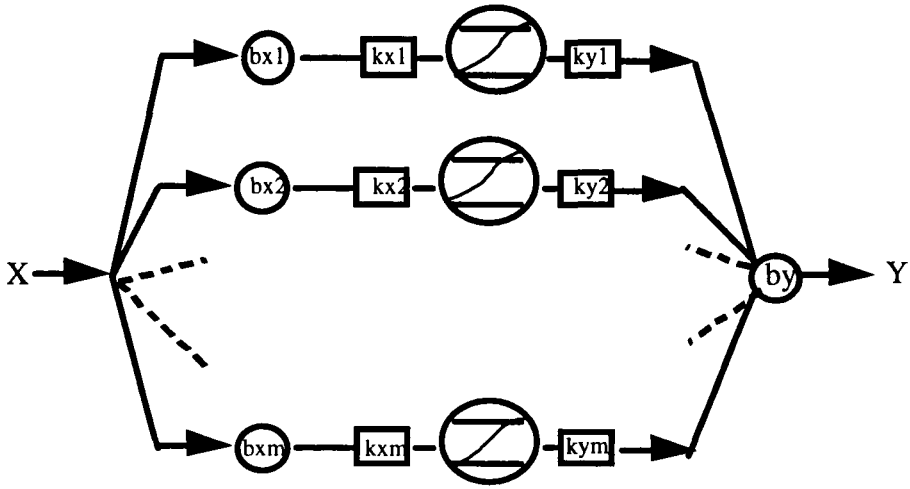

Figure 5. *Schématisation d'un réseau de neurones à une seule variable.*

Méthodes d'apprentissage de la NA :

Le réseau de neurones pour le cas d'une seule variable (figure 5) admet 3m+1 paramètres à ajuster (avec m le nombre de neurones du réseau) qui sont : m seuils à l'amont bx, m poids à l'amont kx, m poids à l'aval ky, 1 seuil à l'aval by. On va présenter dans ce paragraphe les différentes méthodes utilisées pour ajuster les paramètres du réseau. Rappelons que la réponse du réseau Y est une fonction de l'entrée X, qui s'exprime par : $Y = F(X) = by + ky1\ f((X - bx1) / kx1) + ... + kym\ f((X - bxm) / kxm)$. L'objectif principal de l'apprentissage est d'optimiser les paramètres du réseau afin de minimiser l'erreur à la sortie Es. Pour initialiser le calcul et ne connaissant *a priori* aucune information sur la nature de la fonction à estimer, on fixe arbitrairement le nombre de neurones m du réseau. Les m neurones subdivisent équitablement le domaine de recherche de X en m intervalles de spécialisation Di. Le domaine de recherche de X est déterminé par la plus petite et la plus grande valeur de X dans les vecteurs d'apprentissage. Le seuil en amont de chaque neurone est initialisé comme étant le milieu du domaine de spécialisation Di correspondant. On affecte au poids en amont une valeur égale au 1/6 de la longueur du domaine Di de telle façon que la partie active du neurone enveloppe tout le domaine Di (rappelons que la partie active de la fonction sigmoïde élémentaire varie entre -3 et 3). Pour initialiser les poids en aval, on minimise Ey en gardant les paramètres en amont fixes. Remarquons que Es est une fonction quadratique des paramètres en aval du réseau. Si les paramètres amont sont fixes, ce calcul revient à une méthode de régression mathématique pure et simple. Une fois la phase d'initialisation des paramètres en amont terminée et les paramètres en aval du réseau déterminés, on continue à minimiser l'erreur à la sortie en ajustant les paramètres du réseau. Pour ceci, on utilise :

– la méthode des moindres carrés, permettant de minimiser Es en fixant les paramètres en amont.

– la méthode du recuit simulé, tolérant l'augmentation de Es au profit de la diminution de l'erreur à l'entrée du réseau Ee.

– la méthode de la descente du gradient permettant de minimiser Ey en contrôlant l'évolution des paramètres en amont.

2.1.2.2. Les principes généraux de la nouvelle architecture (NA) dans le cas de plusieurs variables

On donne ci-après la généralisation de la théorie précédente au cas de plusieurs variables. Rappelons que la démarche générale est la construction d'une approximation par morceaux dans laquelle chaque neurone se spécialise dans l'approximation d'une portion du domaine (figure 6). En d'autres termes, la nouvelle architecture s'appuie sur l'idée d'une spécialisation des neurones par sous-domaines de variation des variables d'entrée du réseau. Cette structure est, à notre connaissance, originale pour un réseau de neurones. L'intérêt essentiel de cette approximation par sous-domaines particulière est de pouvoir décrire des fonctions fortement non-linéaires ; de plus, dans cette architecture, on arrive à mieux comprendre et analyser le rôle de chaque neurone, ce qui nous conduit à pouvoir contrôler le nombre de neurones nécessaire pour atteindre un certain degré de précision du réseau. Cette approche n'est évidemment pas sans rappeler le concept d'approximation par sous-domaines des éléments finis, mais s'en différencie néanmoins.

La première étape de la démarche consiste à ajuster les paramètres de tous les neurones de façon à minimiser l'erreur de sortie. La configuration du réseau est ajustée dans un deuxième temps. Cela revient à modifier le nombre de neurones et leurs domaines de spécialisation. Pour cela, l'algorithme (figure 7) passe par le calcul des erreurs moyennes relatives et absolues des neurones, la détermination de l'état des neurones dans la configuration suivante du réseau et l'ajustement des limites de spécialisation et des paramètres amonts des différents neurones. Quelques éléments sur l'apprentissage de cette nouvelle architecture (NA) et sur l'organigramme général de la NA dans le cas de plusieurs variables sont donnés ci-après.

2.1.2.3. Apprentissage de la NA (figure 6)

On retrouve ici les principes tels qu'ils ont été définis dans le cas d'une seule variable. Le nombre de neurones m_i de chaque variable est initialisé par l'utilisateur d'une façon aléatoire ou d'après une expérimentation numérique antérieure. Les neurones pour chaque variable subdivisent équitablement le domaine de définition relatif à chaque variable en des domaines de spécialisation. Le domaine de définition de la variable i est limité par la plus faible valeur de la variable i dans le corpus des vecteurs d'apprentissage et la plus grande valeur. Le seuil en amont de chaque neurone est initialisé comme le milieu de son domaine de spécialisation et

son poids amont est arbitrairement initialisé comme le 1/6 de l'amplitude de son domaine (comme dans le cas d'une seule variable). Ensuite on fixe l'ordre maximal de développement des combinaisons des neurones pour déterminer les poids aval à utiliser et leur nombre. Pour initialiser les poids en aval, on minimise l'erreur à la sortie du réseau (notée E_y) qui est une fonction quadratique des poids aval. On est ainsi amené à l'utilisation de la méthode des moindres carrés. Une fois terminées les phases d'initialisation des paramètres en amont et de détermination des paramètres en aval du réseau, l'erreur absolue à la sortie est minimisée en ajustant les paramètres du réseau et en utilisant, suivant les cas, soit la méthode du recuit simulé, soit celle de la descente du gradient.

De plus, l'utilisation du recuit simulé permet aussi d'ajuster les paramètres du réseau en tenant compte à la fois de la minimisation de l'erreur à l'entrée et à la sortie. Désignons par E^i_{Xj} l'erreur à l'entrée du neurone j de la $i^{ème}$ variable. E^i_{Xj} est la somme sur tous les vecteurs d'apprentissage du carré de la différence entre l'entrée exacte X_i et l'entrée O^i_j provoquant la sortie désirée Y. O^i_j du $j_{ème}$ neurone et de la ième variable pour un vecteur d'apprentissage est calculée en présentant à l'entrée de tous les autres neurones du réseau l'entrée exacte et à la sortie du réseau la sortie exacte et en rebroussant chemin vers l'entrée du $j_{ème}$ neurone de la $i^{ème}$ variable. Cependant, comme la fonction d'activation f est non surjective, O^i_j n'existe pas pour tous les vecteurs d'apprentissage. C'est pourquoi la comparaison de l'erreur à l'entrée des neurones pour deux configurations différentes porte sur les vecteurs d'apprentissage possédant une valeur O^i_j dans les deux cas. L'erreur à l'entrée E^i_X du réseau pour la variable i est définie comme la somme directe des erreurs à l'entrée de tous les neurones de cette variable. Par la méthode du recuit simulé, on change alternativement les paramètres en amont et en aval et on génère à chaque fois un certain nombre de solutions potentielles. Pour juger de l'acceptation ou du rejet d'une solution, on calcule à chaque fois les erreurs à l'entrée et à la sortie du réseau.

Pour appliquer la méthode de la descente du gradient dans le but de minimiser l'erreur à la sortie du réseau E_Y , l'erreur de sortie est différenciée par rapport à tous les paramètres du réseau. L'ajustement des paramètres du réseau s'effectue en contrôlant leur évolution afin d'éviter un problème de paralysie. Cela se fait en contrôlant les valeurs des seuils en amont du réseau, ces derniers étant contraints à ne pas quitter le domaine de spécialisation du neurone correspondant. Ainsi, si un seuil amont se trouve à l'extérieur du domaine et si la méthode de la descente du gradient lui permet de s'approcher de son domaine, il se déplace du pas déterminé, sinon il reste immobile. Le déplacement maximal de chaque seuil est déterminé en divisant l'amplitude du domaine du neurone correspondant par un paramètre prédéfini et relatif à la variable i.

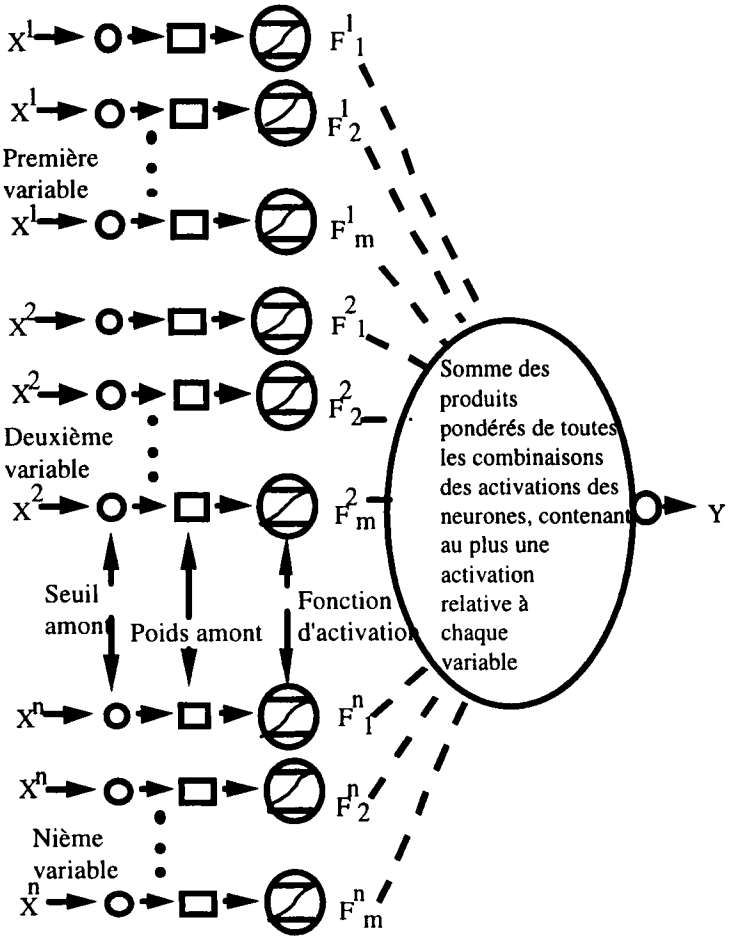

Figure 6. *Schématisation de la nouvelle architecture (pour la compréhension, se reporter aux figures 3 et 5).*

2.1.2.4. Organigramme général de la NA (figure 7)

Une fois le réseau entraîné, l'efficacité de chaque neurone est étudiée afin de trouver la configuration optimale du réseau qui approxime la fonction objective avec une précision donnée. Pour cela, on passe par une étape de détermination des états des neurones et une étape de reconfiguration de la structure du réseau. Les erreurs moyennes absolues et relatives nécessaires pour juger de la pertinence des neurones ont les définitions suivantes :

– erreur moyenne absolue : c'est la moyenne sur tous les vecteurs d'apprentissage dont la variable correspondante tombe dans le domaine de spécialisation du neurone, du carré de la différence entre la sortie du réseau et la sortie désirée.

– erreur moyenne relative : c'est la moyenne sur tous les vecteurs d'apprentissage dont la variable correspondante tombe dans le domaine de spécialisation du neurone, du carré de la différence entre la sortie du réseau et celle obtenue en annulant dans la réponse du réseau l'activation du neurone en cause.

Ainsi les trois étapes d'entraînement, de détermination des états des neurones et de reconfiguration de la structure du réseau sont effectuées conformément à la figure 7.

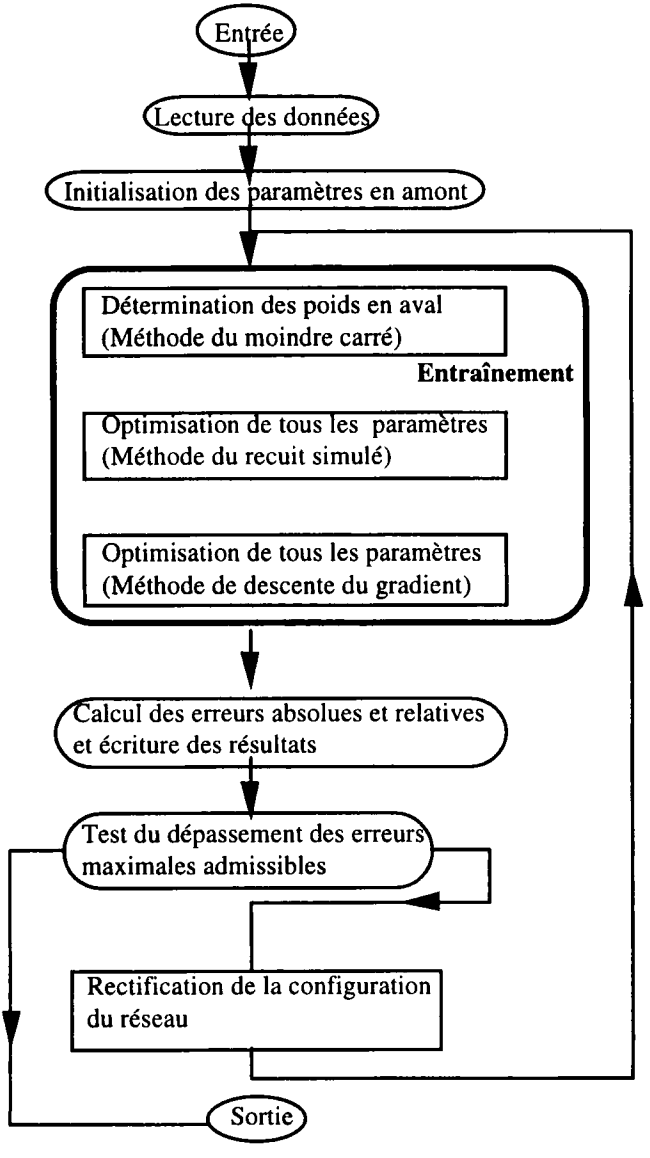

Figure 7. *Organigramme général*

2.1.3. *Comparaison avec un réseau PMC dans le cas de la synthèse dimensionnelle de mécanismes générateurs de trajectoires.*

Le but de cet exemple est de comparer les performances de la nouvelle architecture à un réseau choisi parmi les plus performants qui existent actuellement et sur lequel se concentrent les efforts des recherches connexionnistes. Le réseau en question est un réseau de type Perceptron Multi Couche (PMC). C'est un réseau à couches et à apprentissage supervisé. Ce réseau est décrit dans la référence [JOD 94]. L'information se propage de l'entrée à la sortie en suivant une seule direction et en traversant trois types de couches de neurones :

– couche d'entré : chaque neurone de cette couche ne possède qu'une seule entrée qui est l'entrée extérieure du réseau. Cette couche contient autant de neurones que d'entrées extérieures.

– couche de sortie : les sorties des neurones de cette couche sont les sorties du réseau. On trouve autant de neurones dans cette couche que le système admet de sorties.

– couches cachées : entre les deux couches d'entrée et de sortie, on peut trouver aucune, une ou plusieurs couches cachées. Les entrées des neurones d'une couche cachée sont les sorties des neurones de la couche précédente. Aucune contrainte ou indication sur le nombre de neurones dans ces couches n'existe, si ce n'est le nombre maximal de paramètres du réseau.

Enfin, chaque neurone du PMC reçoit les sorties de tous les neurones de la couche qui précède la sienne, et il est connecté à tous les neurones de la couche suivante. Les neurones d'une même couche ne présentent pas d'interconnexions.

Les différences entre la nouvelle architecture (NA) et le PMC sont les suivantes :

– Premièrement, la NA n'admet qu'une seule couche de neurone, alors qu'on trouve dans le PMC, outre les couches d'entrée et de sortie, plusieurs couches cachées.

– La NA admet plusieurs neurones d'entrée pour une variable donnée, alors qu'on ne trouve qu'un seul neurone d'entrée par variable dans le PMC.

– La NA ne peut admettre qu'une seule sortie par réseau, alors que pour le PMC on peut avoir plusieurs réponses ou plusieurs variables de sortie par réseau. Si on a besoin de plusieurs variables de sortie pour la NA, il faut relancer la NA autant de fois qu'il y a de variables de sortie (comme on le verra dans l'exemple qui suit).

– Dans la NA, on arrive à mieux comprendre et analyser le rôle de chaque neurone, ce qui conduit à pouvoir choisir le nombre de neurones nécessaires pour atteindre un certain degré de précision du réseau. Par contre, l'utilisation du PMC reste empirique et il est difficile de contrôler l'évolution des performances du réseau en changeant sa structure.

– Enfin, dans la NA, il n'est pas besoin de passer par une étape de codage et de décodage des informations à l'entrée et à la sortie du réseau, opérations qui sont

indispensables pour l'utilisation du PMC (il faut normaliser les entrées et les sorties entre 0 et 1).

Dans cette partie, une comparaison entre l'efficacité de la NA et celle du réseau PMC va être établie en utilisant une application mécanique tirée de [VAS 97a] et [VAS 97b], et dont les résultats nous ont été communiqués. Il s'agit de synthétiser les dimensions d'un mécanisme plan dont la fonction est de générer une trajectoire, et ceci à l'aide des deux réseaux de neurones. Seules les informations nécessaires pour poser le problème et préparer les vecteurs d'apprentissage, ainsi que les résultats obtenus sont présentés ici.

Dans les mécanismes à barres (treillis articulés), chaque point du mécanisme décrit une trajectoire qui est fonction de la morphologie du mécanisme et des dimensions des barres constituantes. Dans cet exemple, on s'intéresse au mécanisme 2D à barres de la figure 8, et en particulier à la trajectoire décrite par le point E (la liaison dite motrice étant en A).

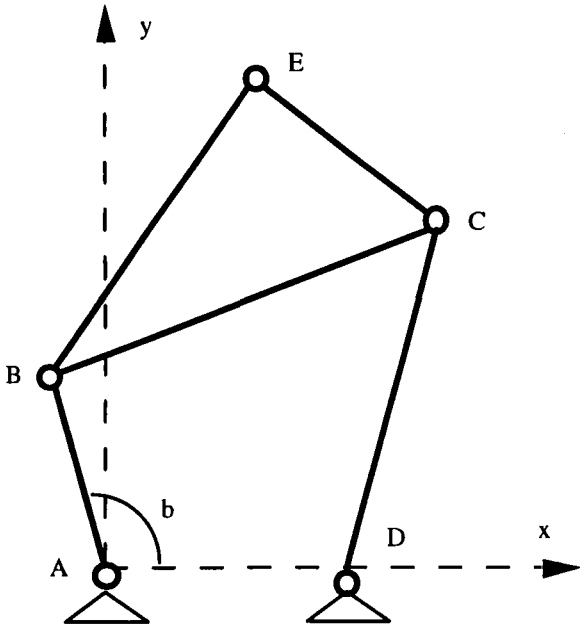

Figure 8. *Mécanisme quatre barres (les liaisons pivots A et D sont fixées : Xa = Ya =0, Xd = 100 et Yd =0).*

A chaque jeu de dimensions du mécanisme correspond une trajectoire pour le point E. Cette trajectoire peut être soit ouverte, soit fermée. Elle est déterminée par les lois de la cinématique. Il est donc possible de définir une application f reliant les formes de trajectoires aux dimensions des barres pour une morphologie donnée. La question posée est de savoir quelles sont les dimensions des barres pour une

trajectoire donnée, autrement dit, il s'agit d'inverser la fonction f. Cependant, f n'est pas toujours inversible. Une discussion sur l'existence de cette fonction inverse est développée dans [VAS 97a]. De plus, dans le cas étudié, des limitations sur les dimensions des différentes barres sont données pour que les trajectoires du point E ne soient que des trajectoires fermées. Bien que f ne soit pas inversible, l'objectif est de déterminer à l'aide des réseaux de neurones les dimensions du mécanisme qui permettent de générer une trajectoire la plus proche possible d'une trajectoire objectif.

Dans le cas de l'utilisation du réseau PMC par exemple (qui est, rappelons-le, un réseau à apprentissage supervisé et destiné à la généralisation), le calcul revient à établir une interpolation entre un certain nombre de vecteurs de cas. Plus précisément, la stratégie est la suivante : on construit un corpus de vecteurs d'apprentissage en choisissant aléatoirement un certain nombre de configurations du mécanisme et en calculant les trajectoires du point E correspondantes. Les vecteurs d'apprentissage contiennent comme variables d'entrée les paramètres définissant les trajectoires et comme variables de sortie les paramètres nécessaires pour monter le mécanisme correspondant. Une fois que le corpus de vecteurs d'apprentissage est prêt, on lance l'entraînement. Après apprentissage, le réseau de neurones doit établir une interpolation entre les données fournies en apprentissage pour une forme de trajectoire spécifique, et donne les dimensions du mécanisme qui permettent d'obtenir la trajectoire du point E la plus proche de la trajectoire désirée.

Pour notre exemple, la barre AB a une position initiale horizontale et orientée à gauche. Les liaisons pivot A et D attachées au mécanisme (figure 8) sont fixées en des positions conventionnelles : Xa = Ya = 0, Xd = 100 et Yd = 0). On a : xB < 0 et yB = 0. Par suite, les cinq coordonnées qui restent, à savoir xB, xC, yC, xE et yE déterminent la configuration initiale du mécanisme. Les variables de sortie d'un vecteur d'apprentissage sont tout simplement les valeurs du paramétrage cartésien absolu des coordonnées des liaisons dans la position initiale du mécanisme, ce qui assure l'unicité de définition (un seul assemblage possible) et l'homogénéité des variables (pas de confusion entre coordonnées et angles). Quant aux variables d'entrée, ce sont les informations qui représentent la forme des trajectoires. La méthode utilisée dans [VAS 97a] pour coder la forme d'une trajectoire donnée consiste à approximer la forme d'une trajectoire par les coefficients des cinq premières harmoniques issues de la décomposition d'une trajectoire en série de Fourier. Celle-ci se fait à partir de sa représentation sous forme d'une liste de points, obtenus par simulation cinématique. La position d'un point appartenant à une trajectoire peut être modélisée par une fonction complexe : $z(t) = x(t) + iy(t)$, avec x et y les coordonnées du point dans le plan et fonctions du paramètre curviligne t. Pour des trajectoires fermées, la fonction z(t) est périodique. Le développement en série de Fourier s'écrit alors : $z(t) = \Sigma a_m \exp(2\pi imt)$. Les coefficients complexes de Fourier sont donnés par la formule : $a_m = \bullet_0^1 \exp(-2\pi imt)z(t)dt$. La fonction z(t) s'obtient par simulation cinématique, mais la trajectoire obtenue est définie par une liste de points zk (k = 0,...,N) particuliers. Pour cela, les intégrales de l'équation précédente sont discrétisées en utilisant la formule des trapèzes. Des vérifications ont montré que le choix des 5 premières harmoniques semble un bon compromis entre la précision obtenue de l'approximation et le nombre de paramètres identifiant la trajectoire. Ainsi, la

définition de la trajectoire est déterminée par 11 paramètres a_m complexes (m = -5,...,5), donc par 22 paramètres réels. L'invariance du lien entre les formes des trajectoires et les mécanismes permet (grâce à une procédure de normalisation) d'obtenir une sorte de classe d'équivalence de forme, les coefficients de Fourier de la trajectoire normalisée étant représentatifs pour une telle classe. Le but de la normalisation est d'obtenir une représentation de la forme de la trajectoire, indépendamment des transformations affines qu'elle peut subir, ainsi que de s'abstraire du point de départ et de la direction de parcours. La normalisation a pour résultat une diminution de 5 du nombre de coefficients de Fourier. On passe ainsi de 22 coefficients à 17 coefficients normalisés. En fin de compte, un vecteur d'apprentissage est composé de 17 variables d'entrée (qui sont 17 coefficients normalisés de la représentation complexe de la trajectoire) et de 5 variables de sortie (qui sont les 5 paramètres nécessaires pour déterminer la configuration du mécanisme dans sa configuration initiale).

La complexité de la fonction recherchée impose l'utilisation d'un échantillon important de vecteurs d'apprentissage. Au total, 60.000 cas ont été générés par simulation pour le mécanisme quatre barres, le tirage des dimensions étant effectué d'une manière aléatoire. Les 60.000 vecteurs générés sont répartis en trois catégories :

– 30.000 comme données d'apprentissage pour le PMC ; ces données sont utilisées pour calculer la fonction erreur durant l'apprentissage.

– 15.000 comme jeu de données de validation ; ces données sont utilisées pour calculer pendant l'apprentissage l'évolution de la capacité de généralisation du réseau ; et 15.000 comme données de test ; ces données sont utilisées pour tester la capacité de généralisation sur des données non utilisées durant l'apprentissage.

Comme d'usage avec le réseau PMC, les entrées et les sorties des 60.000 vecteurs sont normalisées entre 0 et 1. Le PMC utilisé est le SNNS (Stuttgart Neural Network Simulator), réseau de neurones développé à l'université de Stuttgart. La structure du réseau de neurones est composée de 17 neurones d'entrée, correspondant aux coefficients de Fourier normalisés, de 2 couches cachées de 22 neurones chacune et de 5 neurones de sorties correspondant aux paramètres dimensionnels du mécanisme. Cette structure est en fait celle qui a donné les meilleurs résultats après plusieurs modifications du nombre de couches cachées et du nombre de neurones par couche cachée. L'apprentissage a été effectué en 14.000 itérations et la durée totale de calcul a été de 30 heures sur une station *Sun Sparc 10 model 30, Sunos 4.1.3*. On définit une erreur globale comme la moyenne sur un nombre de vecteurs considérés (les vecteurs d'apprentissage, ou les vecteurs de validation) de l'erreur quadratique produite sur les cinq sorties du réseau. C'est ainsi que l'erreur globale calculée en sortie du réseau pour les vecteurs d'apprentissage est de 5 %, tandis que l'erreur quadratique globale sur les vecteurs de validation est de 26,7 %.

On passe maintenant au calcul avec le nouveau réseau de neurones. Pour ce calcul, on n'a pu récupérer de la thèse citée que 30.000 vecteurs de cas, qui ont été répartis comme suit :

– 23.000 comme données d'apprentissage (utilisés pour entraîner le réseau).

– 7000 comme jeu de données de test (utilisés pour estimer la capacité du réseau à la généralisation une fois l'apprentissage terminé).

Cependant, le nouveau réseau de neurones (comme il a déjà été indiqué) n'admet qu'une seule sortie. Par conséquent, cinq réseaux ont été construits, chaque réseau étant dédié à l'approximation d'une des cinq variables à évaluer. On n'a effectué qu'une seule itération d'apprentissage en considérant que chaque variable ne possède qu'un seul neurone de spécialisation. Globalement, pour les cinq réseaux, la durée de l'apprentissage est de moins de 8 heures de calcul au total sur une machine *Silicon Graphics R10000*, de performances légérement inférieures à la *Sun Sparc* utilisée pour le PMC. De plus, l'erreur globale à la sortie du nouveau réseau de neurones (en utilisant la même définition que pour le PMC et en prenant la somme des erreurs obtenues sur les cinq sorties) est de 0,4 %, tandis que l'erreur quadratique moyenne sur les vecteurs de test est de 9,5 % (ce qui signifie que l'erreur de test est en fait de moins de 2 % en moyenne par variable de sortie). Bien qu'une comparaison représentative entre des réseaux de neurones d'architecture différente ne soit pas une chose aisée (c'est pour cela que ces comparaisons n'ont pas été systématisées par la suite), ces résultats montrent la supériorité évidente du nouveau réseau de neurones sur cet exemple difficile, tant sur le volume des calculs que sur la qualité des résultats obtenus. Pourtant le PMC utilisé (le SNNS) fait référence en la matière et on a utilisé un nombre de vecteurs d'apprentissage inférieur pour le nouveau réseau de neurones (23.000 au lieu de 30.000 pour le PMC), et un ordinateur de performances inférieures à celui utilisé dans la thèse [VAS 97a].

2.2 . *Les résultats obtenus*

2.2.1. *Un exemple analytique.*

Comme premier exemple, nous donnons un exemple de test purement analytique, avec la fonction à trois variables suivante :

$f = (1 - \mathrm{Log}^2(z))\,(0.18xz - 0.28x - 0.35z + 2.33)\,(\sin(x)e^{(0.2x-y-1)}) / (1 + e^{(0.2x-y-1)})^2$

Les plages de variation des variables x, y, et z sont :

x appartient à [0,3•]

y appartient à [-3,3]

z appartient à [0.4,2.7]

L'objectif dans ce test est d'établir une approximation de la fonction f à l'aide du nouveau réseau de neurones, et de tester cette approximation. On construit pour ce faire un corpus de vecteurs d'apprentissage formé de 1500 vecteurs choisis d'une façon aléatoire dans le domaine de définition des variables. Le calcul s'effectue sur une machine *Silicon Graphics R10000*. On prend 4 neurones de départ pour chaque variable. L'algorithme converge après 135 secondes de calcul. L'erreur quadratique de sortie est de 0,143. Le nombre de neurones requis est de 5, 3 et 4 pour les variables x, y et z respectivement.

Pour tester l'efficacité de cette approximation, on a construit ensuite un corpus de 500 vecteurs de test choisis aléatoirement dans le domaine de recherche. Dans tout ce qui suit, comme dans cet exemple, la convergence de l'apprentissage est jugée à partir de tests non utilisés pour l'apprentissage. Une erreur est définie pour chaque vecteur de test comme la valeur absolue de la différence de la valeur de la fonction exacte et de l'approximation donnée par le réseau, divisée par l'amplitude de la variation de la fonction dans le corpus des vecteurs de test. C'est ainsi que l'approximation de la valeur de la fonction est obtenue avec moins de 1 % d'erreur pour 279 vecteurs sur les 500 vecteurs de test ; on a une erreur comprise entre 1 % et 2 % pour 98 vecteurs, une erreur comprise entre 2 % et 3 % pour 66 vecteurs, une erreur comprise entre 3 % et 4 % pour 29 vecteurs et enfin une erreur supérieure à 4 % pour seulement 28 vecteurs. L'approximation est très satisfaisante.

2.2.2. *Un exemple d'approximation de calculs par éléments finis d'une plaque composite par le nouveau réseau de neurones (en statique et en dynamique)*

On considère la plaque composite à cinq couches représentée sur les figures 9 et 10. C'est une plaque rectangulaire de 120 x 80 cm^2, maintenue horizontalement et encastrée sur deux bords adjacents. Cette plaque admet une symétrie dans l'empilement des couches de façon à avoir un découplage entre les effets de membrane et les effets de flexion. Ainsi, la couche centrale admet comme plan neutre celui de la plaque, les deux couches extrêmes sont formées du même matériau et présentent la même épaisseur et la même orientation des axes d'orthotropie. Il en est de même pour les deux couches intermédiaires. Les variables de conception retenues pour notre modélisation sont les épaisseurs x1, x2 et x3 respectives des deux couches extrêmes, des deux couches intermédiaires et de la couche centrale, ainsi que les angles d'orientation de leurs axes d'orthotropie b1, b2 et b3. On suppose de plus que la couche centrale est constituée d'un matériau orthotrope dont les paramètres mécaniques sont les suivants : Ex = 2,11E4 Mpa, Ey = 0,53E4 Mpa, Gxy = Gxz 0,26E4 Mpa, Gyz = 0,13E4 Mpa, Nuxy = 0,25, densité = 1,524E3 kg/m^3. Les quatre autres couches sont formées du même matériau orthotrope dont les caractéristiques sont les suivantes : Ex = 5,5E4 Mpa, Ey = 1,83E4 Mpa, Gxy = Gxz 0,91E4 Mpa, Gyz = 0,455E4 Mpa, Nuxy = 0,25, densité = 1,993E3 kg/m^3.

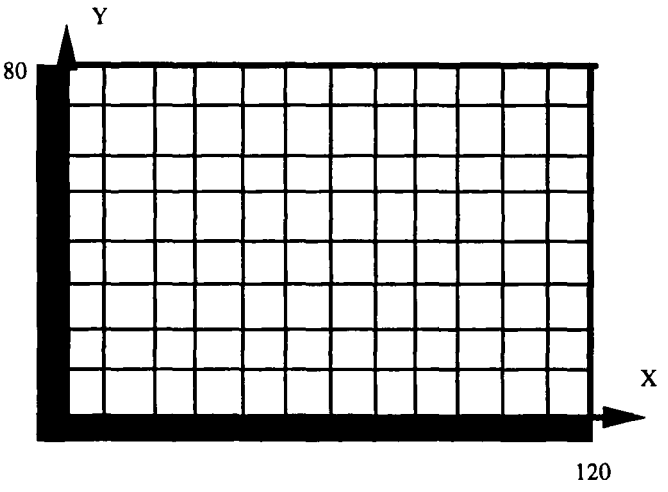

Figure 9. *Maillage de la plaque composite.*

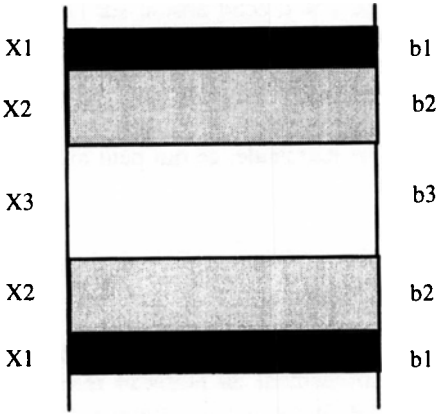

Figure 10. *Coupe transversale de la plaque composite.*

Les limites de variation des six variables de conception sont les suivantes :

$1 < x1 \text{ (cm)} < 2$

$2 < x2 \text{ (cm)} < 4$

$3 < x3 \text{ (cm)} < 6$

Les angles b1, b2 et b3 peuvent varier entre -45° et 135°.

2.2.2.1. Approximation en statique des calculs par éléments finis de la plaque composite par le nouveau réseau de neurones

Dans le cas statique, la plaque composite est soumise à l'action conjuguée de son poids propre et d'une charge surfacique répartie de 2 Mpa perpendiculaire au plan moyen de la plaque. Les calculs par éléments finis sont effectués par le code *Ansys* sur station de travail *Silicon Graphics R10000*. L'objectif du calcul est l'évaluation de la flèche maximale de la plaque (plus grand déplacement des nœuds de la plaque dans la direction de la charge appliquée). Pour l'entraînement du nouveau réseau de neurones, 2000 vecteurs d'apprentissage sont choisis aléatoirement dans l'espace de définition des six variables de conception. Ce nombre de 2000 ne peut être justifié et défini à l'avance car il résulte d'une expérimentation numérique conduisant à un bon compromis entre la qualité et la quantité des calculs. L'apprentissage est lancé avec 4 neurones au départ pour chaque variable et une erreur maximale admissible de 0,05. L'algorithme d'apprentissage converge après 4958 secondes en effectuant 19 itérations de reconfiguration. Ensuite 2000 autres vecteurs de test sont choisis aléatoirement dans le domaine de recherche pour pouvoir comparer les calculs par *Ansys* aux calculs par le nouveau réseau de neurones. Les calculs de comparaison donnent les résultats suivants : 55 % des solutions présentent moins de 1 % d'écart absolu sur la flèche maximale entre elles, 29,7 % des solutions présentent un écart compris entre 1 % et 2 % sur la flèche calculée d'une part par *Ansys* et d'autre part par le nouveau réseau de neurones, et seules 2 % des solutions présentent un écart supérieur à 4 % mais inférieur à 6 % (il n'y a pas de solutions présentant plus de 6 % d'écart). Le réseau donne donc une bonne approximation de la flèche maximale, ce qui peut autoriser son utilisation dans une procédure d'optimisation.

2.2.2.2. Approximation en dynamique des calculs par éléments finis de la plaque composite par le nouveau réseau de neurones

Dans le cas dynamique, l'objectif du calcul est l'évaluation de la première fréquence de résonance de la plaque. Pour l'entraînement du nouveau réseau de neurones, 2000 vecteurs d'apprentissage sont aussi choisis aléatoirement dans l'espace de définition des six variables de conception. L'apprentissage est lancé avec 4 neurones au départ pour chaque variable et une erreur maximale admissible de 0,02. L'algorithme d'apprentissage converge après 3720 secondes en effectuant 7 itérations de reconfiguration. Ensuite 2000 autres vecteurs de test sont choisis aléatoirement dans le domaine de recherche pour pouvoir comparer le calcul dynamique réalisé par *Ansys* aux calculs par le nouveau réseau de neurones. Les calculs de comparaison donnent les résultats suivants : 72,8 % des solutions présentent moins de 1 % d'écart absolu sur la fréquence, 23,4 % des solutions présentent un écart compris entre 1 % et 2 % sur cette première fréquence calculée d'une part par *Ansys* et d'autre part par le nouveau réseau de neurones, et seules 0,1 % des solutions présentent un écart supérieur à 4 % mais inférieur à 5 % (il n'y a pas de solutions présentant plus de 5 % d'écart). Le réseau donne donc une bonne approximation de la fréquence fondamentale.

3. Quelques applications en optimisation.

3.1. *Introduction*

Après une expérience récente en optimisation des structures et des systèmes mécaniques par la technique des algorithmes génétiques [MAR 95a], [MAR 95b], [MAR 95c], [MAR 96a], [MAR 96c] et [KAL 97], on s'est intéressé aux possibilités offertes par les réseaux de neurones. Dans [MAR 96b], [MAR 98] et [MAR 99], on a montré comment on pouvait remplacer graduellement au cours d'un processus de type algorithme génétique les analyses par éléments finis (ou autres logiciels de calcul) par des calculs approchés à base de réseaux de neurones, et ceci dans l'optique de calculs plus rapides.

3.2. *L'optimisation via la nouvelle architecture de réseau de neurones*

On a vu dans la première partie de ce travail que le nouveau réseau de neurones peut avoir des applications variées dans différents domaines : approximation de fonction, reconnaissance de paramètres, recalage de modèles etc... Néanmoins l'objectif principal en construisant ce nouveau réseau de neurones est de l'appliquer à l'optimisation des structures mécaniques. Comme on peut s'affranchir de l'utilisation d'une technique de type algorithme génétique, la nouvelle stratégie d'optimisation consiste dans un premier temps à établir une approximation de la fonction objectif du problème d'optimisation et des fonctions contraintes (limitations technologiques), et dans un deuxième temps à optimiser le problème via la réponse du réseau seul, comme on va l'expliquer dans ce qui suit.

Cela est rendu possible grâce aux caractéristiques de la fonction réponse du réseau. En effet, une fois le réseau de neurone entraîné, la réponse donnée par le réseau f(X) est une fonction continue et infiniment dérivable en X. Toute méthode d'optimisation analytique peut donc être utilisée pour trouver les extremums de la fonction f. Le problème majeur de ces méthodes analytiques est que l'optimum trouvé dépend du point de départ si la fonction et le domaine de recherche ne sont pas convexes. Le risque est alors de tomber sur des optimums locaux.

La stratégie consiste à reconstruire la fonction objective tranche par tranche, chaque tranche étant formée d'une fonction monotone, ce qui indique si la fonction approchée tend à augmenter ou à diminuer dans un domaine précis. En lançant ainsi la procédure d'optimisation sur chaque tranche, on est alors sûr de trouver tous les optimums du problème.

La dérivation de la fonction obtenue par le nouveau réseau de neurones étant extrêmement rapide, on lance le calcul d'optimisation par une méthode analytique de gradient classique et ceci à partir de chaque seuil amont du réseau (il y a m seuils au total pour toutes les variables) et à partir des limites extrêmes du domaine de recherche (au nombre de p). Ainsi, la procédure d'optimisation est lancée m+p fois et les m+p points optimums obtenus sont analysés pour détecter ceux qui sont dans le même voisinage et qui désignent par la suite le même point optimal. De cette façon, on peut détecter tous les points optimums de la fonction à approcher. Bien

évidemment, les points optimums obtenus sont trouvés à partir de la réponse du réseau qui est une réponse approximative de la fonction réelle. Les résultats obtenus dépendent donc de la qualité de l'approximation utilisée. Pour cela une phase de vérification et d'ajustement s'avère nécessaire après avoir localisé les extremums à l'aide du réseau de neurones.

3.3. *Quelques applications en optimisation des structures mécaniques*

3.3.1. *Un exemple analytique.*

Comme premier exemple, on reprend l'exemple analytique déjà considéré dans la partie 2.2.1 où une approximation de la fonction f à l'aide du nouveau réseau de neurones a déjà été construite. On veut maintenant déterminer les maximums de la fonction f et les comparer aux valeurs exactes. D'après ce qui a été expliqué précédemment, on reconstruit la fonction f pièce par pièce et on lance une procédure d'optimisation analytique à partir des points X obtenus par combinaison des valeurs des seuils des neurones et des deux limites extrêmes de définition de chaque variable. En général, on lance la procédure d'optimisation un nombre Nopt de fois égal au produit de i=1 jusqu'à n des (m_i+2) avec n le nombre de variables et mi le nombre de neurones de la variable i. Les différents points de départ ne conduisant pas forcément à des points optimaux différents, il faut inspecter à la fin du calcul les points obtenus qui se trouvent dans le même voisinage et qui désignent par conséquent le même point optimal. Dans le cas présent, le calcul d'optimisation a été lancé 210 fois. On donne ci-après les deux maximums exacts de la fonction f et ceux donnés par le nouveau réseau de neurones :

maximum exact n°1 : x=1,48, y=-0,70, z=0,84 et f =0,344

maximum exact n°2 : x=7,83, y= 0.57, z=1.45 et f =0,204

maximum approché n°1 : x=1,52, y=-0,65, z=0,90 et f =0,331

maximum approché n°2 : x=7,85, y= 0,38, z=1,22 et f =0,188

Il s'avère dans ces tests que l'erreur de sortie maximale se produit dans le voisinage des points optimums. Cela s'explique par le fait que les points optimums sont les points les plus éloignés de la moyenne des valeurs objectives. Par suite, l'apprentissage dans le voisinage de ces points est le moins précis. Cependant, il est plus important de localiser ces points avec précision plutôt que d'obtenir la valeur exacte des points optimums (puisqu'elle peut être calculée *a posteriori* d'une manière exacte).

3.3.2. *Cas de la plaque composite*

On reprend comme exemple d'optimisation le cas de la plaque composite à 5 couches décrit dans la partie 2.2.2. Les variables de conception pour l'optimisation sont au nombre de 6 : les épaisseurs x1, x2 et x3, et les angles d'orientation des axes d'orthotropie b1, b2 et b3. L'objectif est de maximiser la première fréquence de vibration de la plaque, les variables de conception étant limitées comme suit :

$1 < x1\ (cm) < 2$

$2 < x2\ (cm) < 4$

$3 < x3\ (cm) < 6$

Les angles b1, b2 et b3 peuvent varier entre -45° et 135°.

Pour l'optimisation, on utilise les résultats de l'apprentissage donnés dans la partie 2.2.2. La fonction approximative obtenue pour la première fréquence est une fonction dérivable et explicite des 6 variables de conception. L'optimisation, en utilisant cette fonction, ne pose aucun problème et montre qu'il existe 11 optimums locaux qui correspondent tous à des combinaisons d'épaisseurs maximales dans les domaines de variation des variable x1, x2 et x3 et conduisent à des fréquences maximales toutes voisines de 16 Hz (la plaque ayant alors une épaisseur maximale, ce qui est logique).

3.3.3. *Cas d'un treillis articulé*

On considère le treillis articulé de la figure 11. Celui-ci est constitué par l'assemblage de 10 composants identiques, chaque composant est constitué de 10 barres. Ce treillis articulé comporte 32 nœuds et repose horizontalement sur des appuis simples situés aux nœuds 1, 11, 12 et 22. L'ensemble de la structure est constitué de 100 éléments. Tous les éléments ont une longueur de 30 cm sauf les éléments diagonaux de la partie horizontale inférieure qui ont une longueur de 42,43 cm. Les barres de la structure sont en acier de module d'Young 0,21 N/mm^2, de coefficient de Poisson de 0,3 et de densité 7800 kg/m^3. La section des éléments longitudinaux de la partie inférieure de la structure est de 1 cm^2. La section des éléments longitudinaux de la partie supérieure liant les 10 composants de la structure est de 3 cm^2. La section de toutes les autres barres est de 0,5 cm^2. La structure est chargée par deux forces concentrées verticales et identiques d'intensité 20kN appliquées au nœud 6 et au nœud central. La structure admet donc un plan de symétrie vertical.

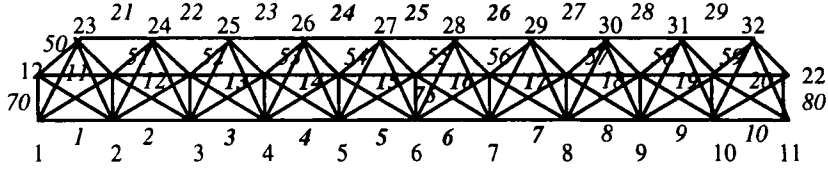

Figure 11. *Le treillis articulé (en italique, le numéro des éléments)*

On trouve ci-après le tableau des connectivités des éléments finis :

élément	nœuds	élément	nœuds	élément	nœuds	élément	nœuds
1	1,2	*26*	28,29	*51*	13,24	*76*	7,18
2	2,3	*27*	29,30	*52*	14,25	*77*	8,19
3	3,4	*28*	30,31	*53*	15,26	*78*	9,20
4	4,5	*29*	31,32	*54*	16,27	*79*	10,21
5	5,6	*30*	1,23	*55*	17,28	*80*	11,22
6	6,7	*31*	2,24	*56*	18,29	*81*	1,13
7	7,8	*32*	3,25	*57*	19,30	*82*	2,14
8	8,9	*33*	4,26	*58*	20,31	*83*	3,15
9	9,10	*34*	5,27	*59*	21,32	*84*	4,16
10	10,11	*35*	6,28	*60*	13,23	*85*	5,17
11	12,13	*36*	7,29	*61*	14,24	*86*	6,18
12	13,14	*37*	8,30	*62*	15,25	*87*	7,19
13	14,15	*38*	9,31	*63*	16,26	*88*	8,20
14	15,16	*39*	10,32	*64*	17,27	*89*	9,21
15	16,17	*40*	2,23	*65*	18,28	*90*	10,22
16	17,18	*41*	3,24	*66*	19,29	*91*	2,12
17	18,19	*42*	4,25	*67*	20,30	*92*	3,13
18	19,20	*43*	5,26	*68*	21,31	*93*	4,14
19	20,21	*44*	6,27	*69*	22,32	*94*	5,15
20	21,22	*45*	7,28	*70*	1,12	*95*	6,16
21	23,24	*46*	8,29	*71*	2,13	*96*	7,17
22	24,25	*47*	9,30	*72*	3,14	*97*	8,18
23	25,26	*48*	10,31	*73*	4,15	*98*	9,19
24	26,27	*49*	11,32	*74*	5,16	*99*	10,20
25	27,28	*50*	12,23	*75*	6,17	*100*	11,21

L'objectif de l'optimisation est de minimiser la flèche maximale de la structure (obtenue au nœud 6) en rigidifiant ses éléments. Le renforcement de la structure s'effectue par une augmentation de la section des barres qui est réalisée en augmentant la masse des barres d'une valeur constante et définie au préalable. Pratiquement, cela peut être obtenu dans cet exemple en majorant la section de toutes les barres de 0,5 cm^2, sauf pour les éléments diagonaux de la partie horizontale inférieure de la structure qui sont plus longs et qui pourront subir par conséquent une majoration de 0,354 cm^2. Comme le poids de la structure est très faible par rapport au chargement appliqué, il est évident que minimiser la flèche de la structure va conduire à rigidifier tous les membres de la structure. Le problème est donc posé comme la recherche d'un choix optimal d'éléments à rigidifier suivant la procédure décrite ci-dessus avec un nombre total maximum d'éléments à rigidifier fixé au préalable.

Pour mettre en œuvre la stratégie d'optimisation neuronale, on effectue dans un premier temps l'apprentissage du calcul de la flèche maximale de la structure en fonction de la configuration de rigidité de la structure. Une fois l'apprentissage terminé, les paramètres du réseau sont analysés pour trouver la configuration optimale. Les variables de conception sont des variables binaires et sont au nombre de 100 (il y a 100 barres au total). Elles représentent l'état de rigidité des différentes barres. Le digit 0 représente l'état par défaut qui désigne une configuration non

rigidifiée de la barre correspondante. Le digit 1 traduit un état rigidifié de la barre correspondante ; la section de cette dernière est alors augmentée, selon le cas, d'une valeur parmi celles citées précédemment. Environ 1500 vecteurs d'apprentissage et 2000 vecteurs de test sont utilisés et calculés au préalable par la méthode des éléments finis (*Ansys*). L'apprentissage conduit à des résultats très satisfaisants puisque 63,7 % des vecteurs de test ont une erreur inférieure à 1 %, 31,1 % des vecteurs de test ont une erreur comprise entre 1 et 2 % et aucun des vecteurs de test n'a une erreur supérieure à 4 %. A partir du réseau ainsi entraîné, une méthode énumérative simple sur les paramètres du réseau permet de localiser le minimum global du problème posé. De plus, la recherche optimale peut être menée en contraignant la solution optimale à respecter certaines valeurs attribuées aux variables par le concepteur. Par exemple, l'utilisateur peut lancer le processus d'optimisation dans le sous-ensemble formé par les solutions dont certains membres peuvent être rigidifiés alors que d'autres ne le peuvent pas. Notons que la recherche n'est pas contrainte à respecter la symétrie de la structure.

A titre indicatif, si on s'intéresse à la détermination du choix optimal de 9 éléments à rigidifier, l'analyse des coefficients du réseau donne les éléments 4, 5, 6, 7, 14, 15, 16, 17 et 25 comme choix optimal d'éléments à rigidifier. La réponse du réseau correspondante est de 2,196 cm alors que la valeur obtenue par éléments finis est de 2,192 cm. Toujours dans ce cas, on exige que les barres 4 et 14 ne soient pas rigidifiées, elles sont alors remplacées par les barres 24 et 26. La nouvelle solution admet comme valeur approximative de la flèche maximale 2,204 cm alors que la valeur calculée par éléments finis est de 2,198 cm. Si maintenant on impose que les barres 3, 4, 13 et 14 soient rigidifiées alors que les barres 5, 6, 15 et 16 ne peuvent pas être rigidifiées, la solution optimale contient les éléments 3, 4, 7, 13, 14, 17, 24, 25, et 75. Elle admet une valeur approximative, d'après la réponse du réseau, de 2,289 cm, alors que la valeur calculée par éléments finis est de 2,294 cm. L'optimalité de ces solutions est vérifiée en lançant plusieurs tests de vérification par éléments finis, obtenus par changement d'une ou de plusieurs valeurs de la solution. On constate que le nouveau réseau de neurones est utilisé dans cet exemple comme un outil d'aide à la conception.

4. Conclusion

La stratégie présentée dans cet article pour optimiser les structures mécaniques consiste à remplacer les fonctions objectifs et les fonctions contraintes d'un problème d'optimisation (quand elles sont calculées par éléments finis) par la réponse approximative d'un réseau de neurones. On a ainsi présenté une nouvelle architecture de réseau de neurones dont l'avantage essentiel est la détermination automatique du nombre de neurones du réseau pour que l'approximation atteigne une précision fixée par l'utilisateur. On a ensuite appliqué la nouvelle stratégie neuronale à plusieurs exemples d'optimisation. Remarquons que les exemples traités n'ont pas nécessité une mise en œuvre particulière. L'originalité de cette stratégie est qu'elle contourne les outils traditionnels d'optimisation, la résolution et la manipulation d'équations matricielles ou différentielles. Seule la valeur objective est nécessaire pour entamer les procédures d'optimisation. On peut insister sur le

fait que le réseau de neurones est utilisé pour établir une approximation de la fonction objective ; par suite le réseau de neurones a une vision générale de la fonction objective sur tout le domaine de validité, c'est pour cela que l'on peut parler de stratégie d'optimisation globale. Cette technique permet de faire des économies de temps de calcul substantielles par rapport aux techniques classiques. De plus, ce nouveau réseau de neurones est suffisamment général et peut être utilisé dans bien d'autres applications et domaines que l'optimisation des structures mécaniques : identification de paramètres, recalage de modèles, traitement de l'information et des signaux, conduite de processus, pour n'en citer que quelques-uns.

Bibliographie

[BER 92] BERKE L., HAJELA P., « Applications of artificial neural nets in structural mechanics », *Structural Optimization*, Vol . 4, 1992, p. 90-98.

[CAO 98] CAO X., SUGIYAMA Y., MITSUI Y., « Application of artificial neural networks to load identification », *Computers and Structures*, Vol.. 69, 1998, p. 63-78.

[FUR 98] FURUKAWA T., YAGAWA G., « Implicit constitutive modelling for viscoplasticity using neural networks », *International Journal for Numerical Methods in Eng.*, Vol . 43, 1998, p. 195-219.

[GHA 98] GHABOUSSI J., PECKNOLD D., ZHANG M., HAJALI, R., « Autoprogressive training of neural network constitutive models », *International Journal for Numerical Methods in Eng.* , Vol. 42, 1998, p. 105-126.

[HAJ 94] HAJELA P., SZEWCZYK Z., « Neurocomputing strategies in structural design- on analysing weigths of feedforward neural networks », *Structural Optimization,* Vol. 8, 1994, p. 236-241.

[HEN 98] HENCHI K., FAFARD M., TALBOT M., LANGIS D., « L'application des réseaux neuronaux artificiels pour l'identification et la détection de l'endommagement dans les ponts », *Revue Européenne des Elements Finis,* Vol. 7, n° 1-2-3, 1998, p. 257-272.

[JOD 94] JODOUIN J.F. , *Les réseaux neuromimétiques* , Editions Hermès, Paris, 1994.

[KAL 97] KALLASSY A., MARCELIN J.L., « Optimization of stiffened plates by genetic search », *Structural Optimization,* Vol. 13, 1997, p. 134-141.

[KAL 98] KALLASSY A., « Intégration des réseaux de neurones dans la conception et l'optimisation des structures mécaniques », Thèse de Doctorat, Université Joseph Fourier Grenoble I.

[MAR 95a] MARCELIN J.L., « CAO d'engrenages par algorithmes génétiques », *Revue Internationale de CFAO et d'Informatique Graphique,* Vol. 10, n°5, 1995, p. 485-498.

[MAR 95b] MARCELIN J.L., « Optimisation des conditions aux frontières par algorithmes génétiques », *Revue Européenne des Elements Finis,* Vol. 4, n° 3, 1995, p. 361-373.

[MAR 95c] MARCELIN J.L., SHAKHESI S., POURROY F., « Optimal constrained layer damping of beams : experimental and numerical studies », *Shock and Vibration,* Vol. 2, n° 6, 1995, p. 445-450.

[MAR 96a] MARCELIN J.L., « Optimisation stochastique de mécanismes », *Revue Internationale de CFAO et d'Informatique Graphique,* Vol. 11, n°6, 1996. p. 635-651.

[MAR 96b] MARCELIN J.L., KALLASSY A., « Approximation de calculs éléments finis par réseaux neuronaux : quelques applications en optimisation », *Revue Européenne des Elements Finis,* Vol. 5, n° 4, 1996, p. 443-460.

[MAR 96c] MARCELIN J.L., KALLASSY A., « Vers une optimisation intégrée des structures ou des systèmes mécaniques », *Revue Internationale de CFAO et d'Informatique Graphique,* Vol. 11, n°3, 1996, p. 289-306.

[MAR 98] MARCELIN J.L., « Optimisation intégrée de mécanismes à l'aide de réseaux neuromimétiques et de méthodes d'optimisation évolutionnaires », *Revue Internationale de CFAO et d'Informatique Graphique,* Vol. 13, n°3, 1998, p. 265-281.

[MAR 99] MARCELIN J.L., « Evolutionary optimization of mechanical structures », *Engineering Optimization,* Vol. 31, n°5, 1999.

[SZE 93] SZEWCZYK Z., HAJELA P., « Neural network approximations in a simulated annealing based optimal structural design », *Structural Optimization,* Vol. 5, 1993, p. 159-165.

[SZE 94] SZEWCZYK Z., HAJELA P., « Neurocomputing strategies in structural design-decomposition based optimization », *Structural Optimization,* Vol.. 8, 1994, p. 242-250.

[VAS 97a] VASILIU A., « Une approche CAO pour la préconception des mécanismes plans générateurs de trajectoire », Thèse de Doctorat, Ecole Centrale de Paris.

[VAS 97b] VASILIU A., YANNOU, B., « Synthèse dimensionnelle de mécanismes générateurs de trajectoires par réseaux de neurones », *Actes du 5e colloque PRIMECA*, La Plagne, avril 1997, pp. 315-323.