N° d'ordre : /2014/DM

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

MEMOIRE

Présenté

AU DEPARTEMENT DE MECANIQUE
FACULTE DES SCIENCES DE L'INGENIEUR
UNIVERSITE DE BATNA

Pour L'obtention du diplôme de

MAGISTERE EN GENIE MECANIQUE

Option : Construction Mécanique

Par

CHEFRI BELKACEM

ETUDE DU COMPORTEMENT NON LINEAIRE ET DE L'ENDOMMAGEMENT SOUS SOLLICITATION THERMO MECANIQUES DES STRUCTURES MECANIQUES

Soutenue le : 20 /02 / 2014

Encadré par : Pr. Zidani Kamel

Devant Le jury :

Dr. Manaa Rabeh	Prof	Université de Batna	Président
Dr. Zidani Kamel	Prof	Université de Batna	Rapporteur
Dr. Brioua Mourad	Prof	Université de Batna	Examinateur
Dr. Cherfia A/ El-Hakim	Prof	Université de Constantine	Examinateur

Année Universitaire 2013 / 2014

REMERCIEMENTS

Je tiens à remercier tout d'abord, mon encadreur Mr. k.Zidani, professeur à l'université de Batna d'avoir accepté de diriger ce travail, mes remerciements vont également à Mr kaddouri wahid, qui m'a encouragé pour réaliser ce travail.

Je remercie Mr. Manaa Rabeh, professeur à la faculté de mécanique, Pour m'avoir fait l'honneur de présider mon jury.

Toute ma reconnaissance va également aux membres de jury :
MR. Brioua Mourad, Professeur à l'université de Batna.
MR. Cherfia A/El-Hakim, Professeur à l'université de Constantine.

Mes plus vifs remerciements à toute ma famille, et à tous ceux, qui n'ont pas cessé de m'encourager, et me soutenir constamment, durant toute la durée de l'élaboration de ce travail.

SOMMAIRE

CHAPITRE II : MODELE DE LA RETRO PROPAGATION D'ERREUR

CHAPITRE III : ANALYSE EXPERIMENTALE

Nomenclatures

$\mathbf{X_i}$: Entrée du neurone i.

$\mathbf{W_{ij}}$: Poids pondéré entre le neurone i et le neurone j .

$\sum$: Somme.

f ( ) : Fonction de transfert.

$\boldsymbol{\eta}$: Pas d'apprentissage.

$\mathbf{d_i}$: Sortie désirée du neurone formel.

$\mathbf{e_j}$: Entrée du neurone formel.

$\mathbf{X_E}$: Valeur brute.

$\mathbf{X_N}$: Valeur normalisée.

$\mathbf{V\_max}$: Valeur max sur un intervalle.

$\mathbf{V\_min}$: Valeur min sur un intervalle.

$\mathbf{b_1}$; $\mathbf{b_2}$: Biais.

$\mathbf{i_j}$: Entrée d'un neurone j.

$\mathbf{E_p}$: Erreur à la sortie du réseau pour un élément p .

$\mathbf{O_j}$: Sortie calculé d'un neurone j.

$\mathbf{T_\ell}$: Sortie désirée d'un élément de la couche ℓ.

$\boldsymbol{\delta_\ell}$: Signal d'erreur de l'élément ℓ.

$\boldsymbol{\alpha}$: Coefficient de correction « Momentum ».

$\boldsymbol{\sigma}$: Contrainte de Von mises.

F : Change statique.

Liste des tableaux et des figures :

INTRODUCTION

INTRODUCTION GÉNÉRALE

L'étude du comportement non linéaire de l'endommagement sous sollicitation thermomécanique dans les structures mécaniques revient à déterminer les paramètres mécaniques fondamentaux de ce comportement.

Le module de Young qui est l'un de ces paramètres est considéré comme étant une caractéristique essentielle permettant d'apprécier de façon représentative le niveau d'endommagement, la connaissance de sa variation en fonction de la température présente un intérêt majeur car elle permet de suivre l'état de l'endommagement dans ces structures.

Les modèles mathématiques établies pour traiter ce genre de problèmes mènent à des formulations compliqués et leur résolution par les méthodes classiques prennent beaucoup de temps et même dans des cas aboutissent à des résultats incohérents.

Une méthode dite intelligente a attiré l'intention de beaucoup de chercheurs ces derniers temps par sa simplicité et sa rapidité et qui repose sur la détermination de ces paramètres à partir des mesures expérimentales, par exemple on peut déterminer les modules élastiques à divers endroits d'une structure mécanique à partir des fréquences naturelles ou des déflections mesurées à un endroit donné.

Autrement dit exploiter les informations que renferment ces données, provenant des expériences pour résoudre ces problèmes et éviter, le recours aux formulations complexes et couteuses.

Cette méthode nommée, la méthode des réseaux de neurones, comme son nom l'indique est une modélisation du fonctionnement des neurones du cerveau humain et lui ressemble par la capacité de mémoriser la connaissance par l'exemple.

Son avantage, c'est qu'elle n'a pas besoin de formulation mathématique pour établir

la relation existante entre les différentes variables physiques d'un problème donné. Mais elle se sert seulement des mesures pour identifier ces grandeurs, après une phase dite d'apprentissage. Dans la pratique, on n'utilise pas les réseaux de neurones pour réaliser des approximations de fonctions connues.

Le plus souvent, le problème qui se pose à l'ingénieur est le suivant : il dispose d'un ensemble de mesures des variables d'un Processus de nature quelconque (physique, chimique, biologique, économique ,Financier...), ainsi que des mesures du résultat de ce processus ; il suppose qu'il existe une relation déterministe entre ces variables et ce résultat, et il cherche une forme mathématique de cette relation, valable dans le domaine où les mesures ont été effectuées; celles-ci sont en nombre fini, elles sont certainement entachées de bruit, et toutes les variables qui déterminent le résultat du processus ne sont pas forcément mesurées. En d'autres termes, l'ingénieur cherche à élaborer un « modèle » du processus qu'il étudie, à partir des mesures dont il dispose.

Donc, un réseau de neurones est un modèle de comportement se forme à partir d'un certain nombre d'exemples de ce comportement. Le modèle se construit par apprentissage, comme un chien apprenant à déceler des explosifs ou un enfant apprenant à reconnaître les lettres de l'alphabet.

Le réseau de neurones, "ignorant" au départ, est mis au point à partir des exemples et devient un modèle rendant compte du comportement observé.

Les réseaux de neurones sont exploités dans divers domaines telles qu'en traitement du signal, la reconnaissance de forme, dans la conception des robots autonomes et l'évaluation des ponts.

Leur application dans le domaine de l'ingénierie, et plus particulièrement les structures mécaniques a occupée ces dernières années une grande place.

En effet, l'identification, la détection et la localisation de l'endommagement, l'évaluation, la prédiction, l'optimisation, l'analyse et la conception, constituent les

principaux domaines, où cette méthode a prouvé son efficacité et sa rapidité.

Beaucoup de recherches ont été menées dans ce contexte, on cite, Les travaux de Wu et de al **[20]**, qui ont utilisé un réseau de neurones à rétro propagation. Notamment, Le Perceptron multicouches. Avec une et deux couches intermédiaires pour la détermination du **niveau d'endommagement** des membrures d'un bâtiment à trois étages.

Ils ont effectué une série d'analyse dynamique du bâtiment soumis à une sollicitation sismique pour différents niveaux d'endommagement de la structure.

La réponse de la structure (accélération horizontale), étant mesurée au niveau du troisième étage. Ils ont utilisé 200 mesures prises sur le spectre de la réponse de la structure comme données d'entrée du réseau ; les sorties étant le niveau d'endommagement à chaque Palier de la structure sur une échelle de O à 1: Une valeur de 1 représentant l'état non Endommagé. Et une valeur 0 endommagé.

Le réseau était constitué de 200 nœuds à la couche d'entrée.10 nœuds dans la couche intermédiaire et 3 nœuds dans la couche de sortie (architecture 200-(10)-3).L'apprentissage du réseau comportait 43 exemples.

Hajela et Berke [6], ont étudié les possibilités, qu'apportent les réseaux de neurones dans les **problèmes d'optimisation**.

Le problème traité consistait à déterminer les dimensions des membrures d'un treillis soumis à une charge concentrée statique de façon à minimiser les déformations dues à cette charge.

Les données d'entrée du réseau étaient les sections des cinq membrures du treillis, les quatre données de sortie étaient les déplacements horizontaux et verticaux en deux points du treillis. Ils ont utilisé un réseau avec couches intermédiaires de même qu'un réseau où certaines connexions étaient privilégiées. L'apprentissage, pour les deux réseaux. Comportait 50 exemples.

Ils arrivent à la conclusion, que le réseau avec lien fonctionnel des données (connexions privilégiées) donne de meilleurs résultats que le réseau avec couches Intermédiaires. Ils notent aussi que les résultats se détériorent, en présence de parasites de mesures et de données.

Barai et Pandey, **[2]** de leur coté, ont publié les résultats de leurs recherches sur la **détection de l'endommagement**, dans les membrures d'un pont en treillis à l'aide des réseaux de neurones. Ils simulent la réponse dynamique d'un pont sollicité par une charge mobile, ponctuelle progressant à vitesse constante.

Les données d'entrée du réseau de neurones, consistaient en 69 points de la réponse temporelle en déplacement vertical auxquels, s'ajoutaient des nœuds de contrôle déterminant le point de mesure utilisé. Les points de mesure correspondaient aux cinq jonctions entre le tablier du pont et les membrures du treillis.

Trois cas distincts ont été étudiés. Dans chacun des cas, des réseaux à deux couches intermédiaires de 21 nœuds chacune ont été utilisés, la couche de sortie étant constituée de 21 nœuds correspondant aux sections des 21 membrures du treillis (la variation de la section des membrures étant utilisée pour représenter l'endommagement).

Dans le premier cas. Un seul point de mesure a été utilisé, le réseau ayant alors 69 données d'entrée pour une architecture 69-(21)-(21)-21. Pour le deuxième et le troisième cas. 3 et 5 points de mesures ont été utilisés.

Aux 69 données d'entrée s'adjoignaient alors 3 et 5 nœuds, dont la valeur associée déterminait à quel point de mesure les 69 autres données d'entrée étaient associées. L'architecture des réseaux était alors 72-(21)-(21)-21 pour 3 points de mesures et 74-(21)-(21)-21 pour 5 points de mesures.

L'apprentissage des réseaux comportait 16 exemples d'apprentissage et 5 exemples de test Pour chaque point de mesures.

Les résultats obtenus de ces analyses sont excellents pour les trois cas. Les auteurs arrivent à la conclusion qu'un grand nombre de points de mesures et d'exemples d'apprentissage ne donne pas nécessairement de meilleurs résultats qu'un petit nombre. La conclusion ne fait Cependant pas état de l'influence des données binaires utilisées pour les deuxième et troisième Cas.

Notre objectif est d'exploiter cette technique (méthode des réseaux de neurones), pour prédire les paramètres physiques caractérisant l'endommagement ici le module de Young des structures mécaniques, sous sollicitation thermomécanique et de comparer l'efficacité et la rapidité de cette méthode avec les méthodes classiques.

Nous présentons dans le premier chapitre, les fondements biologiques des réseaux de neurones, leur modélisation qui est la base des réseaux artificiels et de leurs différents types, une étude sur le concept de l'apprentissage, ainsi que les différents types de l'apprentissage, suivi d'un aperçu des domaines d'utilisation des réseaux de neurones.

Le deuxième chapitre sera consacré à l'étude du modèle de la rétro propagation d'erreur, qui sera détaillé, en commençant par la représentation du modèle de La rétro propagation, son apprentissage, à la fin du chapitre nous présentons un résumé de l'algorithme.

Dans le troisième chapitre, nous présentons trois applications traitées par cette méthode (**un réseau perceptron multicouche avec la retro propagation**).

- **Une première** application consiste à tester cette méthode pour prédire les grandeurs physiques d'une structure (la contrainte maximale de Von mises, ainsi que sa localisation), et de comparer ses résultats avec celles de la méthode des éléments finis.

- **Une deuxième**, consiste à localiser l'excitation générée par des défauts dans un arbre d'une machine (assimilé à une poutre), à partir de la mesure de la flèche.

- **Une troisième** application a pour objectif l'identification et la localisation de l'endommagement(détermination du module de Young) dans des poutres d'un portique, connaissant les fréquences naturelles.

- **Une conclusion générale** à travers laquelle on essaye de montrer l'importance de cette méthode pour résoudre des problèmes complexes qui ne sont connus, que par des données mesurées de certains paramètres physiques décrivant ces problèmes.

CHAPITRE I

GENERALITE

CHAPITRE I

GÉNÉRALITÉS SUR LES RÉSEAUX DE NEURONES

I.1. INTRODUCTION ET HISTORIQUE :

Le cerveau humain est un appareil de calcul très complexe qui reste inaccessible. Sa grande capacité de penser, de mémoriser et de résoudre les problèmes a inspiré les scientifiques, qui ont essayé de modéliser son fonctionnement durant ces opérations.

C'est ainsi, qu'un groupe de chercheurs a crée un modèle de calcul, qui tend à imiter le cerveau humain. Ce qui a introduit l'étude du calcul neuronal (réseaux de neurones).

La première période a commencé il ya une dizaine d'années avec William James, et s'est terminé par la publication du perceptron et c'est ainsi que les progrès ont commencé.

En 1943, Mac-Culloch et Pitts ont constitué le premier modèle simplifié du neurone biologique **[19],** appelé aussi neurone formel.

En 1949, le psychologue Donald O. Hebb a introduit le terme de connexionnisme, pour parler des modèles massivement parallèles et connectés. Il a été le premier à définir la règle de mise à jour des poids synaptiques. Connue sous le nom "règle de Hebb " **[19]**.

En 1958, le psychologue Frank Rosenblatt a développé le modèle du perceptron **[17]**.Il s'agit d'un réseau de neurones, capable d'apprendre à différencier des formes simples, et à calculer certaines fonctions logiques. Il est inspiré du système visuel. Il a réussi à l'appliquer Pour la reconnaissance de formes.

Les travaux de Rosenblatt (1958), ont vécu un grand intérêt dans le milieu Scientifique.Il a définit une structure de réseau appelée Perceptron, qui a la caractéristique de traiter les problèmes linéairement séparables.

Au début des années 60, Bernard Widrow et Marcian Hoff ont inventé le réseau Adaline (Adaptive Linear), il se distingue par un algorithme d'apprentissage plus rapide que de celui du perceptron. Cet algorithme ajuste les poids par la minimisation de l'erreur quadratique, cette minimisation appelée aussi 'la descente du gradient'.

En 1969, deux scientifiques américains, appelés Minsky et Papert **[19]** ont publié un livre, dans lequel ont démontré les limites du perceptron proposé par Rosenblatt. En particulier, son incapacité de traiter des problèmes non linéaires.
De son coté, Teuvo Kohonen (1972) a introduit un modèle, il l'a appelé Mémoire Associative. Son apprentissage est non supervisé, il a trouvé son application, dans la reconnaissance de forme et de parole.

Les travaux se sont ensuite, ralentis considérablement jusqu'aux années 80. Précisément en 1982, où John Hopfield a donné un nouveau souffle aux réseaux de neurones, en publiant un article introduisant un nouveau modèle de réseaux « nommé réseau complètement connecté » **[19].** Parallèlement, Werbos a conçu un mécanisme d'apprentissage, pour les réseaux multicouches de type perceptron : la rétro propagation (Back-Propagation).

Cet algorithme qui permet de propager l'erreur vers les couches cachées sera Popularisé en 1986 dans un livre "Parallel Distributed Processing" par Rumelhart et al **[15]**. Les réseaux de neurones ont par la suite connu un essor considérable, et plusieurs applications sont apparues , ce qui a donné un avancement significatif à la science neuronale **[10].**
Actuellement, Les réseaux de neurones sont exploités sous forme de circuit intégré,

et qui sont utilisés dans des domaines très variés, citons à titre d'exemple, la reconnaissance de forme, les problèmes d'optimisation, la prédiction, l'évaluation et le contrôle.

I.2. PRINCIPES DES RÉSEAUX DE NEURONES :

I.2.1. DU NEURONE BIOLOGIQUE AU NEURONE ARTIFICIEL :

Le cerveau humain contient de l'ordre de 10 milliards de neurones.

Un neurone biologique est une cellule vivante, consacrée au traitement de l'information (Figure 1.1). De son corps cellulaire ou soma, rayonnent de nombreuses dendrites (jusqu'à 100000) qui reçoivent des signaux provenant d'autres neurones ou cellules sensorielles**.**

Ces signaux sont traités par le neurone, qui transmet à son tour un signal, si certaines conditions sont réunies le long de son axone à d'autres neurones, ou à de cellules effectrices (cellule musculaire par exemple) : On dit que le neurone est alors activé. La (Figure 1.2) montre la structure d'un neurone artificiel. Chaque neurone artificiel est un processeur élémentaire. Il reçoit un nombre de variables d'entrées, en provenance des neurones en amont. A chacune de ces entrées, est associé un poids w représentatif de la force de connexion. Chaque processeur élémentaire est doté d'une sortie unique, qui se ramifie ensuite, Pour alimenter un nombre de neurones en avals, à chaque connexion est associé un Poids.

Le modèle général du neurone artificiel, est composé des éléments suivants :

– Une ou plusieurs entrées pondérées.

– Un sommateur ; Une fonction de transfert ; Une sortie. S ;et une entrée xi .

– wij est la valeur du poids synaptique reliant l'entrée i au neurone j.

– Σ est la somme pondérée des entrées xi.

– F () est la fonction de transfert.

– s est la sortie du neurone. $S = F(\sum wij.\ xi)$.

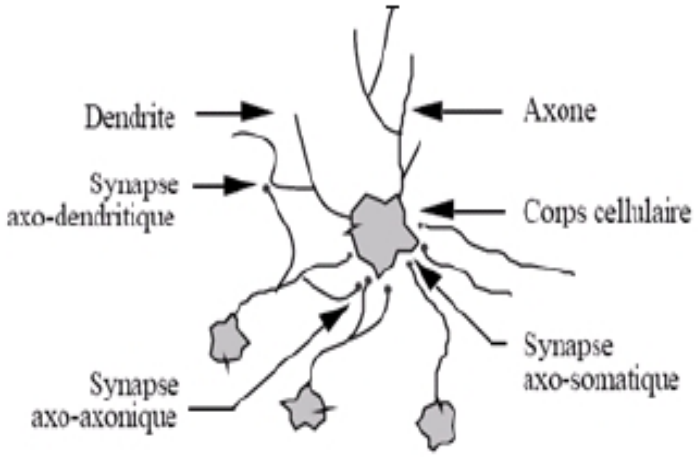


Figure 1.1 : Un neurone avec son arborisation dendritique

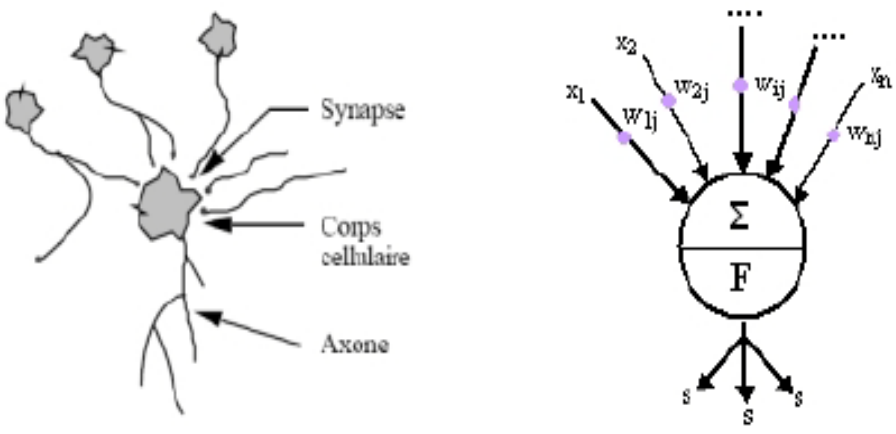


Figure 1.2 : Mise en correspondance neurone biologique / neurone artificiel

I.2.2. FONCTIONNEMENT DU NEURONE ARTIFICIEL :

Définition: Du point de vue modélisation mathématique, on peut définir un neurone artificiel, Par les quatre éléments suivants :

- La nature des entrées (xi), et des sorties (s) : Elles peuvent être :

– Binaires : (-1 ; +1) ou (0,1).

– Réelles.

- La fonction d'entrée totale, qui définit le prétraitement effectué sur les entrées. Elle peut Être :

– Booléenne (0 ou 1)

– Linaire.

– Affine.

– Polynomiale de degré supérieur à deux.

- La fonction d'activation du neurone qui définit son état en fonction de son entrée totale. Elle peut être :

– Une fonction binaire à seuil ; fonction de Heaviside (Figure 1.3.a) ou la fonction signe (Figure 1.3.b) :

– Une fonction linéaire à seuil (Figure 1.4.a) : SATUR (x).

– Une fonction sigmoïde (Figure 1.4.b) : sig(x)=a.1/1+e-x.

– La fonction de sortie qui calcule la sortie du réseau en fonction de son état d'activation ; en général cette fonction est considérée comme la fonction identité.

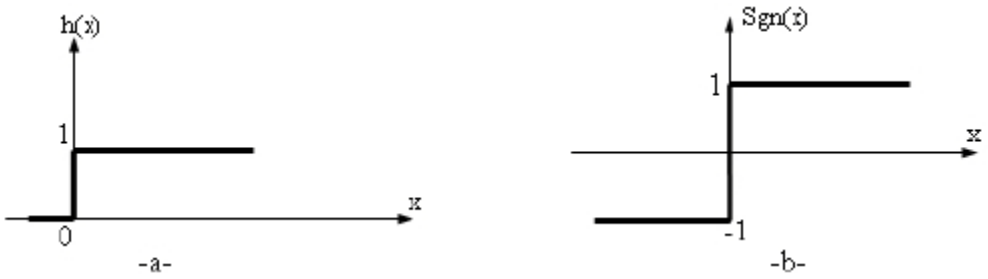


Figure 1.3 : a-Fonction de Heaviside, b-Fonction signe

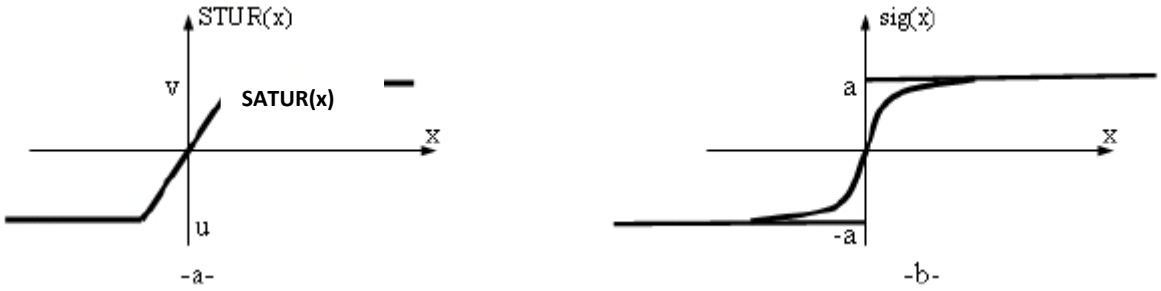


Figure 1.4 : a-Fonction linéaire à seuil, b-Fonction sigmoïde

I.3. LES PRINCIPAUX TYPES DE RÉSEAUX DE NEURONES :

On distingue deux types de réseaux de neurones : Les réseaux non bouclés et les réseaux bouclés.

I.3.1. RÉSEAUX DE NEURONES NON BOUCLÉS (OU STATIQUES) :

Un réseau de neurones non bouclé, est donc représenté graphiquement par un ensemble de neurones connectés entre eux, l'information circulant des entrées vers les sorties sans "retour en arrière"**[22]**. Le graphe d'un réseau non bouclé est acyclique.

En effet, si on se déplace dans ce type de réseau à partir d'un neurone quelconque en suivant les connexions, on ne peut pas revenir au neurone de départ. Les neurones qui effectuent le dernier calcul de la composition de fonctions, sont les neurones de sortie. Ceux qui font des calculs intermédiaires, sont les neurones cachés (Figure 1.5).

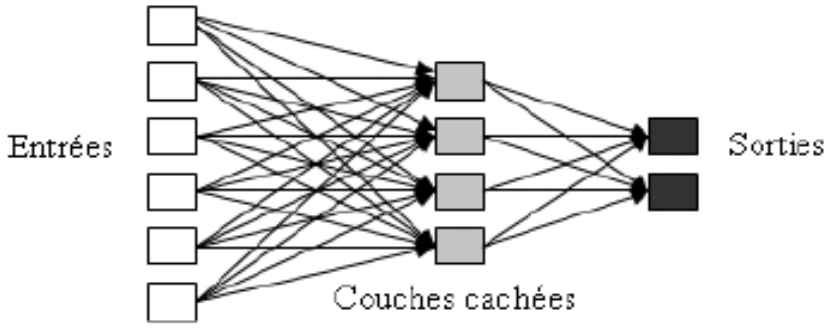


Figure 1.5 : Structure d'un réseau de neurones non bouclé

Les réseaux non bouclés à couche, sont structurés tel que les neurones, qui appartiennent à une même couche, ne soient pas connectés entre eux, chacune des couches recevant des signaux de la couche précédente, et transmettant le résultat de ses traitements à la couche suivante. Les deux couches extrêmes correspondent à la couche d'entrée, qui reçoit ses entrées du milieu extérieur d'une part, et à la couche de sortie qui fournit le résultat des traitements Effectués d'autre part. Les couches intermédiaires sont appelées couches cachées, leur nombre est variable.
Sous cette catégorie de réseaux, on trouve le perceptron, l'Adaline, le Madaline, le perceptron multicouche et le RBF (Radial Basic fonction).

I.3.1.1. LE PERCEPTRON :

Le perceptron était la première signification complète de l'architecture adaptive, inventé par Frank Rosenblatt en 1957, le perceptron est actuellement une classe entière des architectures.il suit un apprentissage supervisé selon la règle de Hebb. Il est limité à deux couches (couche d'entrée et couche de sortie), il a été connu pour son utilisation dans la reconnaissance de forme, dans la classification et pour résoudre des opérations logiques ('ET' ou 'OU') **[13]**.
Sa principale limitation est qu'il ne peut résoudre, que les problèmes linéairement séparables. Ces limitations ont été publiées dans un livre par minsky et parpet **[4].**

Chaque neurone j calcule la somme selon la formule :

$$X_j = \sum W_{ji}\, x_i$$

La sortie du neurone j est calculée suivant une fonction de transfert à seuil (Figure 1.8).

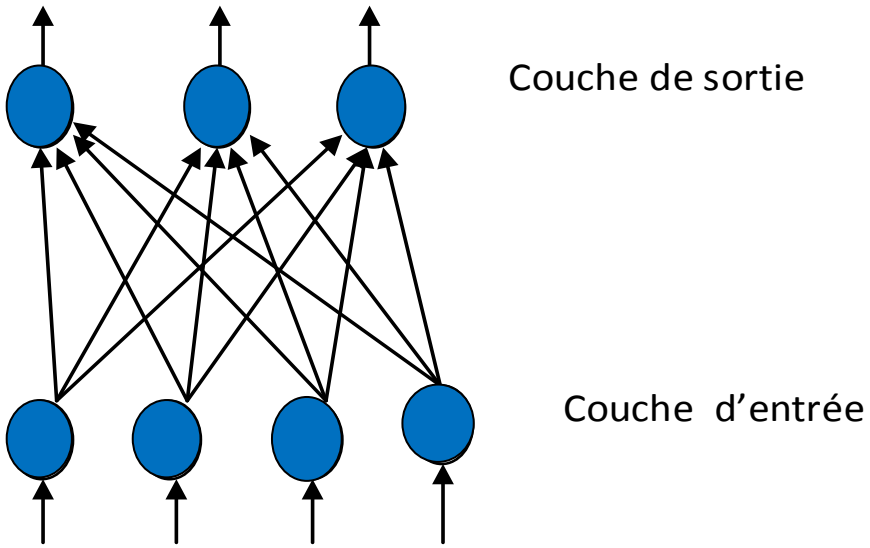


Figure 1.6 : Perceptron à deux couches

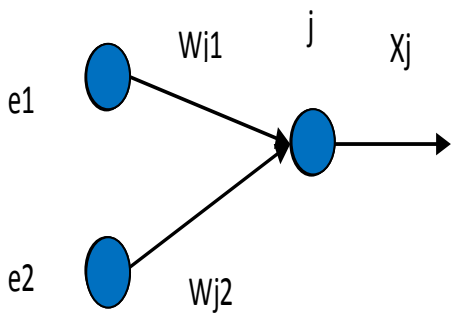


Figure 1.7 : Unité de traitement du perceptron

- $S_j = 1$ si $X_j > 0$
- $S_j=0$ si $X_j < 0$

Le calcul des poids se fait comme il est montré :

- $\Delta W_{ij} = \eta (S_i - d_i)e_j$ ou :
- η: pas d'apprentissage.
- S_i : Sortie fournie.
- d_i : Sortie désirée
- e_j : entrée.

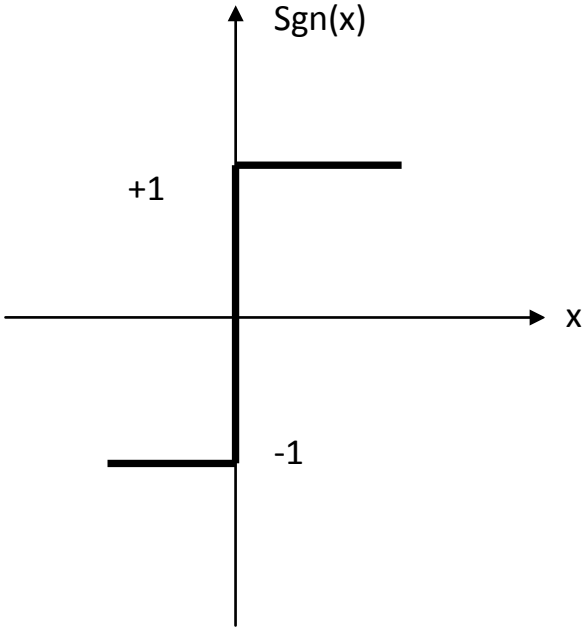


Figure 1.8 : Fonction de transfert à seuil

I.3.1.2. L'ADALINE ET LE MADALINE :

L'architecture Adaline (Adaptive Linear Element) a été inventée par Bernard widrow en 1959.

L'architecture Madaline (Multi layer Adaline) inventée aussi par B.Widrow a une configuration qui inclut deux Adaline ou plus (Figure 1.10).

L'Adaline et le Madaline utilisent la fonction de transfert linéaire, et un type d'apprentissage similaire à celui du perceptron **[14]**.

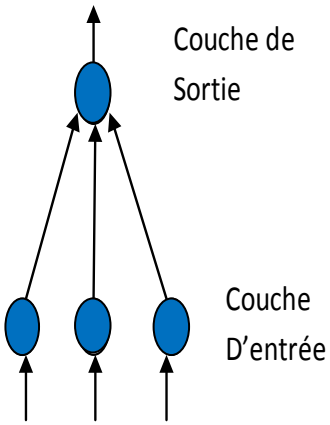


Figure 1.9 : Architecture Adaline

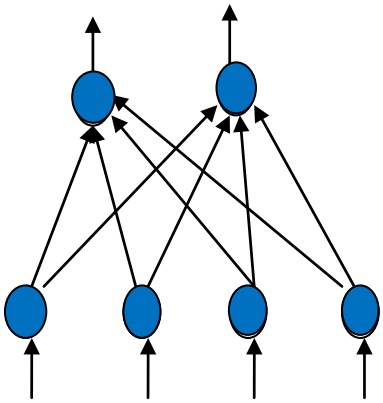


Figure 1.10 : Architecture Madaline

I.3.1.3. LE PERCEPTRON MULTICOUCHE :

Bien que les réseaux monocouches, soient déjà largement utilisés dans plusieurs applications, comme la reconnaissance de visages et le traitement de signal, leur efficacité pour traiter les problèmes compliqués est très limitée.les théorèmes de limitation du perceptron mis au point par minsky et papert **[9]** ; qui ont conclu que les cellules d'association dans le perceptron sont codées à la main, et pas par un algorithme d'apprentissage **[4]**.

L'idée de base des réseaux multicouches, est tout simplement de décomposer une tache, en plusieurs étages, dont chacun est censé être plus simple à résoudre. La différence entre le perceptron et le perceptron multicouche, réside dans la façon de génération des cellules intermédiaires, elles sont à la main dans le perceptron, tandis que la fonction des cellules intermédiaires dans un réseau multicouche, pourrait être générer automatiquement par un algorithme d'apprentissage **[9]**.

La caractéristique fondamentale de l'apprentissage est la règle du gradient, les unités de ces réseaux sont organisées en couches, les neurones d'une couche sont connectés à la couche suivante seulement (il n'ya pas de rétroaction).

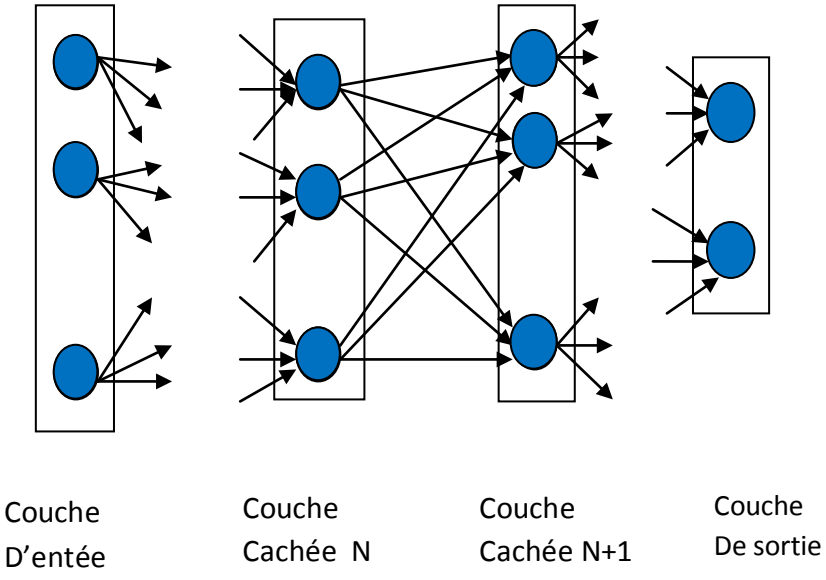


Figure 1.11 : Architecture d'un perceptron multicouche

Les réseaux multicouches comportent une couche d'entrée, une de sortie et une ou plusieurs couches cachées **[8]**.

L'apprentissage de ce type de réseau est un apprentissage supervisé.

Son architecture est arbitraire et dans la plus part des cas, il faut essayer plusieurs configurations, pour y arriver à l'architecture convenable au problème envisagé.

I.3.1.4. LE RÉSEAU À FONCTION RADIALE :

C'est un réseau que l'on nomme aussi RBF (Radial Basic Fonction) .L'architecture est la même que pour les perceptrons multicouches, cependant la fonction de base utilisée est la fonction gaussienne. Le réseau 'RBF' est employé dans les mêmes types de problèmes que les PMC. Les règles d'apprentissage sont soit, la règle par correction de l'erreur soit, la règle par compétition.

I.3.2. RÉSEAUX DE NEURONES BOUCLÉS (OU RECCURENTS) :

Un réseau de neurones bouclé est schématisé par un graphe des connexions, qui est cyclique.

Lorsqu'on se déplace dans le réseau en suivant le sens des connexions, il est possible de trouver au moins un chemin, qui revient à son point de départ (un tel chemin est désigné sous le terme de "cycle"). Ainsi, un retard entier, multiple de l'unité de temps choisie est attaché à chaque Connexion d'un réseau de neurones bouclé (ou à chaque arrête de son graphe). Une grandeur à un instant donné ne peut pas être fonction de sa propre valeur au même instant. Tout cycle du graphe du réseau doit avoir un retard non nul.

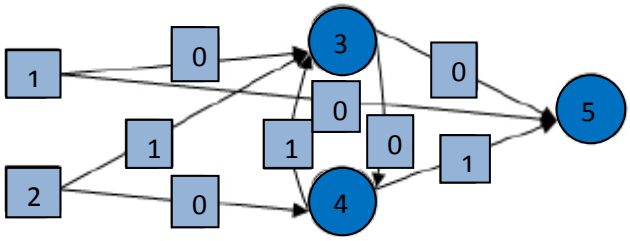


Figure 1.12 : Un réseau de neurones bouclé à deux entrées

La Figure (1.12) représente un exemple de réseau de neurones bouclé. Les chiffres dans les carrés indiquent le retard attaché à chaque connexion, exprimé en multiple de l'unité de Temps. Ce réseau contient un cycle, qui part du neurone 3 et revient à celui-ci en passant par le neurone 4. La connexion de 4 vers 3 ayant Un retard d'une unité de temps.

Sous cette catégorie on distingue : le modèle de Kohonen et le modèle de Hopfield.

I.3.2.1. MODÈLE DE KOHONEN (CARTE AUTO-ORGANISATRICE) :

Ce modèle a été présenté par Kohonen en 1982 , il s'appuie sur l'observation neurophysiologique**[8]**.une carte auto-organisatrice de Kohonen est un réseau à deux couches , la première couche est la couche d'entrée, la deuxième couche qui est organisée comme une grille à deux dimensions est la couche de sortie, toute

interconnexion vient de la première couche vers la deuxième couche, et les deux couches sont complètement connectées, de telle façon, que chaque neurone de la couche d'entrée est connecté à tous les neurones de la couche de sortie Figure (1.13). **L'apprentissage se déroule comme suit :**

- **Première étape** :

Quand un motif d'entrée est présenté, les unités de la deuxième couche font les sommes de leurs entrées, et concourent de trouver une seule unité gagnante : $\|E - Ui\|$ avec :

E : (e_1,……e_n) motif d'entrée.

U_i = (U_{il},……U_{in}) vecteur de poids.

Cette valeur est la distance entre le vecteur E et les poids U_i ; Calculée par : $\sqrt{(\sum ej - Uij)^2}$; et la petite valeur gagne la compétition, cette unité est notée c choisie tel que :$\|E - Uc\| = min\{\|E - Ui\|\}$

Si deux unités ont la même valeur, alors par convention, on prend l'unité avec le petit indice.

- **Deuxième étape** : Apres l'identification de l'unité gagnante c, la deuxième étape consiste à identifier ses voisins.

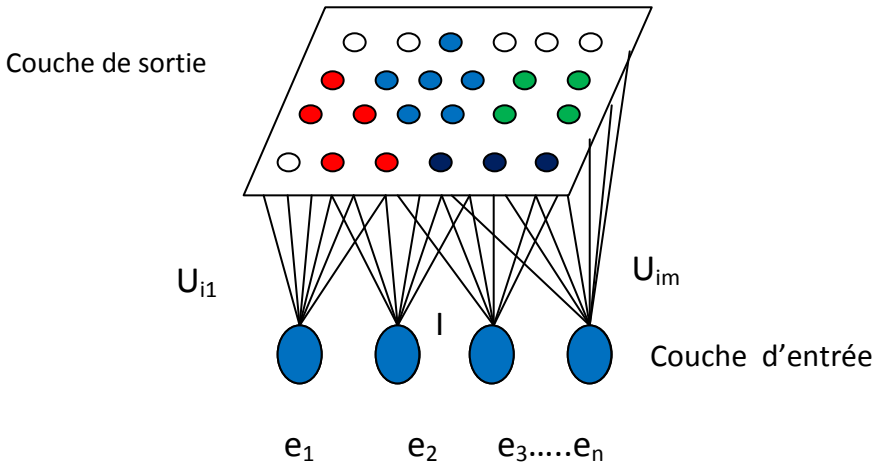


Figure 1.13 : Architecture de base d'une carte auto-organisatrice

- Le voisinage dans ce cas, consiste aux unités qui sont dans le carré, qui entoure l'unité c. les voisins sont dénotés par l'ensemble des unités $N_{c.}$

Les poids sont adaptés pour tous les neurones, qui sont au voisinage de l'unité c par l'équation:

$\Delta u_{ij} = \eta (e_j - u_{ij})$ si i est voisin de c

$\Delta u_{ij} = 0$ sinon

η : coefficient d'apprentissage. **[11][8].**

I.3.2.2. MODÈLE DE HOPFIELD :

En 1982 John Hopfield a publié un article intitulé « Neural network and physical system with émergent collective computation habilites », dans lequel, il introduit l'architecture du réseau et il a décrit comment les capacités de calcul peuvent être construites, il illustre une mémoire associative, qui peut être implantée avec Son réseau **[13]**. Selon lui, le système recherche des états stables, attracteurs dans son espace d'états. Les états voisins, tendent à se rapprocher d'un état stable, ce qui autorise la correction des erreurs et la capacité à compléter des informations manquantes.

Hopfield propose un modèle basé sur le réseau de Mc-Culloch et Pitts, tous interconnecté et utilise la règle d'apprentissage de Hebb **[5]**.

I.3.2.2.1. LE RÉSEAU DE HOPFIELD BINAIRE:

Ce réseau a une seule couche de neurones, chaque neurone a un état binaire (0 et 1). Le réseau est capable d'acquérir un état à chaque moment, cet état est un vecteur de 0 état et de 1 état (0_S ,1_S), chaque entrée dans le vecteur des états correspond à un neurone de réseau, à chaque moment on peut représenter le réseau par un vecteur des états :

$U= (e_1,\ldots\ldots\ldots\ldots,e_n) = (+++----++---+)$ ou :

+ correspond à une valeur binaire 1 ; **-** correspond à une valeur binaire 0

Les neurones du réseau de Hopfield sont complètement connectés. Cette topologie d'interconnexion rend le réseau « récurrent », car les sorties de chaque neurone peuvent être utilisées comme entrées pour les autres Neurones, cette organisation récurrente va permettre au réseau de se détendre à un état stable dans l'absence d'une entrée externe.

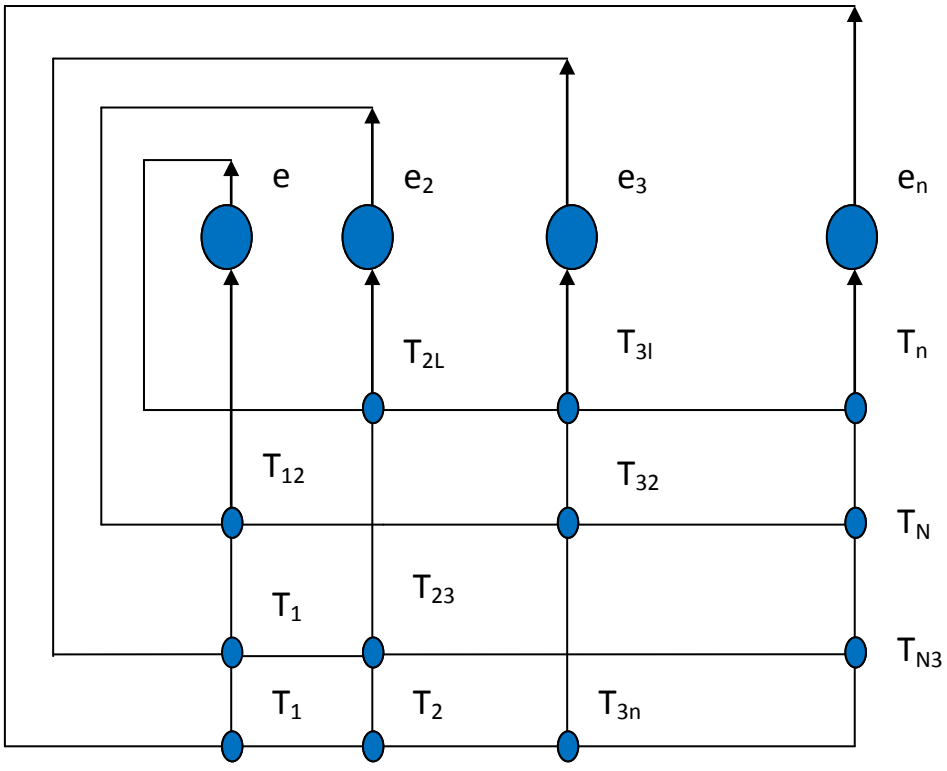
- **<u>Procédure d'adaptation des poids</u>** :

La figure (1.14) illustre le traitement de base fait par une unité de réseau de Hopfield binaire, accordé à l'équation suivante : $Sj = \sum e_i W_{ji}$ ou : i= 1…….n

Quand cette somme est calculée, on fait l'évaluation suivante :

Si $S_j \geq 0$ alors $e_j = 1$; Si $S_j < 0$ alors $e_j = 0$

On adapte les unités en séquence alors il faut répéter la séquence jusqu'à atteindre un état stable.



<u>Figure 1.14</u> : Réseau de Hopfield

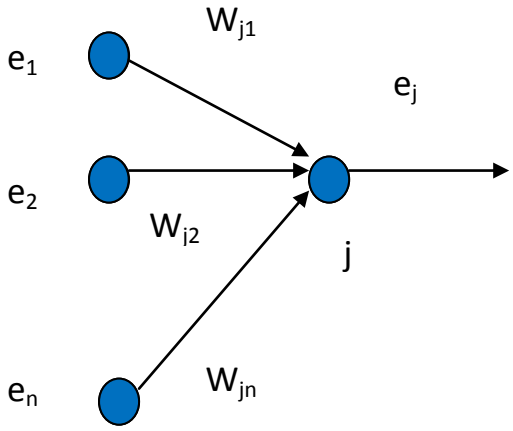


Figure 1.15 : Unité de traitement d'un réseau de Hopfield

I.4. APPRENTISSAGE DES RÉSEAUX DE NEURONES :

Pour un réseau de neurones, l'apprentissage peut être considéré comme le problème de la mise à jour des poids de connexions au sein du réseau, afin de réussir la tache qui lui est demandée.

L'apprentissage est la caractéristique principale des RNA et il peut se faire de différentes manières et selon différentes règles **[27].**

I.4.1. L'APPRENTISSAGE SUPERVISÉ :

L'apprentissage "supervisé" pour les réseaux de neurones formels, consiste à calculer les coefficients synaptiques, de telle manière que les sorties du réseau soient, aussi proches que possible des sorties "désirées". Ils peuvent être la classe d'appartenance de :

- La forme que l'on veut classer.
- La valeur de la fonction que l'on veut approcher.
- La sortie du processus que l'on veut modéliser.
- La sortie souhaitée du processus à commander.

On connaît donc, en tout point, ou seulement en quelques points, les valeurs que doit avoir la sortie du réseau en fonction des entrées correspondantes : C'est en ce sens que l'apprentissage est "supervisé". Cela signifie qu'un "professeur"**[8]** peut fournir au réseau des "exemples", de ce que celui-ci doit faire.

Ce type d'apprentissage, sera adopté par la suite dans notre algorithme de prédiction.

La plupart des algorithmes d'apprentissage des réseaux de neurones formels, sont des algorithmes d'optimisation : Ils cherchent à minimiser par des méthodes d'optimisation non Linéaires, une fonction de coût qui constitue une mesure de l'écart entre les réponses réelles du réseau et les réponses désirées.

Cette optimisation se fait de manière itérative, en modifiant les poids en fonction du gradient de la fonction de coût :

Le gradient est estimé par une méthode spécifique aux réseaux de neurones, dite méthode de rétro propagation, puis il est utilisé par l'algorithme d'optimisation proprement dit.

Les poids sont initialisés aléatoirement avant l'apprentissage, puis modifiés itérativement, jusqu'à obtention d'un compromis satisfaisant, entre la précision de l'approximation sur l'ensemble d'apprentissage, et la précision de l'approximation sur un ensemble de validation disjoint du précédent.

I.4.2. L'APPRENTISSAGE PAR RENFORCEMENT :

Le renforcement est en fait une sorte d'apprentissage supervisé. Dans cette approche le réseau doit apprendre la corrélation entrée/sortie via une estimation de son erreur, c'est-à-dire du rapport échec/succès. Le réseau va donc tendre à maximiser un index de performance qui lui est fourni, appelé signal de renforcement. Le système étant capable ici, de savoir si la réponse qu'il fournit est correcte ou non, mais il ne connait pas la bonne réponse.

I.4.3. L'APPRENTISSAGE NON SUPERVISÉ :

Un réseau de neurones non supervisé peut être également utilisé dans un but de visualiser ou d'analyser des données :On dispose d'un ensemble de données, représentées par des vecteurs de grande dimension. Et l'on cherche à les regrouper selon des critères de ressemblance qui sont inconnus à priori. Ce type de tâches est connu en statistique sous le nom de méthodes "d'agrégation".

On peut utiliser les réseaux de neurones bouclés pour réaliser une tâche assez voisine : à partir des données décrites par des vecteurs de grandes dimensions, on cherche à trouver une représentation de ces données dans un espace de Dimension beaucoup plus faible, (Typiquement de dimension 2) tout en conservant les "proximités" ou "ressemblances" entre ces derniers.

Il n'y a pas là, donc de "professeur", puisque c'est au réseau de découvrir les ressemblances entre les éléments de la base de données, et de les traduire par une proximité dans la "carte" de dimension 2 qu'il doit produire.

I.4.4. L'APPRENTISSAGE HYBRIDE:

L'apprentissage hybride utilise les deux approches. Puisque une partie des poids est déterminée par l'apprentissage supervisé, et l'autre partie par l'apprentissage non supervisé.

I.5. RÈGLES D'APPRENTISSAGE :

I.5.1. RÈGLE DE CORRECTION D'ERREUR :

Cette règle s'inscrit dans le paradigme de l'apprentissage supervisé, c'est à dire, on fournit au réseau une entrée et la sortie correspondante.si on considère y la sortie calculée par le réseau et d la sortie désirée, le principe de cette règle est d'utiliser (d-y) pour modifier les poids synaptiques en minimisant l'écart (d-y), jusqu'à ce que y=d.

I.5.2. RÈGLE DE HEBB :

Cette règle est basée sur des données biologiques, elle modélise le fait que si des neurones de part et d'autre d'une synapse sont activés, la force de connexion va croitre. L'apprentissage dans cette règle est localisé. Puisque la modification des poids ne touche que les neurones liés par cette synapse activée.

I.5.3. RÈGLE D'APPRENTISSAGE PAR COMPÉTITION:

Le principe de cet apprentissage est de regrouper les données en categories.les patrons similaires sont rangés dans une même classe, en se basant sur les corrélations des données, et seront représentés par un seul neurone « on parle du neurone gagnant ».Dans un réseau à compétition, chaque neurone de sortie est connecté aux neurones d'entrée, et aux autres neurones de sortie (connexions inhibitrices) et à lui-même (connexions excitatrices).

I.6. PROPRIETÉS DES RÉSEAUX DE NEURONES :

I.6.1. L'APPRENTISSAGE PAR L'EXEMPLE :

Contrairement aux systèmes experts, où la connaissance est explicitée sous forme de règles, les réseaux de neurones génèrent leurs propres règles par l'apprentissage à partir des exemples existants. L'apprentissage est fait à travers la règle, qui adapte les poids des connexions du réseau pour la réponse aux entrées de L'exemple, et éventuellement aux sorties désirées des entrées.

S'il s'agit d'un apprentissage supervisé, pour chaque activité des entrées, une activité de sortie est présentée au réseau, qui se configure à lui-même graduellement afin d'arriver à ce que les entrées-sorties désirés, soient obtenues. Si l'apprentissage est non supervisé, seules les activités de l'entrée sont indiquées aux réseaux, qui s'organisent à lui-même, afin que chaque élément de traitement caché,

réponde aux différentes séries d'activités présentés à l'entrée.

I.6.2. INSENSIBILITÉ AUX DÉFAILLANCES :

Les réseaux de neurones sont quasiment insensibles aux défaillances.

Cette insensibilité est défini par le fait que, si quelques éléments de traitements sont détruits, ou bien leurs connexions sont changées, le comportement global du réseau sera légèrement modifie. Cette insensibilité est due, au fait que l'information n'est pas localisée, elle est distribuée à travers toutes les unités du réseau **[21]**.

I.7. NORMALISATION DES ENTRÉES ET DES SORTIES :

I.7.1. Normalisation dans l'intervalle [0,1] : Bien que le choix des poids initiaux et des valeurs des paramètresη et α influent sur la convergence, la manière de présenter les données au réseau joue aussi un rôle très important dans la performance, Cela revient au choix de la méthode de Normalisation des données. On doit normaliser chaque type par sa valeur maximale. Cette normalisation des données est donnée par le schéma suivant : **Si** : V_ max est la valeur maximale dans la série de données. V_min est la valeur minimale dans la série de données, et ΔV= V_ max - V_min.

Alors pour chaque type de données (p) et pour chaque élément de la série des données(i) , on a : Entrée[p](i)N={Entrée[p](i)E - V_min [p] }/ΔV[p].

Ou N : désigne la valeur normalisé, et E : désigne la valeur brute (avant normalisation). Ce type de codage s'appelle le codage classique, car chaque type d'entrée est représenté par une unité.

Il arrive des fois, que la normalisation par la valeur maximale, ne donne pas de Bonnes résultats, soit pas de convergence, soit le temps écoulé est très grand.

Une méthode de normalisation typique peut être utilisée sur plusieurs intervalles.

I.7.2. Normalisation dans l'intervalle [0.2, 0.8] :

Etant donné que les valeurs extrêmes de l'intervalle [0,1] ne sont atteintes qu'asymptotiquement par la fonction sigmoïde, l'intervalle [0,1] est ramené à [0.2, 0.8] par la transformation linéaire suivante :

$X_N = 0.2 + 0.6 * X_E$ ou ; X_N : la valeur normalisé et X_E : la valeur brute.

I.7.3. Normalisation dans l'intervalle [0.3, 0.7] :

$X_N = 0.3 + 0.4 * X_E$ ou ; X_N : la valeur normalisé et X_E : la valeur brute.

I.7.4. Normalisation dans l'intervalle [0.4, 0.6] :

$X_N = 0.4 + 0.2 * X_E$ ou ; X_N : la valeur normalisé et X_E : la valeur brute.

I.8. DOMAINES D'UTILISATION DES RÉSEAUX DE NEURONES :

En plus de l'habilité des réseaux de neurones à apprendre et à construire des Structures uniques pour un problème particulier, ils fournissent une approche aussi voisine de la perception, et de la reconnaissance humaine, que celle des méthodes classiques. On présente ici quelques applications qui ont pu montrer le favoritisme des réseaux de neurones **[4]**.

I.8.1. TRAITEMENT DE LANGAGES :

I.8.1.1 CONVERSION DE TEXTE EN PAROLE :

La conversion du texte en parole est la transformation des symboles, à un langage parlé. Bien que la conversion du texte en parole ait déjà été réalisée par un autre Système de calcul, la nouvelle approche neuronale, élimine le besoin d'un programme très complexe pour générer les règles de prononciation dans la machine **[25], [26].** La première application neuronale à système dynamique a été le modèle appelé NETTALK **[28]**. C'est un réseau capable d'apprendre à prononcer l'anglais.son entrée est une fenêtre de sept caracteres.la sortie est la

correspondance Phonétique du caractère situé au centre de la fenêtre.

I.8.2.LA RECONNAISSANCE DE CARACTÈRES :

Le système de calcul neuronal, peut traiter de grandes quantités d'informations à la fois. Ils sont surtout doués dans la reconnaissance des termes. Ces Caractéristiques, leurs permettent d'être utilisés dans la reconnaissance des Caractères, dans l'interprétation visualisée et dans la classification des symboles. Parmi ces applications on peut citer :

I.8.2.1. LA RECONNAISSANCE DE L'ÉCRITURE :

Un système de calcul neuronal a été développé **[4]** dans cette pensée .il accepte l'écriture sur pilote numérisé, semblable à l'entrée d'un réseau, après avoir été entrainé par une série de type d'écritures. Le système est capable d'interpréter un nouveau type d'écriture, qui lui est inconnu.la précision et l'exactitude de ce système peuvent encore être améliorée par une session d'entrainement de ce type. Apres une phase d'apprentissage, le système sera capable de reconnaitre une variété de styles d'écritures, et de faire de meilleures estimations lorsqu'il se trouve en face d'un caractère embrouillant.

I.8.2.2. LA RECONNAISSANCE DES TERMES DEVELOPPÉS :

Les réseaux de neurones, dans ce domaine sont appliqués actuellement à la reconnaissance des chiffres et aussi à celle des caractères généraux **[13]**

I.8.3. LES PROBLÈMES DE CONTROLE :

La capacité des réseaux de neurones à apprendre par l'exemple des fonctions variées, font d'eux, de bons candidats à l'apprentissage des lois de commande en automatique et en robotique **[24]**.

I.8.4. LA CLASSIFICATION DES DONNÉES :

Un des grands problèmes de la classification est d'associer un motif d'entrée à un motif de sortie, parmi les nombreux motifs pré-spécifiés. Comme nous avons vu, les réseaux de neurones sont capables d'apprendre l'association entrée/sortie.

I.8.5. L'APPROXIMATION DES FONCTIONS :

Les capacités des réseaux sont exploités aussi, pour approximer une fonction inconnue (dont on connait les paires entrées/sorties). Ce type de problème est très fréquent dans le domaine de la modélisation et dans le domaine de l'ingénierie.

I.8.6. LA PRÉDICTION/PRÉVISION :

Les réseaux de neurones sont utilisés également, lorsqu'on a besoin de prédire les valeurs des données à un temps t+1, connaissant leurs valeurs à un instant t.

I.9. CONCLUSION :

Dans ce chapitre nous avons présenté l'historique des réseaux de neurones artificiels. Suivi des notions fondamentales sur les réseaux de neurones.

Nous avons aussi, présenté les règles d'apprentissage d'un réseau non bouclé et Bouclé .Enfin les différents domaines, où les réseaux peuvent être appliqués.

Dans le chapitre qui suit nous étudions en détail un type de réseaux « le modèle de la rétro propagation d'erreur » et qui sera la base des applications faites dans le 3^{eme} chapitre.

CHAPITRE II

LA RETRO PROPAGATION D'ERREUR

CHAPITRE II

LE MODÈLE DE LA RÉTRO PROPAGATION D'ERREUR

II.1. INTRODUCTION :

La rétro propagation, est un algorithme d'apprentissage pour résoudre les problèmes posés par minsky et papert dans le livre « perceptrons » .David Rumelhart, est la personne à qui est associée l'invention de la rétro propagation.

Plusieurs classes non linéaires, ont connues des solutions grâce aux réseaux multicouches. Cependant, si la sortie est erronée, comment peut-on arriver à déterminer quelle interconnexion, ou quelle unité doit être ajustée ? La rétro-propagation résout le problème par la correction de l'erreur, sur toutes les unités et interconnexions **[26].**

Elle emploie trois couches d'unités et plus, et utilise une interconnexion complète, et sans connexions entre les unités de la même couche. L'apprentissage est supervisé. C'est à dire que le réseau est présenté avec une paire de motifs : un motif d'entrée et un motif de sortie **[13].**

II.2. DESCRIPTION DU MODÈLE DE LA RÉTRO PROPAGATION :

*P*our décrire le modèle de la rétro propagation, on doit regarder dans un premier temps, chacun de ses éléments à part, et voir comment ils se combinent, pour former la topologie du réseau.

La topologie de ces réseaux est bien définie comme suit :

Le nombre des unités d'entrée et de sortie, est défini à partir du problème à étudier, le nombre de couches cachées doit obéir à un compromis optimisant Le temps d'apprentissage, en évitant le sur-apprentissage, qui est la conséquence d'un excès de neurones sur la couche cachée, et le sous-apprentissage résultant de

L'insuffisance de leurs nombres sur la même couche. Ce choix résulte souvent d'un savoir faire et d'une expérience pratique.

On présente ci-après, les équations décrivant l'entrainement et les opérations du réseau.

Ces équations sont réparties en deux catégories. Les équations décrivant la propagation de l'information de l'entrée à la sortie, et les équations déterminant l'erreur qui se propage à travers le réseau de la sortie vers l'entrée.

La figure (2.1) représente un modèle simple d'un réseau à trois couches, chaque neurone est représenté par un cercle, et chaque interconnexion est représentée par son poids associé, et par sa direction (une flèche).

Les (**b1**, **b2**) sont les biais, ils agissent sur les unités, Pour améliorer les propriétés de convergence du réseau.

Les **Ni** sont introduites dans la première couche. Toutes ces entrées sont présentées au réseau simultanément.

Les entrées doivent être normalisées entre **0** et **1**, puisqu' on utilise la fonction sigmoide.la continuité des données permet au réseau une flexibilité significative.

Après la normalisation des entrées. Que se passe t-il au niveau de la couche d'entrée?.

Les neurones d'entrée, distribuent l'information à travers les connexions aux neurones de la couche cachée. La sortie de chaque unité d'entrée, est égale à son entrée, puisque c'est la fonction linéaire qui est appliquée pour ces elements.de même, les unités de la couche cachée, distribuent l'information arrivée à travers la connexion à la couche de sortie.

On peut remarquer que d'après **la Figure (2.1)** l'information se propage de gauche à droite, c'est ce qu'on appelle « réseau sans Rétroaction ».

L'entrée de chaque élément de la couche cachée, est calculée selon l'équation :

$$i_j = \sum W_{ij} o_i \qquad (2.1)$$

Notons que, cette somme contient l'entrée du biais. Celle ci possède une sortie égale à 1 à tout instant.

W_{ij} est le poids de la connexion entre un élément **j** de la couche cachée et un Neurone i de la couche d'entrée.

i_j est l'entrée d'un élément j de la couche cachée.

O_i est la sortie d'un élément i de la couche d'entrée.

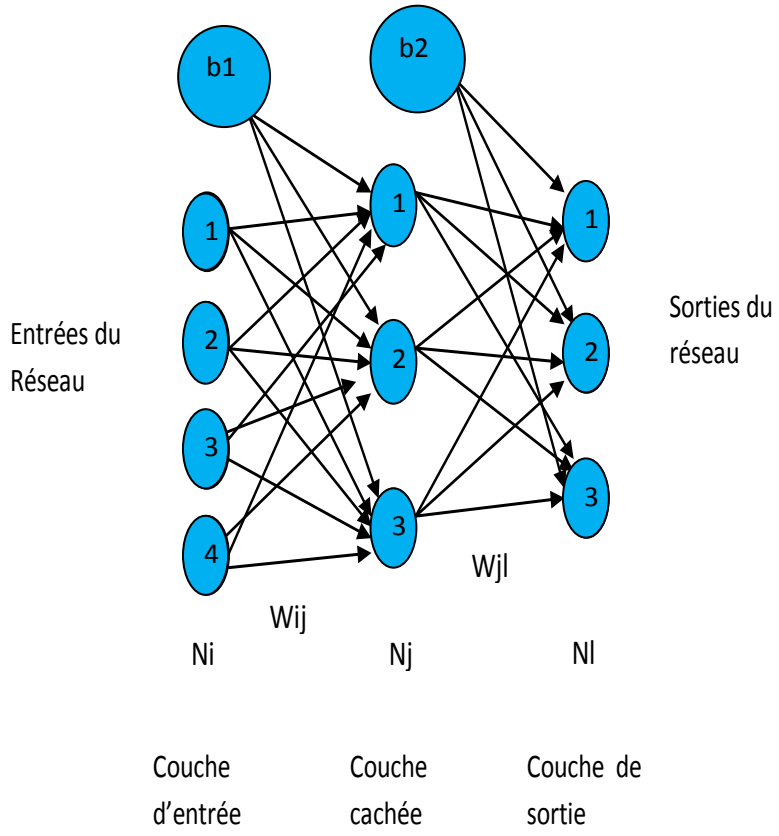


Figure 2.1 : structure d'un réseau de rétro propagation

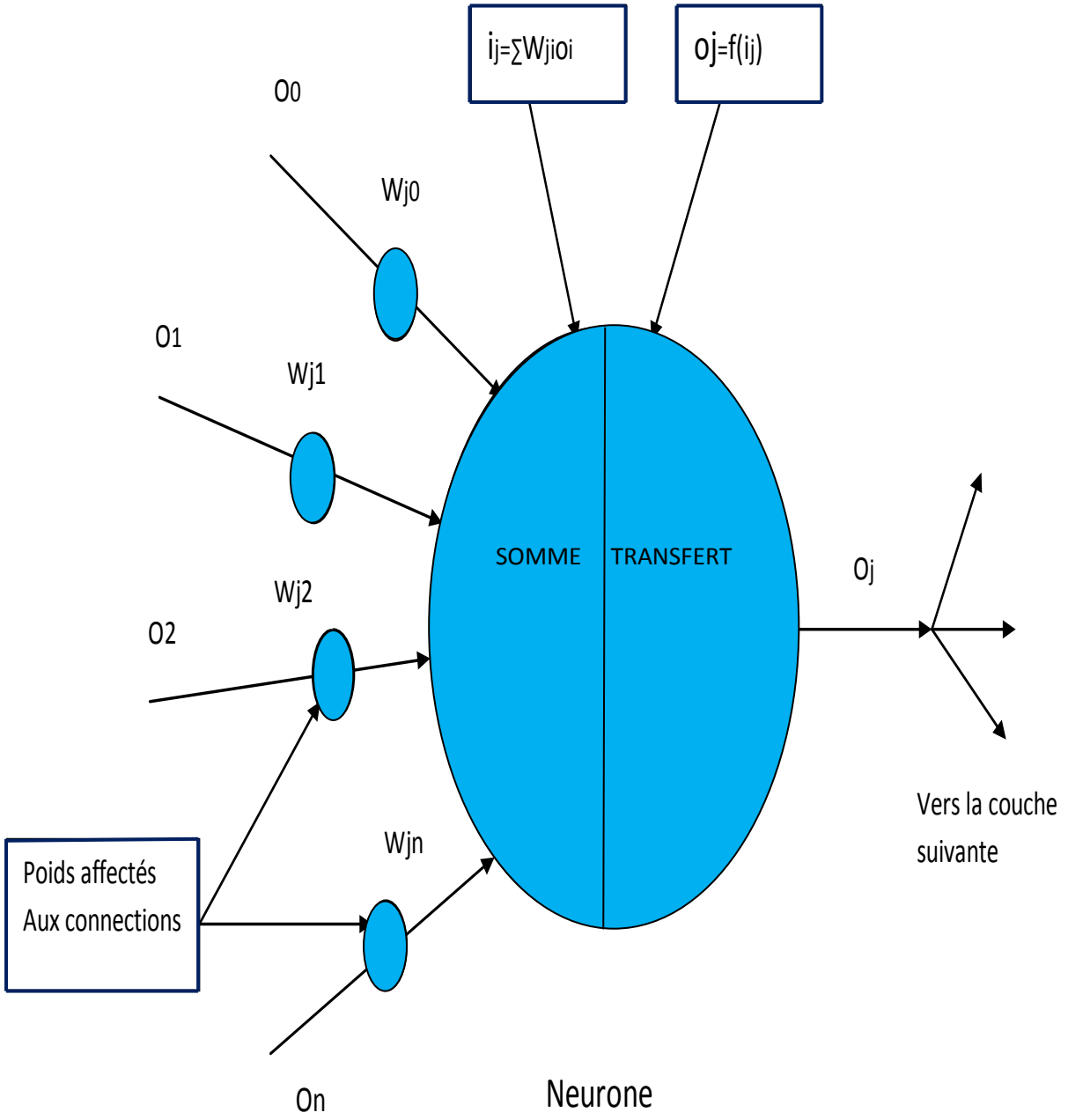


Figure 2.2 : principe de calcul d'un neurone caché

Le calcul de la sortie d'un élément de la couche cachée est donné par l'équation Suivante :

$$o_j=1/ (1+ \exp (-i_j)) \qquad (2.2)$$

ou bien : $o_j=f(i_j)$; Ou f est la fonction de transfert sigmoïde de l'élément j.

La non linéarité de cette fonction de transfert, joue un rôle très important dans la performance du réseau. Il est à noter que, toute fonction dérivable et croissante de l'entrée i_j , peut servir de fonction de transfert.

Toutefois, la plus utilisée est la fonction sigmoïde :

$$f(x)=1/e^{-x}+1 \text{ avec } f'(x)=e^{-x} / (e^{-x} + 1)^2$$

L'entrée et la sortie de chaque élément de la couche de sortie, sont calculées de la même manière en utilisant les équations (2.3) et (2.4)

$$i_l= \sum W_{lj}o_j \qquad (2.3)$$

$$o_j= f(i_l)= 1/ 1+\exp(-i_l) \qquad (2.4)$$

La sortie de chaque neurone de la sortie du réseau, constitue le résultat obtenu Pendant la propagation de l'information, de l'entrée vers la sortie à travers les unités cachées du réseau.

Deux opérations mathématiques sont exploitées par chaque neurone du réseau.

- **La première** opération est une sommation de sortie des unités affectées de poids des connections correspondants. - **La seconde** est une fonction d'activation qui donne la réponse à la sortie de chaque neurone du réseau. Cette fonction joue le rôle d'un amplificateur électronique.

II.3.MODÉLISATION DE L'APPRENTISSAGE SUPERVISÉ :

Le principe est simple. Partant d'une configuration aléatoire des poids, on Présente au réseau un certain nombre d'exemples, dont on connait la sortie.

Après chaque essai, on compare la sortie obtenue avec celle désirée, et on calcule L'écart entre les deux, puis on corrige les poids de façon à minimiser cet écart. Dans Ce qui suit, on présente la stratégie suivie pour la modification des poids. : D'après Rumelhart et mc-Clelland, l'erreur à calculer est définie comme la différence entre les sorties fournies par le réseau et les sorties désirées.

Cette erreur est donnée pour un exemple p par l'équation suivante :

$$Ep= 0.5 \sum_{l=1}^{nl}(t_l-o_l)^2 \qquad (2.5)$$

t_l : la sortie désirée à un élément l de la couche de sortie.

o_l : la sortie obtenue à un élément l de la couche de sortie.

nl : nombre d'élément sur la couche de sortie.

Le but du processus d'apprentissage est alors de minimiser cette erreur .On utilise la méthode de la descente du gradient pour minimiser le terme d'erreur. Cela revient à corriger les poids dans la direction indiquée par le gradient de (Ep) tel que pour l'exemple p.

$$\Delta W_{lj} =- \eta\, \partial Ep/\partial W_{lj} \qquad (2.6)$$

Ou η est un coefficient de proportionnalité appartient à l'intervalle] 0 ,1[, appelé pas d'apprentissage. La méthode du gradient consiste à calculer W_{lj}, après avoir présenté au réseau p exemples. ($E=\sum_{p=1}^{N} Ep$)

$$\Delta W_{lj}=-\eta\sum_{p=1}^{N} \partial Ep/\partial W_{lj} \qquad (2.7)$$

La sommation est effectuée sur tout l'ensemble des exemples. On doit préciser maintenant le calcul de $\partial Ep/\partial W_{lj}$, en montrant comment le réseau pourra effectuer ce calcul de façon locale. l et j sont des éléments appartenant à des couches adjacentes (l désigne les éléments de la couche de sortie, et j désigne les éléments de la couche cachée), et reliés par une connexion de poids W_{lj} . D'après les règles de dérivation des fonctions composées on peut écrire :

$$\partial Ep/\partial W_{lj}= (\partial Ep/\partial o_l)(\partial o_l/\partial i_l)(\partial i_l/\partial W_{lj}) \qquad (2.8)$$

D'après l'équation (2.3)

$$\partial i_l /\partial W_{lj} =o_j \qquad (2.9)$$

On pose :

$$\delta_l = - (\partial Ep/\partial o_l)(\partial o_l/\partial i_l) \quad (2.10)$$

δ_l s'interprète comme le signal d'erreur de l'élément l, C'est-à-dire la contribution de l'entrée i_l de l'unité l à l'erreur quadratique constatée à la sortie. On obtient alors la formule qui calcule $\partial Ep/\partial W_{lj}$ donné par :

$$\partial Ep/\partial W_{lj} = - \delta_l o_j \quad (2.11)$$

Par suite la correction du poids W_{lj} de la connexion reliant l'élément j à l'unité l est donnée par :

$$\Delta W_{lj}=\eta \delta_l o_j \quad (2.12)$$

On applique ensuite la formule (2.7) pour l'ensemble des exemples. Le signal d'erreur δ_l sera calculé par le principe de la retro propagation. Si on considère une unité de sortie l, on a (d'après l'équation : (2.5)

$$\partial Ep/\partial o_l = -(t_l-o_l) \quad (2.13)$$

D'après la fonction de transfert des unités du réseau (équation) on peut écrire :

$$\partial o_l / \partial i_l= f'(i_l)[1-f(i_l)] \quad (2.14)$$

En substituant (2.14) et (2.13) dans l'équation (2.10) on obtient la formule qui calcule le terme d'erreur :

$$\delta_l = f'(i_l)(t_l-o_l) = o_l(1- o_l)(t_l-o_l) \quad (2.15)$$

L'état d'un élément de sortie l, dépend de celui du neurone caché j. Si la couche de sortie contient nl éléments, le calcul de l'erreur quadratique Ep relative à l'exemple p, dépend de l'ensemble des unités l.de même ces dernières dépendent des éléments j de la couche cachée. Alors le signal d'erreur d'une unité j est :

$$\delta_j = - (\partial Ep/\partial o_j)(\partial o_j/\partial i_j) \quad (2.16)$$

Pour calculer $(\partial Ep/\partial o_j)$, on doit introduire toutes les dérivées partielles $(\partial Ep/\partial o_l)$:

$$\partial Ep/\partial o_j = \sum_{nl} (\partial Ep/\partial o_l)(\partial o_l/\partial o_j) \quad (2.17)$$

$$\partial Ep/\partial o_j = \sum_{nl} (\partial Ep/\partial o_l)(\partial o_l/\partial i_l)(\partial i_l/\partial o_j) \quad (2.18)$$

Donc $$\partial Ep/\partial o_j = \sum_{nl} (-\delta_l W_{lj}) \quad (2.19)$$

Le signal d'erreur à un élément de la couche cachée, est donc d'après les équations (2.14) et (2.16).

$$\delta_j = f'(i_j)\sum_{nl} \delta_l W_{lj} \quad (2.20)$$

On résume le principe de la retro propagation par les schémas (Figure 2.3). La formule (2.20), nous permet de calculer toutes les termes des erreurs des éléments d'une couche cachée, à partir de ceux de la suivante du réseau en direction de la sortie.

Le calcul de ce signal d'erreur, se fait par couches successives, en partant des éléments de sortie vers les neurones d'entrée. Ce calcul s'effectue au niveau de chaque élément du réseau, en utilisant les informations relatives aux éléments voisins. Ces éléments se propagent dans le sens inverse de l'information.

Lors du calcul de la réponse du réseau.la particularité du calcul du terme d'erreur justifie le nom de rétro propagation.

L'apprentissage selon la formule (2.12), conduit à des oscillations des valeurs des poids, alors que l'on considère que l'apprentissage est fini, lorsqu'ils convergent vers des valeurs stables. Ces oscillations dépendent du choix du coefficient d'apprentissage. Pour éviter un tel comportement, il est préférable de choisir un coefficient petit. Mais une faible valeur de ce coefficient, peut amener à une Longue durée d'apprentissage. Pour parer à ce problème, on ajoute un terme à la

précédente variation du poids. α appelé « momentum » : $\Delta W_{jk}(p)$, la dernière variation des poids W_{jk}.

Après la présentation du p ième exemple, on corrige alors cette variation selon la formule suivante :

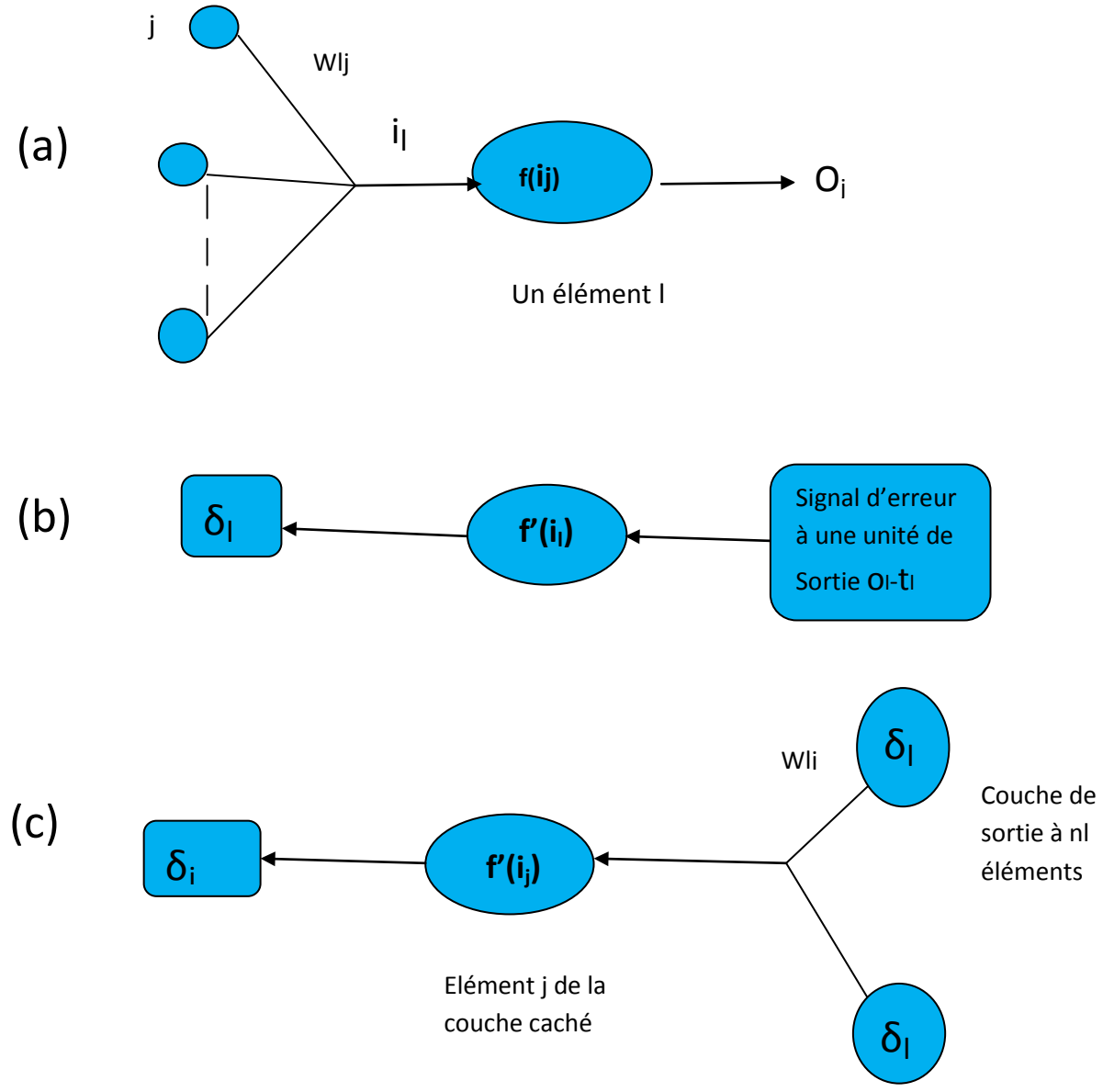


Figure 2.3 : principe de l'apprentissage par rétro propagation **(a-** architecture en couche et fonction de transfert ; **b-** calcul du signal d'erreur δl relatif à une unité de sortie ; **c-** calcul du signal d'erreur d'une unité cachée par rétro propagation**)**.

$$\Delta W_{jk}(p+1)= \eta\, \delta_j\, o_k + \alpha\, \Delta W_{jk}(p) \qquad (2.21)$$

Les valeurs de η et α, sont toujours comprises entre 0 et 1. Couramment 0.9 et 0.5 sont utilisés Respectivement pour η et α. ils peuvent varier selon la situation des neurones dans le réseau.

Le paramètre α, influe sur les termes $\Delta W_{jk}(p)$ comme un filtre passe-bas, filtrant les oscillations de haute fréquence des variations des poids $\Delta W_{jk}(p)$ selon la formule (2.21).

Le choix des poids initiaux, influe aussi sur la convergence de l'algorithme. Donc on prend les précautions suivantes : on ne doit surtout pas, les prendre tous égaux, car cela pousse tous les neurones à faire le même calcul. Ce qui résulte d'un blocage de l'apprentissage. Donc, il faut les prendre de manière aléatoire, et avec des valeurs suffisamment faibles afin d'éviter la saturation des neurones.

II.4- CRITÈRE D'ARRÊT :

Plusieurs critères d'arrêts, peuvent être utilisés avec l'algorithme de rétro propagation des erreurs.

Le plus commun, consiste à fixer un nombre maximum de périodes d'entrainement, ce qui fixe effectivement une limite supérieure sur la durée de l'apprentissage. Ce critère est important, car La rétro propagation des erreurs n'offre aucune garantie, quant à la convergence de l'algorithme.

Il peut arriver, par exemple, que le processus d'optimisation reste pris dans un minimum local. Sans un tel critère, l'algorithme pourrait ne jamais se terminer.

Un deuxième critère commun, consiste à fixer une borne inferieure sur l'erreur quadratique moyenne, ou encore sur la racine carrée de cette erreur. Dépendant de l'application, il est parfois Possible de fixer à priori un objectif à atteindre.

Lorsque l'indice de performance choisi, diminue en dessous de cet objectif, on considère simplement que le réseau à suffisamment bien appris ses données, et on

arrête l'apprentissage. **Les deux critères** précédents, sont utiles mais ils comportent aussi des limitations. Le critère relatif au nombre maximum de périodes d'entrainement, n'est aucunement lié à la performance du réseau. Le critère relatif à l'erreur minimale obtenue, mesure quant à lui un indice de performance.

Mais ce dernier, peut engendrer un phénomène dit de sur-apprentissage, qui n'est pas désirable dans la pratique, surtout si l'on ne possède pas une grande quantité de données d'apprentissage, ou si ces dernières ne sont pas de bonne qualité.

Un processus d'apprentissage par correction des erreurs, comme celui de la rétro propagation, il vise à réduire autant que possible l'erreur que commet le réseau. Mais cette erreur, est mesurée sur un ensemble de données d'apprentissage. Si les données sont bonnes, c'est-à-dire quelles représentent bien le processus physique sous-jacent que l'on tente d'apprendre, ou de modéliser, et que l'algorithme a convergé vers un optimum global, alors il devrait bien performer sur d'autres données issues du même processus physique. Cependant, si les données d'apprentissage sont partiellement corrompues, par du bruit, ou par des erreurs de mesure, alors il n'est pas évident, que la Performance optimale du réseau sera atteinte en minimisant l'erreur, lorsqu'on la testera sur un jeu de données différent de celui qui a servi à l'entrainement. On parle alors de la capacité du réseau à généraliser, c'est-à-dire de bien performer avec des données qu'il n'a jamais vu auparavant. Par exemple, **la figure 2.4** illustre le problème du sur-apprentissage dans ce contexte. La droite en pointillés montre une fonction linéaire que l'on voudrait approximer en ne connaissant que les points noirs.

La courbe en trait plein, montre ce qu'un réseau pourrait apprendre. On constate que la courbe passe Par tous les points d'entrainement, et donc que l'erreur est nulle. De toute évidence, ce réseau ne généralisera pas bien, si l'on échantillonne d'autres points sur la droite !.

Une solution à ce problème, consiste à utiliser un autre critère d'arrêt basé sur une

technique dite de **validation croisée** (en anglais «cross-validation»). Cette technique consiste à utiliser deux ensembles indépendants de données pour entrainer le réseau : un pour l'apprentissage (l'ajustement des poids) et l'autre pour la validation, c'est à dire le calcul d'un indice de performance (une erreur, un taux de reconnaissance, ou toute autre mesure pertinente à l'application).

Le critère d'arrêt consiste alors, à stopper l'apprentissage, lorsque l'indice de performance calculé sur les données de Validation cesse de s'améliorer pendant plusieurs périodes d'entrainement. **La figure 2.5** illustre Le critère de la validation croisée dans le cas d'un indice de Performance que l'on cherche à minimiser. La courbe en pointillés de ce graphique représente, l'indice de performance d'un réseau, calculé sur les données d'apprentissage, alors que La courbe en trait plein montre le même indice mais calculé sur les données de validation.

On voit qu'il peut exister un moment au Cours de l'apprentissage, où l'indice de Validation se détériore, alors que le même indice continue à s'améliorer pour les données d'entrainement. C'est alors le début du «sur-apprentissage».

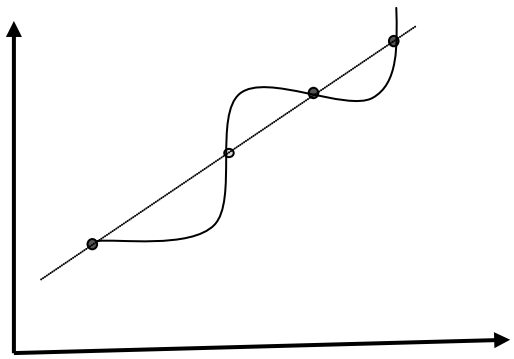


Figure 2.4 : – Illustration du phénomène de sur-apprentissage
Pour le cas simple d'une approximation de fonction.

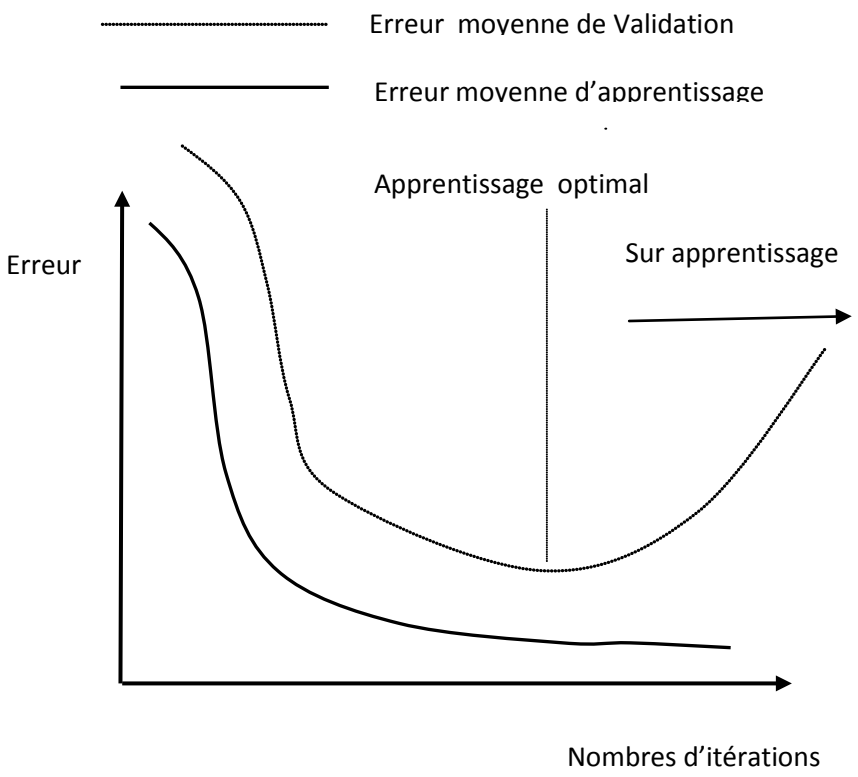


Figure 2.5 : – Illustration de la validation croisée

II.5. RESUMÉ DE L'ALGORITHME :

L'algorithme de rétro propagation se résume donc, par les étapes suivantes :

1- Partager les données en trois parties (données d'apprentissage, données de validation et données de test)

2- Initialiser tous les poids et biais à de petites valeurs aléatoires dans l'intervalle [−0.5, 0.5] ;

3- Normaliser les données ;]0,1[;

4- Permuter aléatoirement les données d'apprentissage ;

5- Pour chaque donnée d'apprentissage n :

a)-Calculer les sorties observées en propageant les entrées vers l'avant.

b)-Ajuster les poids en rétro propageant le gradient de l'erreur depuis la dernière

couche vers la première couche. $Wji(n) = wji\,(n-1) + \eta\delta j\,(n)\,yi(n) + \alpha\Delta w$.

Avec $0 \leq \eta \leq 1$ représentant le taux d'apprentissage.

Et $0 \leq \alpha \leq 1$ est un paramètre nommé « momentum » qui représente une espèce d'inertie dans le changement de poids.

6- Arrêt de l'apprentissage selon les critères suivants :

a)-La tolérance sur l'erreur est atteinte.

b)-L'erreur sur les données de validation commence à augmenter (sur-apprentissage ou oubli).

c)-Nombre d'itération atteint.

CHAPITRE III

ANALYSE EXEPERIMENTALE

CHAPITRE III

ANALYSE EXPÉRIMENTALE

III.1. APPLICATION N°1 :

Cette application a deux objectifs à concrétiser :

1- Examiner la capacité des réseaux de neurones (dans notre cas :un perceptron multicouche à rétro propagation d'erreur), pour prédire la localisation et la magnitude, de la contrainte maximale d'une structure sollicitée par une charge statique, connaissant la position de cette charge.

2- Montrer leur aptitude à remplacer les méthodes classiques pour résoudre des problèmes, nécessitant des formulations délicates, en exploitant des données obtenues par des expériences.

III.1.1 DONNÉES DU PROBLÈME :

Nous présentons une structure plane (panneau) de dimensions : 0.3m x 0.6m épaisseur 1mm avec une cavité à l'intérieur 0.2m x 0.2m ;(Figure 3.1), chargée d'une force axiale suivant XX , et fixée d'un coté (appui fixe),Coefficient de poisson v = 0.3, E =$2.1x10^5$ $N.mm^{-2}$, charge axiale Fx=100N.

L'objectif est de construire un réseau de neurones, capable de prédire la localisation et la valeur de la contrainte maximale selon le critère de Von mises, lorsque la charge statique **Fx** change de position.

Pour que le réseau puisse faire cette tache, il doit apprendre la relation existante entre ces grandeurs (variables), à partir des données qu'on met à sa disponibilité.

Pour cela, on génère par la méthode des éléments finis, une base de données par la procédure suivante :- On discrétise le panneau en 153 éléments et 106 nœuds, le

Panneau est fixé aux nœuds (6, 59, 58, 57,56, 55, 54, 53,8) Dans la direction x et y voir **figure (3.1)**.

- Chaque fois qu'on applique la charge Fx sur un nœud, on prend les coordonnées de la charge (**Xc**, **Yc**) , on calcule la contrainte maximale σ ainsi que sa localisation (**xσ**, **yσ**).

- Le tableau qui suit, montre les nœuds qui sont choisis pour appliquer la force Fx.

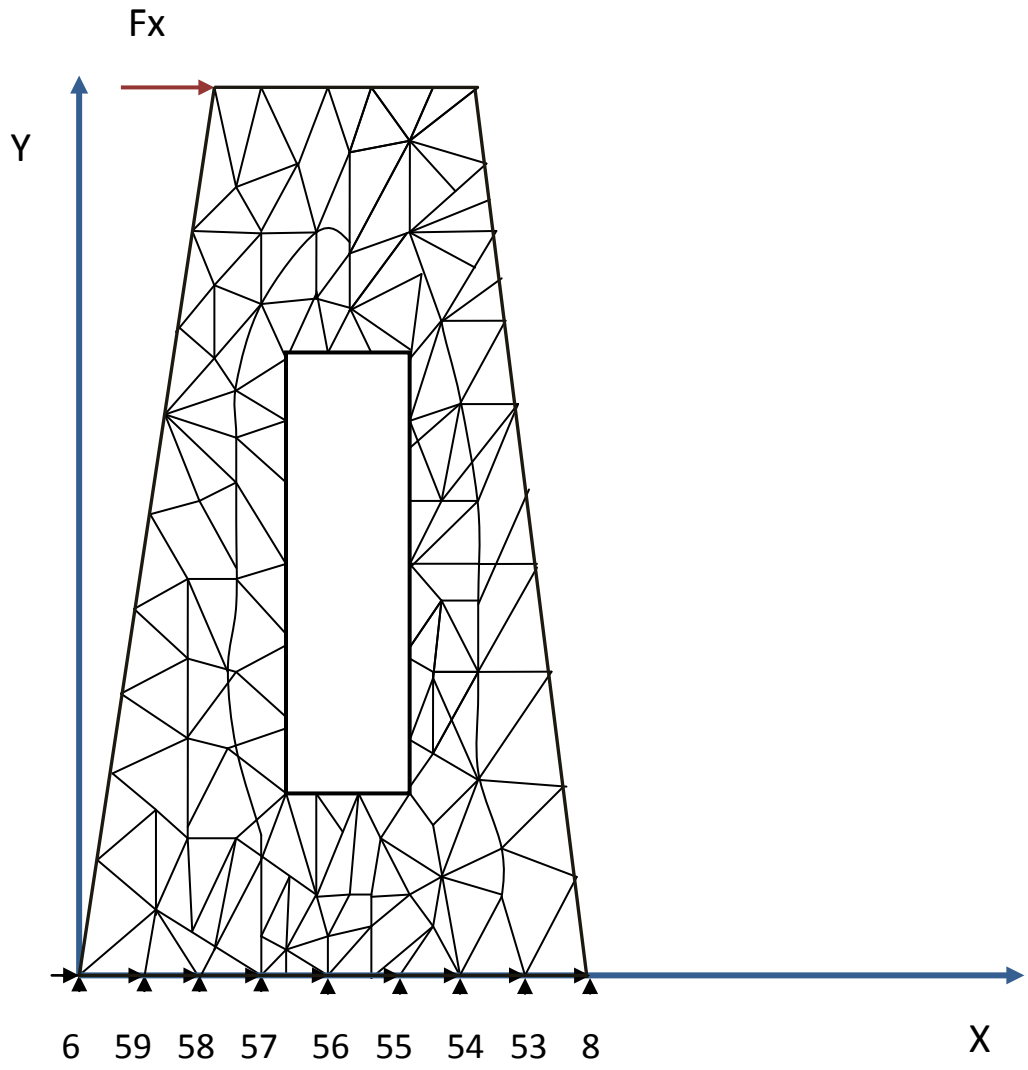


Figure 3.1 : Structure (panneau) sollicitée par une charge axiale.

Cas de charge	N° du Nœud chargé
Fx= 100N	5 35 34 36 37 7 39 41 42 44 46 48 50 51 52 19 21 22 24 26 27 30 33 32 28 99 93 80 71 95 81 18 1 12 11 10 2 13 15 17 4 93 85 87 86 105 88 63 90 69 92 97 78 73 74 60 101 40 72 26 9 60 61 82 38 43

- On aura donc : une base de données composée de (Xc ,Yc) et (σ, Xσ , Yσ).
- Cette base de données crée par (EF) est normalisée dans l'intervalle] 0,1[, puis elle est partagée en trois ensembles : Ensemble d'apprentissage, de validation et de test). Voir les tableaux ci-après.

- L'ensemble d'apprentissage sert à calculer les poids du réseau.
- L'ensemble de validation permet de vérifier les performances de généralisation du réseau.
- L'ensemble de test ne doit pas intervenir dans la détermination du reseau.il sert à évaluer les performances finales du réseau.

III.1.1.1 DONNÉES DE L'APPRENTISSAGE :

Coordonnées de la charge		Coordonnées et valeur de la contrainte		
Xc	Yc	Xσ	Yσ	σ
0.2072	0.9983	0.20723	0.9983	0.94181
0.4444	0.9983	0.94736	0.2533	0.77636
0.3256	0.9983	0.94736	0.2533	0.77818
0.5625	0.9983	0.94736	0.2533	0.77636
0.6809	0.9983	0.94736	0.2533	0.77454
0.7993	0.9983	0.94736	0.2533	0.77454
0.8240	0.8741	0.05923	0.2533	0.77454
0.8486	0.7500	0.94736	0.2533	0.57818
0.8610	0.6879	0.94736	0.2533	0.53636
0.8856	0.5637	0.94736	0.2533	0.45818
0.9103	0.4395	0.99671	0.0049	0.34545

0.9350	0.3153	0.99671	0.0049	0.28010
0.9597	0.1912	0.99671	0.0049	0.24727
0.9720	0.1291	0.97203	0.1291	0.28727
0.9843	0.0670	0.99671	0.0049	0.37818
0.1949	0.9362	0.94736	0.2533	0.72727
0.1702	0.8120	0.94736	0.2533	0.62545
0.1578	0.7501	0.05921	0.2533	0.57636
0.1332	0.6258	0.05921	0.2533	0.49818
0.1085	0.5016	0.05921	0.2533	0.39818
0.0962	0.4395	0.00986	0.0049	0.35090
0.0838	0.3774	0.00986	0.0049	0.31454
0.0592	0.2533	0.05921	0.2533	0.31818
0.0345	0.1291	0.03453	0.1291	0.28545
0.0222	0.0670	0.00986	0.0049	0.38363
0.3616	0.9288	0.94736	0.2533	0.72000
0.3902	0.8619	0.94736	0.2533	0.66545
0.6757	0.9303	0.05921	0.2533	0.72002
0.7246	0.8056	0.05921	0.2533	0.61818
0.6889	0.7278	0.94736	0.2533	0.56000
0.3217	0.7269	0.05921	0.2533	0.55636
0.5032	0.7256	0.94736	0.2533	0.55272
0.5032	0.6672	0.94736	0.2533	0.50545
0.3717	0.6672	0.05921	0.2533	0.51636
0.3717	0.4023	0.00986	0.0049	0.32909
0.3717	0.5347	0.05921	0.2533	0.42909
0.3717	0.4685	0.00986	0.0049	0.36727
0.3717	0.3360	0.00986	0.0049	0.28363
0.5032	0.3360	0.00986	0.0049	0.26909
0,6348	0.3360	0.99671	0.0049	0.28012
0.6348	0.4685	0.99671	0.0049	0.36363
0.6348	0.6009	0.94736	0.2533	0.48181
0.6348	0.6672	0.94736	0.2533	0.51818
0.7640	0.5849	0.94736	0.2533	0.47454
0.7231	0.4876	0.94736	0.2533	0.38727
0.8497	0.2570	0.99671	0.0049	0.25454
0.8250	0.3337	0.99671	0.0049	0.28545
0.6965	0.2812	0.99671	0.0049	0.25272
0.8526	0.1083	0.99671	0.0049	0.24363
0.5499	0.0761	0.50328	0.0049	0.17927

0.5700	0.2088	0.99671	0.0049	0.19454
0.4173	0.2204	0.00986	0.0049	0.20727
0.1533	0.1085	0.00986	0.0049	0.25636
0.4083	0.0691	0.37993	0.0049	0.18727
0.1811	0.3336	0.00986	0.0049	0.28909
0.1828	0.4046	0.00986	0.0049	0.32909
0.2444	0.5060	0.05921	0.2533	0.40727
0.2659	0.6477	0.05921	0.2533	0.50909
0.2736	0.8705	0.94736	0.2533	0.67454
0.1949	0.9362	0.94736	0.2533	0.72727
0.8363	0.8120	0.94736	0.2533	0.62545
0.3105	0.2820	0.00986	0.0049	0.25454
0.2533	0.5792	0.05921	0.2533	0.46909

III.1.1.2.<u>DONNÉES DE VALIDATION</u> :

Coordonnées de la charge		Coordonnées et valeur de la contrainte		
Xc	Yc	Xσ	Yσ	σ
0.4083	0.0691	0.37990	0.0049	0.18727
0.3902	0.8619	0.94734	0.2533	0.66545
0.8497	0.2570	0.99671	0.0049	0.25454
0.8222	0.4049	0.99671	0.0049	0.32545
0.7478	0.6507	0.94736	0.2533	0.51272
0.7246	0.8056	0.05921	0.2533	0.61818
0.3037	0.8043	0.94736	0.2533	0.61818
0.1828	0.4046	0.00986	0.0049	0.32909
0.3717	0.6009	0.05921	0.2533	0.48010
0.3717	0.6672	0.05921	0.2533	0.51636
0.1085	0.5016	0.05921	0.2533	0.39818
0.0559	0.2533	0.05921	0.2533	0.31818
0.9843	0.0670	0.99671	0.0049	0.37636
0.2072	0.9983	0.20723	0.9983	0.94181
0.8116	0.9362	0.05921	0.2533	0.72363
0.7993	0.9983	0.94736	0.2533	0.77454
0.8733	0.6258	0.94736	0.2533	0.50120

II.1.1.3.DONNÉES DE TEST :

Coordonnées de la charge		Coordonnées et valeur de la contrainte		
Xc	Yc	Xσ	Yσ	σ
0.3616	0.9288	0.94736	0.2533	0.7200
0.6889	0.7278	0.94736	0.2533	0.5600
0.3717	0.5347	0.05921	0.2533	0.42909
0.6348	0.4023	0.99671	0.0049	0.32545
0.3717	0.4023	0.00986	0.0049	0.32909
0.3717	0.3360	0.00986	0.0049	0.28363
0.6348	0.6672	0.94736	0.2533	0.51818
0.1949	0.9362	0.94736	0.2533	0.72727
0.1455	0.6879	0.05921	0.2533	0.53454
0.4173	0.2204	0.00986	0.0049	0.20727
0.2533	0.5792	0.05921	0.2533	0.46909
0.4083	0.0691	0.37993	0.0049	0.18727
0.8363	0.8112	0.94736	0.2533	0.62545
0.5087	0.9091	0.94736	0.2533	0.70363
0.9597	0.1912	0.99671	0.0049	0.24727
0.9473	0.2533	0.94736	0.2533	0.3200

III.1.2. CONCEPTION DU RÉSEAU :

-Le modèle utilisé pour cette application est un réseau de type perceptron multicouche à rétro propagation d'erreur, nous l'avons choisi pour sa simplicité, et sa fiabilité pour les problèmes de prédiction, d'identification et de diagnostique.

III.1.2.1 DONNÉES DU RÉSEAU :

Nous avons : **- 2 neurones** dans la couche d'entrée correspondent aux coordonnées de la charge Xc et Yc.

- **3 neurones** dans la couche de sortie correspondent à la valeur de la contrainte maximale σ et à ses coordonnées Xσ et Y σ. en d'autres termes.

- **Xc** et **Yc** : les entrées du réseau.- **σ**, **Xσ** et **Yσ** : les sorties du réseau.

- La fonction de transfert de la couche cachée et de sortie est la **sigmoïde**, puisque la rétro propagation nécessite des fonctions d'activation dérivables.
- Sur la **couche cachée**, le nombre de neurone n'obéit à aucune règle générale. On utilise la méthode constructive, on commence par une couche cachée, avec un neurone par exemple, et on ajoute progressivement des neurones sur la couche cachée, et en agissant en même temps sur **les paramètres d'apprentissage**, jusqu'à l'obtention de la précision voulue.
- Le choix des poids initiaux est aléatoire. Il faut prendre la précaution pour qu'ils ne soient pas égaux, car il ya un risque de saturation des neurones, et par conséquent un blocage de l'apprentissage.
- Le choix du coefficient d'apprentissage **η** est toujours entre 0 et 1. On choisit une grande valeur au début de l'apprentissage, et de la réduire, au fur et à mesure qu'on s'approche des minimums locaux, pour atténuer les oscillations.
- Le choix du coefficient « momentum : **α** » Sa valeur est généralement entre 0.5 et 0.9.

III-1.2.2 .LES ÉTAPES DE L'APPRENTISSAGE :

1- initialiser les poids et les biais avec des valeurs aléatoires faibles.

2- présenter le vecteur d'entrée (coordonnée de la charge), ainsi que le vecteur de sortie désiré (les coordonnées et la valeur de la contrainte).

3- calculer la sortie du réseau.

4- calculer l'erreur entre la valeur désirée et celle calculée par le réseau, ensuite l'erreur est propagé vers les neurones de la couche cachée pour calculer Le gradient des poids.

5- ajuster les poids et les biais du réseau.

6- présenter un autre vecteur et l'aller à l'étape 3.

7- fin.

Le processus d'apprentissage s'arrête, lorsque l'erreur se stabilise à un niveau acceptable. **Ou** à un nombre d'itération fixé.

III.1.2.3. ARCHITECTURE DU RÉSEAU :

La méthode de recherche de l'architecture optimale se fait de la manière suivante :
- On va entrainer le réseau avec plusieurs architectures. (Varier le nombre de couche cachée.et le nombre de neurone sur chaque couche), on garde les architectures, qui ont donné des erreurs minimums. Pour chaque architecture, on fait plusieurs initialisations des poids. On conserve l'architecture, qui a une erreur minimum, et un nombre de neurones moins.

III.1.2.3.1. UTILISATION DE LA BOITE À OUTILS MATLAB:

- Le logiciel utilisé pour la simulation est le module « NNTOOL » disponible au sein du logiciel Matlab.

Création du réseau MLP :

La création du réseau se fait par la commande de la fonction newff :

-Nnet = newff (PR. [S1 S2 S3 … SN].{FA1.FA2……FAN}.AAP.AAI.FC).

-PR: Plage des variations des entrées (représentée par minimax(p)).

-Si : Nombre de neurone dans la couche i. pour N couches.

-FAi : fonction d'activation dans la couche i. elle est dans notre cas la sigmoïde.

Elle peut être :

- Transig : Fonction Tangente sigmoïde.
- Hardlim : Fonction Heaviside.
- Pureline : Fonction linéaire.

-AAP : Représente l'algorithme d'apprentissage par paquets (les poids sont modifiés après le passage de tous les exemples). Elle peut être :

- Trainlim : apprentissage par l'algorithme Levenberg- Marquardt.
- Trainrp : apprentissage par l'algorithme résilient propagation.

- Trainscg : apprentissage par le gradient conjugué (scg).
- Traincgf : apprentissage par le gradient conjugué (cgf).
- Traingdm : apprentissage avec momentum et pas d'apprentissage contrôlés.
- Traingdx : apprentissage avec pas d'apprentissage contrôlé.

-AAI : L'algorithme d'apprentissage incrémental (les poids sont modifiés après chaque représentation d'une entrée). Il peut être :

- Learngd : l'algorithme d'apprentissage est la descente de gradient avec Pas d'apprentissage fixe.
- Learngdm : l'algorithme d'apprentissage est la descente de gradient avec pas d'apprentissage et momentum fixes.

-FC : Fonction de cout elle peut être :

- Mse : erreur quadratique moyenne.
- Mserg : erreur quadratique moyenne avec pondération des poids.
- Sse : somme des carrés des erreurs.

III.1.2.3.2. PARAMÈTRES DE PERFORMANCE DU RÉSEAU :

Apres le test de plusieurs architectures.et avec des différents algorithmes d'apprentissage disponibles. On a pu obtenir la configuration suivante :

Net = newff (Minmax(p)[Nncc Nncs].{Fac Fas}.' trainrp '.

Avec P : Les entrées (**2 neurones dans la couche d'entrée**).

- Nncc : nombre de neurone de la couche cachée (**27 neurones**).
- Nncs : nombre de neurone de la couche de sortie (**3 neurones**).
- Fac : fonction d'activation de la couche cachée (**fonction sigmoïde**).
- Fas : fonction d'activation de la couche de sortie (**fonction sigmoïde**).
- Trainrp : **l'algorithme d'apprentissage** :
- Nombre d'itération : **303**
- L'erreur de performance : **0.0014** (Figure 4.3).
- Num show : **25** ; - Erreur désirée : **1e-3** ; Min_gr : **1e-6.**

- Delt_inc : **1.2** ; - Delt_ dec : **0.5** ; - Delta : **0.07** ; - Delta max : **50**
- Biais de la couche caché est une matrice colonne de **27 valeurs**.
- Biais de la couche de sortie est une matrice colonne de **3 valeurs**.
- Matrice des poids (entrée – couche caché) **(2x27)**.
- Matrice des poids (couche caché – sortie) **(3x27)**.

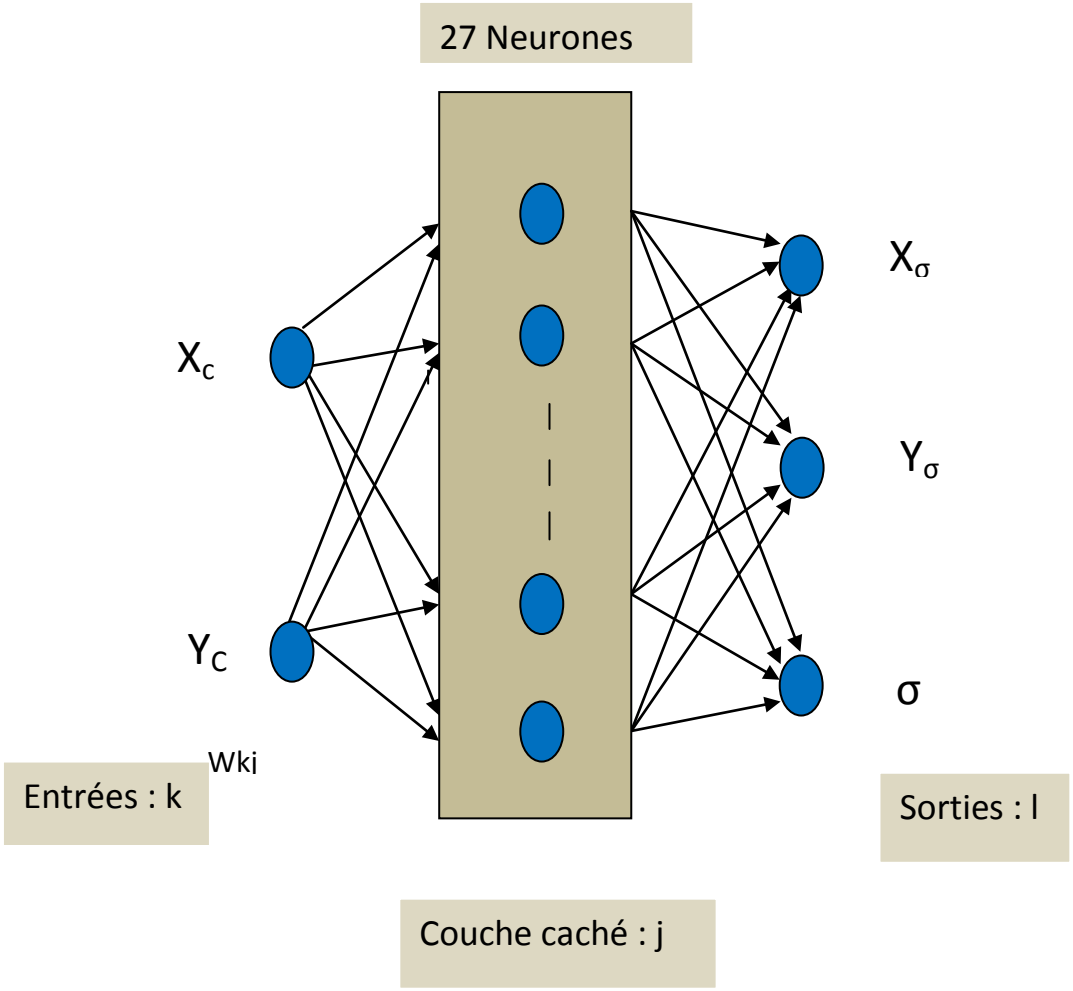


Figure 3.2 : Architecture du réseau

III.1.2.3.3. GRAPHE DE L'ERREUR :

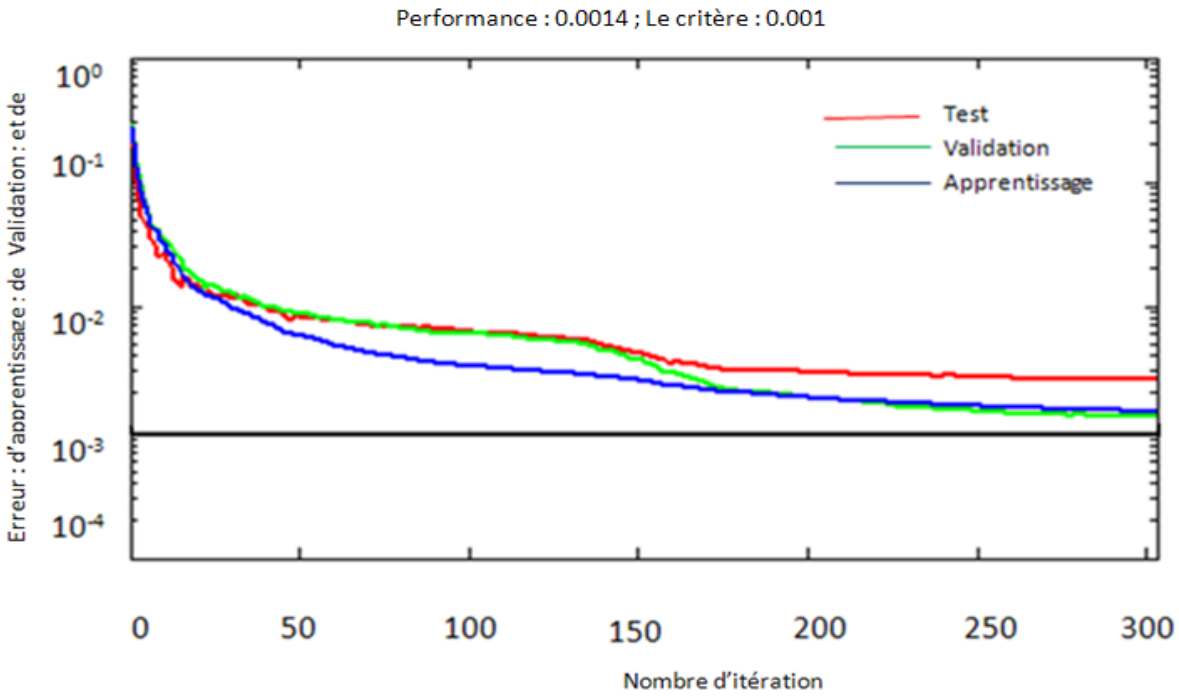


Figure 3.3 : graphes de l'erreur (d'apprentissage. de test et de validation).

III.1.2.3.4 COURBES DE RÉGRESSION :

Les courbes de regression permettent de valider la performance du reseau construit.Les valeurs en ordonnées représentent les sorties du reseau , pour les entrées de (l'apprentissage.de validation,de test et pour l'ensemble). Les valeurs en abscisses représentent les valeurs desirées.

- Les cercles en noir representent les valeurs desirées.
- Les droites continues representent l'approximation faite par le reseau.
- Les droites en pointillé representent la parfaite approximation.

On constate que : - Les données de l'apprentissage sont classées à 95%.

- Les données de validation sont classées à 96%.
- Les données de test sont classées à 89%.
- Globalement les données sont classées à 94%.

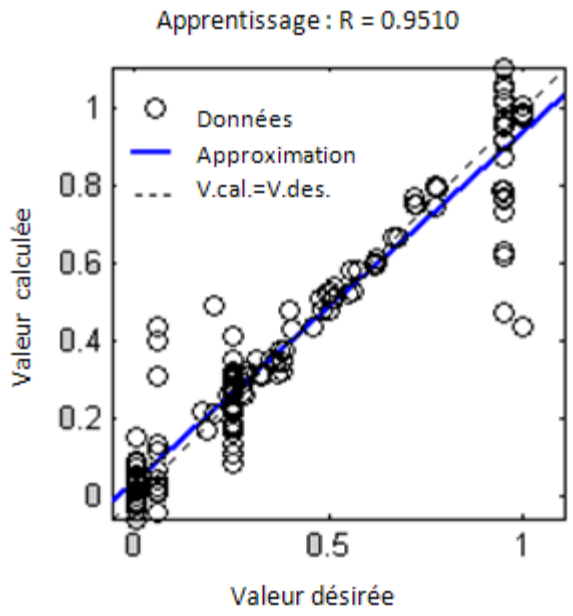


Figure 3.4 : Courbe de régression (Apprentissage)

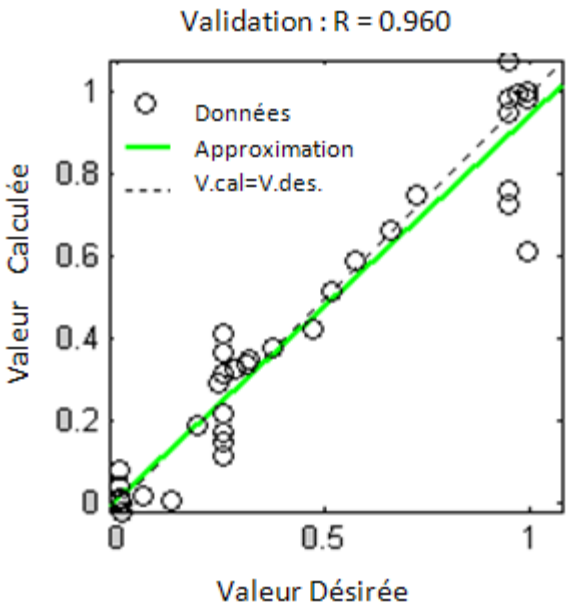


Figure 3.5 : Courbe de régression (Validation)

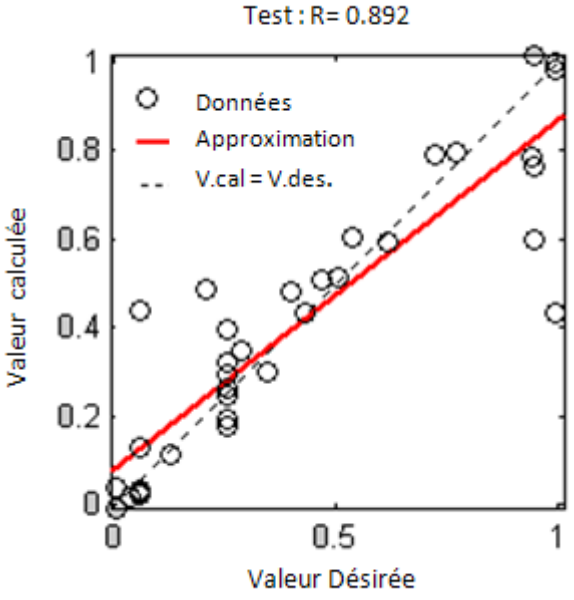


Figure 3.6 : Courbe de régression (Test)

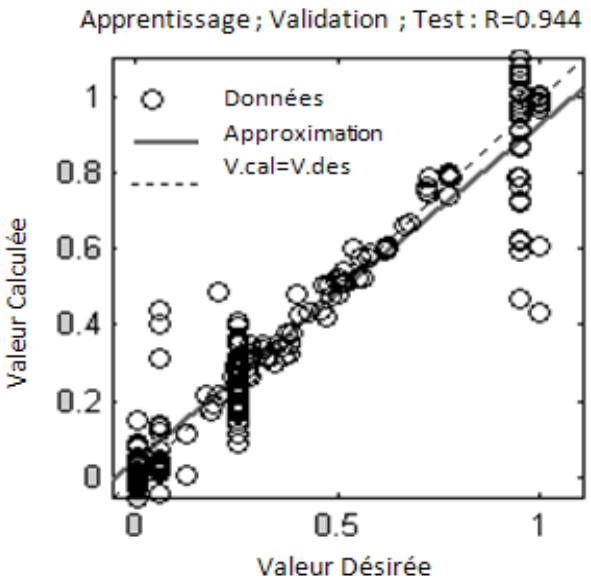


Figure 3.7 : Courbe de régression (l'ensemble)

TABLEAU DES POIDS ET BIAIS :

B1{27}	W[2 .27]	$W[27 .3]^T$	B2{3}
$\begin{Bmatrix} 1.78 \\ -33.94 \\ 3.78 \\ -21.54 \\ 16.41 \\ -27.65 \\ -16.96 \\ 26.96 \\ 10.39 \\ 25.05 \\ 2.4 \\ 22.62 \\ 21.62 \\ 13.62 \\ 7.08 \\ 18.03 \\ 17.25 \\ 15.58 \\ -7.54 \\ 13.26 \\ 10.47 \\ -18.32 \\ 6.98 \\ -9.36 \\ 6.61 \\ 22.8 \\ -1.18 \end{Bmatrix}$	$\begin{bmatrix} -16.65 & 159.24 \\ 21.91 & 22.53 \\ -13.08 & 27.58 \\ 27.64 & -6.30 \\ -27.63 & 12.90 \\ 28.80 & 10.53 \\ 28.15 & -9.29 \\ -24.92 & -16.92 \\ -26.11 & 13.01 \\ -23.32 & -18.20 \\ -26.86 & 17.80 \\ -22.44 & -18.62 \\ -24.07 & -18.77 \\ -26.64 & 4.45 \\ 13.75 & -30.58 \\ -17.00 & -18.47 \\ -0.33 & -31.14 \\ -21.16 & -13.94 \\ 14.02 & 5.80 \\ -4.81 & -25.04 \\ -10.75 & -40.60 \\ -9.57 & 27.18 \\ -0.79 & -34.46 \\ 23.34 & 17.79 \\ -4.17 & -37.00 \\ -18.06 & 23.41 \\ 14.16 & -10.15 \end{bmatrix}$	$\begin{bmatrix} -1.23 & -0.02 & 0.14 \\ -1.30 & -2.02 & 0.89 \\ -0.28 & 0.91 & 1.49 \\ -1.67 & 1.15 & -0.79 \\ 0.52 & 0.43 & 0.69 \\ -1.71 & 1.02 & -1.27 \\ 0.53 & 1.88 & 0.95 \\ -0.63 & -1.08 & 0.01 \\ 0.41 & -0.96 & 0.84 \\ 0.61 & 0.59 & 0.17 \\ 1.37 & 0.09 & 1.46 \\ -1.67 & 1.29 & -1.51 \\ 0.01 & 0.44 & -0.45 \\ -1.40 & 1.21 & -2.07 \\ 12.3 & 0.10 & 1.65 \\ -1.24 & 1.01 & 0.62 \\ -0.17 & -0.92 & -0.43 \\ 0.26 & 2.11 & -1.18 \\ 1.06 & -0.41 & -1.51 \\ 1.19 & -2.02 & 1.86 \\ -1.11 & 0.47 & -0.23 \\ 1.51 & 0.09 & 0.19 \\ 0.92 & -0.27 & -1.21 \\ -1.67 & -1.18 & 1.41 \\ 0.51 & -1.77 & 0.21 \\ 1.10 & -1.28 & -0.11 \\ -1.53 & 0.16 & -1.88 \end{bmatrix}^T$	$\begin{Bmatrix} 5.08 \\ -0.52 \\ 2.94 \end{Bmatrix}$

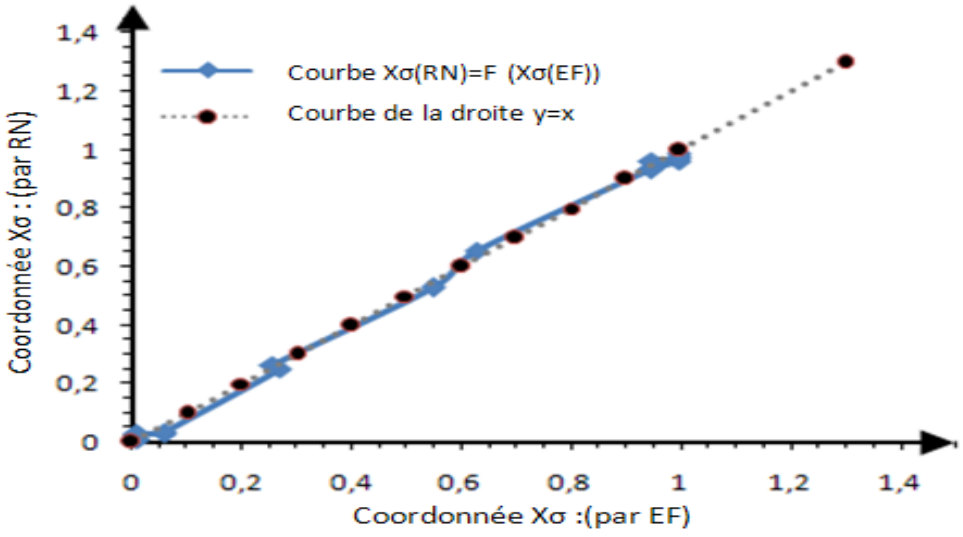
III.1.3 COMPARAISON DES RÉSULTATS :

Apres la détermination des caractéristiques du réseau, nous présentons au réseau des nouveaux exemples. Pour cela, on utilise la fonction simulation définit comme suit : a = Sim (net. p). p : représente les entrées des nouveaux exemples ; a : les sorties calculées par le réseau.

Nous présentons ci-après le tableau renfermant les valeurs calculées par le réseau par la fonction Sim (net. p), et celles obtenues par la méthode des éléments finis.
En comparant les résultats, nous pouvons dire qu'ils sont satisfaisants pour la prédiction de la magnitude de la contrainte maximale. Et assez bon, on ce qui concerne la prédiction de sa localisation, surtout pour les nœuds qui ont une ordonnée nulle. à cause de la normalisation faite, puisque La fonction de transfert sigmoïde est bornée entre 0 et 1 ; lors de la normalisation, les valeurs qui sont proches des asymptotes (borne 0 et 1).ne participent pas efficacement à l'ajustement des poids, ce qui rend la tache de l'apprentissage difficile, et par conséquent une généralisation mauvaise dans ces zones.

TABLEAU DES RÉSULTATS :

Contrainte de Von mises : Coordonnées et Valeur						
Nœud chargé	Résultats obtenus par (EF)			Résultats obtenus par (RN)		
	Xc	Yc	Contrainte Von mises	Xσ	Yσ	Contrainte Von mises
66	0.9967	0.00496	0.34363	0.9612	0.0080	0.3273
67	0.9967	0.00496	0.30181	0.9723	0.0087	0.2960
68	0.9474	0.25331	0.61818	0.9312	0.2573	0.6406
70	0.0098	0.00496	0.30545	0.0180	0.0082	0.2679
76	0.0098	0.00496	0.34909	0.0068	0.0017	0.3264
75	0.0592	0.25331	0.46909	0.0301	0.2719	0.4837
77	0.0098	0.00496	0.36181	0.0072	0.0064	0.3617
79	0.0098	0.00496	0.25636	0.0213	0.0076	0.2825
91	0.0098	0.00496	0.21818	0.0049	0.0096	0.2698
89	0.9967	0.00496	0.21636	0.9861	0.0360	0.2803
84	0.9474	0.25331	0.43272	0.9620	0.3460	0.4379
96	0.0592	0.25331	0.66727	0.0345	0.2916	0.7104
98	0.6266	0.00496	0.17545	0.6538	0.0207	0.2041
103	0.0098	0.00496	0.22727	0.0253	0.0043	0.2314
104	0.5499	0.07614	0.14747	0.5284	0.0332	0.2044
106	0.2683	0.06133	0.16763	0.2519	0.0558	0.1847
94	0.0098	0.00496	0.15072	0.0059	0.0085	0.1770
100	0.2566	0.00496	0.24121	0.2618	0.0010	0.2690



Figue 3.8 : Courbe de la coordonnée Xσ

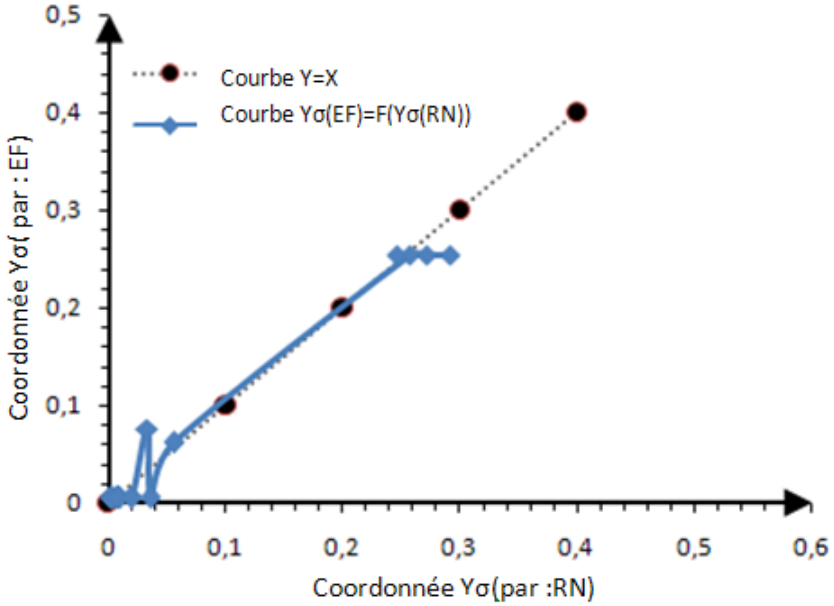


Figure 3.9 : Courbe de la coordonnée Yσ

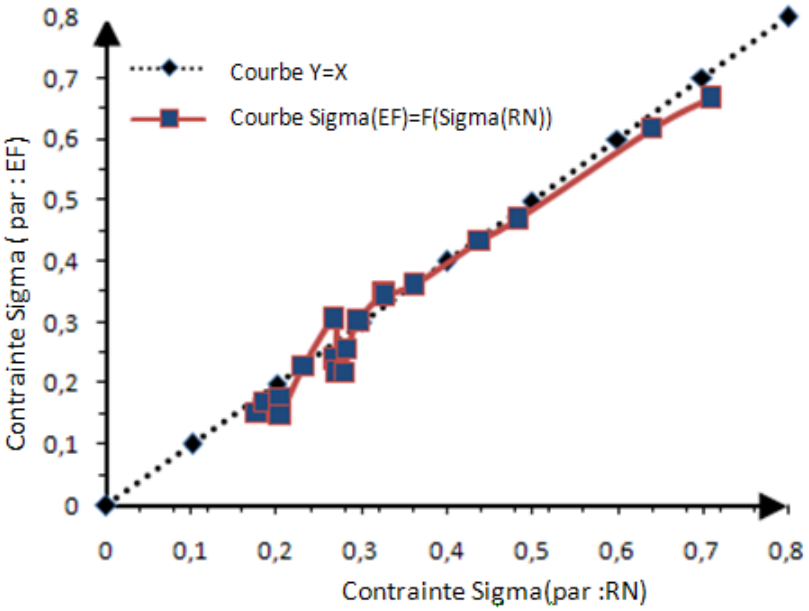


Figure 3.10 : Courbe de la contrainte maximale (σ)

III.2. APPLICATION N°2 :

Le système de contrôle pour détecter les défauts d'un arbre d'une machine générés par des vibrations, ou le mauvais fonctionnement de certains organes a toujours une réelle importance chez les industriels, vu la place signifiante qu'occupe un tel organe dans le processus de production. Actuellement les spécialistes étudient des procédures pour détecter ces défauts et minimisent ainsi les dégâts qui peuvent se produire. Parmi ces procédures l'emploi des systèmes de contrôle utilisant les réseaux de neurones exploités sous forme de circuit intégré. Selon les lois de la physique l'action d'une force à toujours son effet les paramètres (déplacements. contraintes et flèches), l'inverse est aussi vrai.

En effet, la position des excitations, nous donne toujours des renseignements sur la nature et la position du défaut que présente l'arbre, par exemple une faible déformation, provoque des déséquilibres, et des voilements, ou une diminution de la fréquence naturelle.

Dans cette application nous essayons d'assimiler cet arbre, à une poutre et d'appliquer le modèle de la rétro propagation pour déterminer la position de l'excitation sur cette poutre, connaissant la flèche d'un point fixe (pris ici au milieu de la poutre).

Comme cette méthode nécessite une base de données obtenue. Soit à l'aide des mesures expérimentales, soit avec des méthodes classiques, pour l'utiliser à réaliser l'apprentissage du réseau afin qu'il apprenne le phénomène, qu'il devra le contrôler.

Nous employons ici, la méthode des éléments finis, pour générer une base de données par la procédure suivante :

On fait varier la position d'une charge donnée, et on calcule à chaque fois la flèche correspondante d'un point donnée de la poutre (ici on choisit le milieu de la poutre).

III.2.1. DONNÉES DU PROBLÈME :

Nous prenons pour cet exemple, une poutre encastrée d'un coté et libre de l'autre bout, de section circulaire, ne nous tenons compte ici que des excitations statiques.

Les données de la poutre sont : longueur L=1m, diamètre 90mm, module de Young E =2.1x10^5 N.mm^{-2}, densité = 7800kg/m^3, ν =0.3 et la charge Fy=200N.

- D'une manière similaire à l'application N°1, la base de données par élément finis est obtenue par discrétisation de la poutre en 20 nœuds et 19 éléments.
- on applique à chaque fois, la charge de 200N dans un nœud choisi, et on calcule la flèche, au point du milieu de la poutre (Xm) , on obtient ainsi un couple de données, l'abscisse (Xc) de la charge, et la flèche Xm.
  - Les données sont regroupées dans le tableau ci-après.
  - L'ensemble des exemples est partagé en trois parties (apprentissage, validation et test)
  - La première, sert pour faire l'apprentissage (détermination des poids).
  - La deuxième, de validation pour vérifier la capacité de généralisation du réseau, et éviter par conséquent le réseau d'apprendre trop : (phénomène de sur-apprentissage).
  - La dernière, de test sert à évaluer les performances finales du réseau.
  - **Observation** : les valeurs des données (apprentissage, de validation, de test et celles utilisées pour comparaison) sont normalisées.

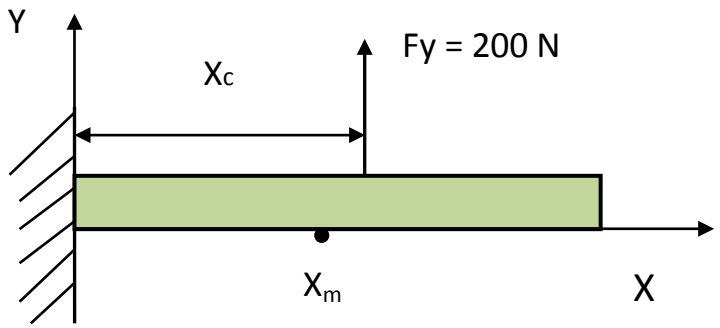


Figure 3.11 : Poutre encastrée

TABLEAU DES DONNÉES :

Charge	Nœuds chargés	Coordonnée Xc (mm)	La flèche : Fm(mm)
Fy=200N	3	50	0.00017
	4	100	0.00069
	5	150	0.00192
	6	200	0.00256
	7	250	0.00386
	8	300	0.00532
	9	350	0.00694
	10	400	0.00867
	11	450	0.00898
	12	500	0.01232
	13	550	0.01417
	14	600	0.01610
	15	650	0.01781
	16	700	0.01971
	17	750	0.02152
	18	800	0.02341
	19	850	0.02523
	20	900	0.02714
	21	950	0.02895

III.2.1.1. DONNÉES D'APPRENTISSAGE :

Fleche : Fm (mm)	Coordonnée : Xc (mm)
0.0059	0.1527
0.0658	0.1523
0.1273	0.2519
0.2290	0.3515
0.2861	0.4013
0.4065	0.5009
0.5281	0.6009
0.6501	0.7001
0.7722	0.7998
0.8943	0.8994
0.9537	0.9492
1.0057	0.9992
0.5874	0.6500
0.0844	0.2021
0.8316	0.8496

III.2.1.2. DONNÉES DE VALIDATION :

Fleche : Fm	Coordonnée : Xc
0.0227	0.1025
0.1273	0.2519
0.2963	0.4511
0.4065	0.5009
0.8316	0.8496
0.9813	0.9991
0.1755	0.3017
0.0658	0.1523

III.2.1.3. DONNÉES DE TEST :

Fleche : Fm	Coordonnée : Xc
0.0059	0.1527
0.1755	0.3017
0.4676	0.5507
0.8316	0.8496
0.5874	0.6503
0.8943	0.8994
0.9537	0.9492
0.0227	0.1025
0.0658	0.1523

III.2.2. CONCEPTION DU RÉSEAU :

III.2.2.1.DONNÉES DU RÉSEAU :

Le réseau que nous utilisons est le perceptron multicouche à rétro propagation d'erreur avec apprentissage supervisé ses données sont :

- Les entrées : la flèche du milieu de la poutre **Fm**.
- Les sorties : l'abscisse de la charge d'excitation **Xc**.

III.2.2.2 L'ARCHITECTURE DU RÉSEAU :

La recherche de l'architecture optimale consiste à chercher des minimums locaux sur la surface de l'erreur, dans l'espace des poids et de choisir le meilleur minimum répondant au critère, que nous avons choisi au préalable.

Il n'est pas nécessaire de chercher un minimum absolu, car il est généralement couteux économiquement, surtout lorsque les exigences techniques ne demandent pas des données très précises, en plus de ça les données utilisées pour l'apprentissage sont toujours entachées de bruit. Il est illusoire de chercher une erreur ne correspond pas à l'ordre de grandeur de la précision de ces données.

-**Pratiquement** l'architecture se construit de la façon suivante :

On part d'une architecture, 2 neurones d'entrée, 2 sorties et un neurone dans la couche cachée (appelé méthode ascendante). On met le processus de l'apprentissage en marche, on fait plusieurs initialisations des poids, en agissant en même temps sur les paramètres (**α** et **η**), si une des architectures a satisfait l'erreur désirée, on arrête l'apprentissage, sinon on ajoute un neurone à la couche cachée, et on répète la même procédure, jusqu'à ce que nous arrivons à une architecture satisfaisante.

-La fonction de transfert utilisée est la sigmoïde, puisque le modèle de rétro Propagation nécessite une fonction dérivable et continue.

-Les données sont normalisées ente 0 et 1.

Certains experts préconisent de les normaliser dans l'intervalle [0.2. 0.8] pour éviter la saturation aux asymptotes 0 et 1.

-Les biais et les poids sont initialisés aléatoirement et à des petites valeurs.- Les tests sont faits à l'aide de 'nntool ', le module (neural network Tools), disponible au sein du logiciel Matlab.

-Après le test de plusieurs architectures, rappelons-nous, que ce test doit être fondé sur le savoir faire :

-On a choisi l'architecture suivante :

Une couche d'entrée avec **1 neurone**.

Une couche cachée avec **8 neurones**.

Une couche de sortie avec **1 neurone**.

La fonction d'apprentissage ' **traingdm** 'puisqu'elle est efficace lorsqu'on ne dispose pas d'une base de données riche.

Les paramètres finals :

- Nombre d'itération : **10000**
- Pas d'apprentissage : **0.05**
- Max Fail : **5**
- Mc : **0.9**
- Min_grad : **1e-10**
- Show : **25 ;** Erreur désirée **: 0.001**
- Erreur : **0.0044** (Figure : 3.8) ; - Biais et poids (voir tableau ci-après).

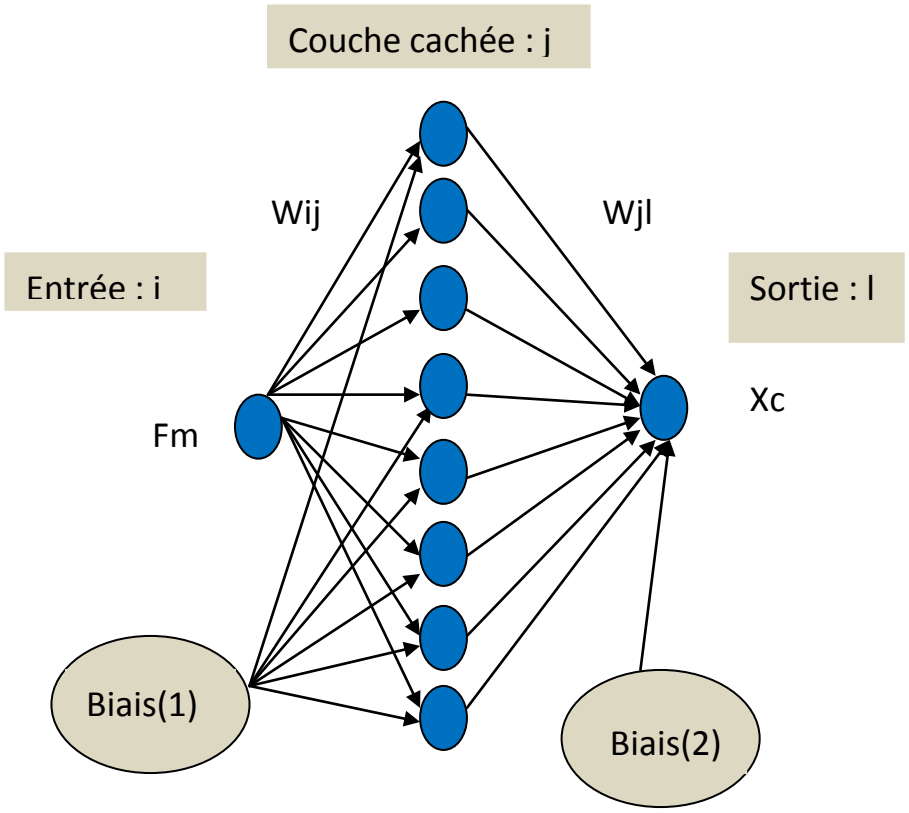


Figure 3.12 : Architecture du réseau

TABLEAU DES POIDS ET BIAIS :

B1 {8}	W [8]	W [8]^T	B2 {1}
-49.90 44.46 -37.99 -31.41 25.56 19.36 12.52 -6.35	50.88 -50.07 50.25 50.44 -50.02 - 50.41 - 50.38 50.40	2.000 0.852 1.933 -0.472 -1.96 0.691 -0.042 1.720 T	-2.296

III.2.2.3. LE GRAPHE DE L'ERREUR :

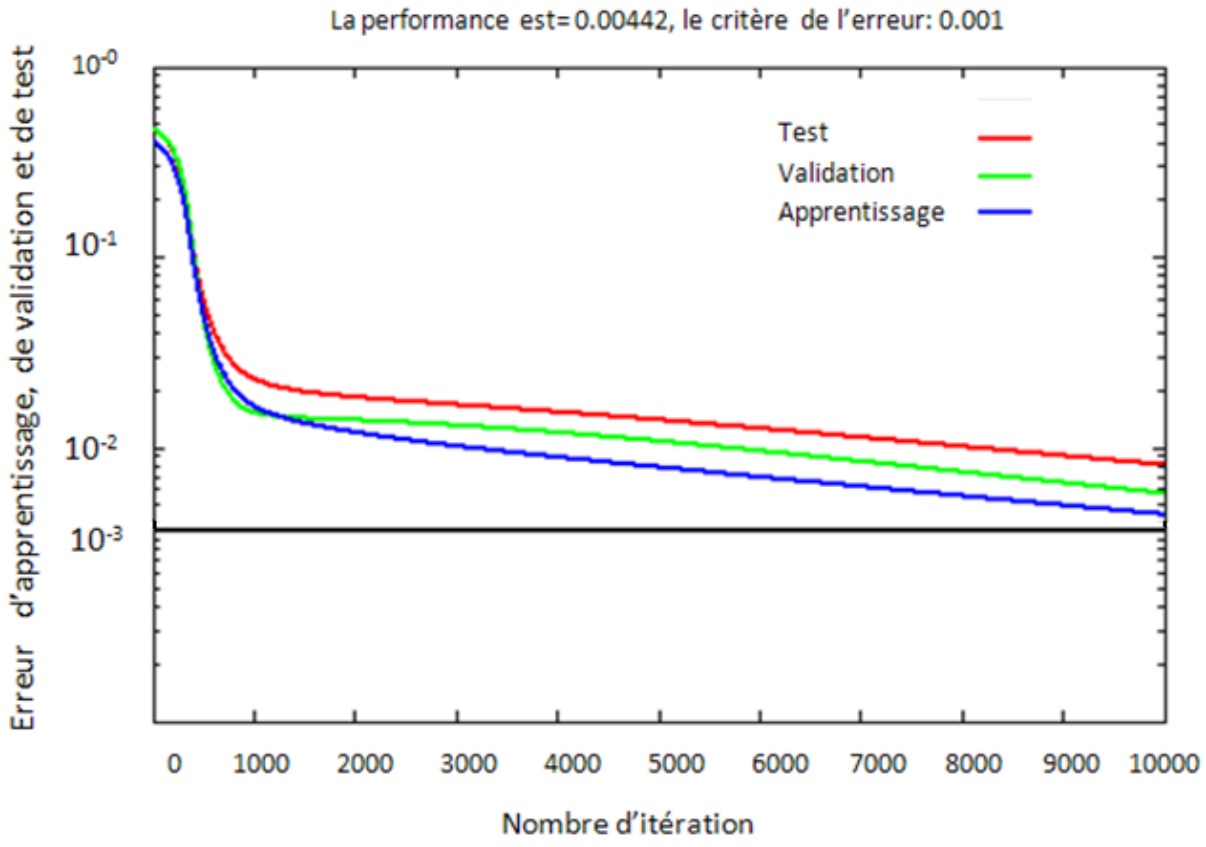


Figure 3.13 : Erreur moyenne sur : l'ensemble de (test. validation et apprentissage)

III.2.2.4 COURBES DE RÉGRESSION :

Ces courbes définissent Le rapport entre les valeurs désirées en abscisses et les valeurs calculées par le réseau en ordonnées. **On constate que** :

- Toutes les données sont concentrées sur la droite en pointillées avec un taux de classification de 99%. Ce qui signifie que la tache de l'apprentissage a réussi. La performance du réseau est évaluée à 92%.

I

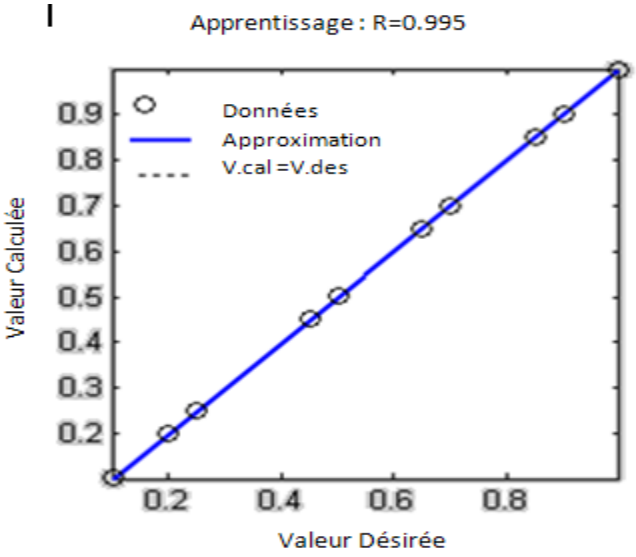


Figure 3.14 : Courbe de régression(Apprentissage)

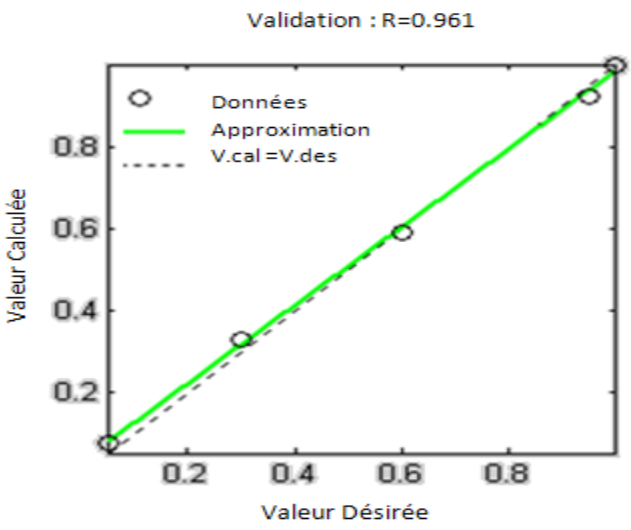


Figure 3.15 : Courbe de régression(Validation)

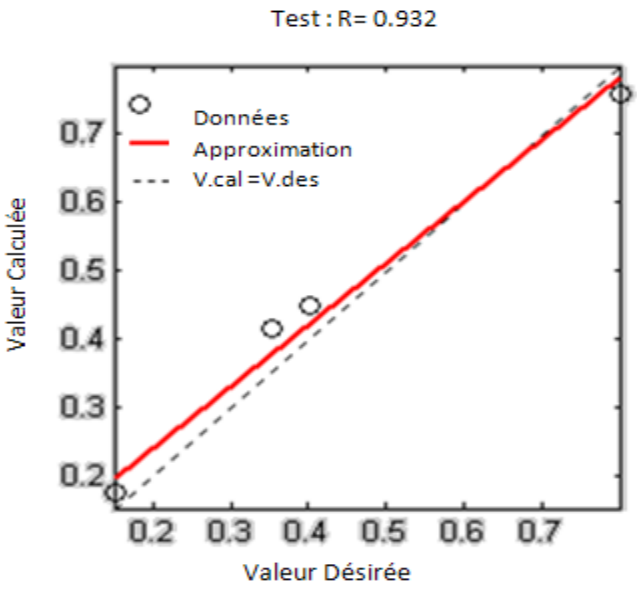


Figure 3.16 : Courbe de régression(Test)

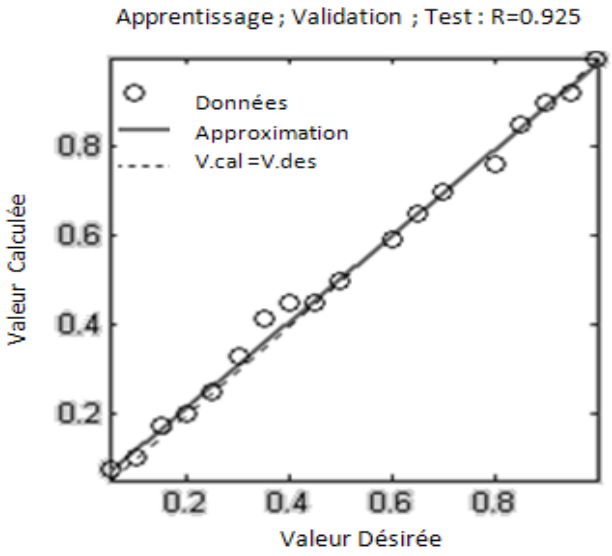


Figure 3.17 : Courbe de régression (l'ensemble)

III.2.3. COMPARISON DES RÉSULTATS :

Coordonnée : Xc (E.F)	Coordonnée : Xc (R.N)
0.0976	0.0991
0.2719	0.2890
0.4213	0.4481
0.6205	0.6681
0.8695	0.8325
0.9741	0.9687

- Le but est de prédire la position de l'excitation d'une poutre, en mesurant la flèche.
- Les valeurs que nous avons obtenues sont acceptables, donc le réseau a pu généraliser facilement la relation existante entre les deux grandeurs en question.

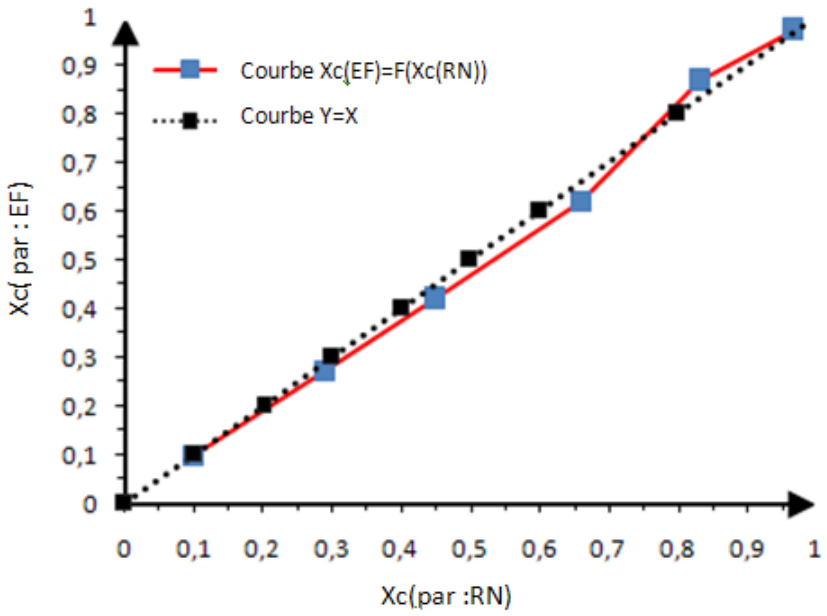


Figure 3.18 : Courbe de la coordonnée Xc

III.3.APPLICATION N°3 :

Une structure endommagée, se manifeste toujours par un changement de la réponse de cette structure. Une réduction du module de rigidité d'un élément de la structure par exemple, engendre une variation (des déplacements statiques ou des propriétés dynamiques te que : les fréquences naturelles).

Nous présentons dans cet exemple, la méthode des réseaux de neurones pour identifier l'endommagement (sa présence, sa localisation et sa sévérité) dans cette structure.

Nous exploitons ce concept pour prédire la dégradation du module de rigidité de la structure (portique) sollicitée, en mesurant avec des capteurs les fréquences naturelles.

III.3.1.DONNÉES DU PROBLÈME :

Un portique composé de 5 poutres rigides entre elles : - section de l'élément N°1 : 22x12 cm^2 ; - section de l'élément N°2 : 10x12 cm^2. - section de l'élément N°3 : 16x12 cm^2 ; - section de l'élément N°4 : 14x12 cm^2. - section de l'élément N°5 : 18x12 cm^2 ; - Longueur des poutres = 3m.- ν = 0.3, E =2.1x10^5 N.mm^{-2}, p=7800kg/m^2, F=3000N

-L'objectif dans cette application est d'identifier l'endommagement (sa présence et sa sévérité), dans un portique sous sollicitation (Figure3.19) en utilisant la méthode des réseaux de neurones.

Cette méthode consiste à construire un réseau, dont ses entrées sont les fréquences naturelles et les sorties sont les modules de rigidité des cinq poutres. En utilisant une base de données servant à apprendre le phénomène reliant les entrées aux sorties. Après une phase d'apprentissage, le réseau sera capable de déterminer les valeurs des modules de rigidité des cinq poutres pour des mesures données (des fréquences naturelles ou des déplacements ;….).

La base de données est obtenue en variant le module de rigidité de chaque élément (poutre), de 0.95E à 0.1E par pas de 0.05. On obtient 18 scenarios d'endommagement pour chaque élément poutre.

-90 états d'endommagement seront utilisés pour faire l'apprentissage.

Chaque état d'endommagement est employé avec le code (élément fini). Pour donner les fréquences naturelles correspondantes.

Nous obtenons ainsi les données qui sont, après normalisation, partagées en (données d'apprentissage. de validation et données de test).

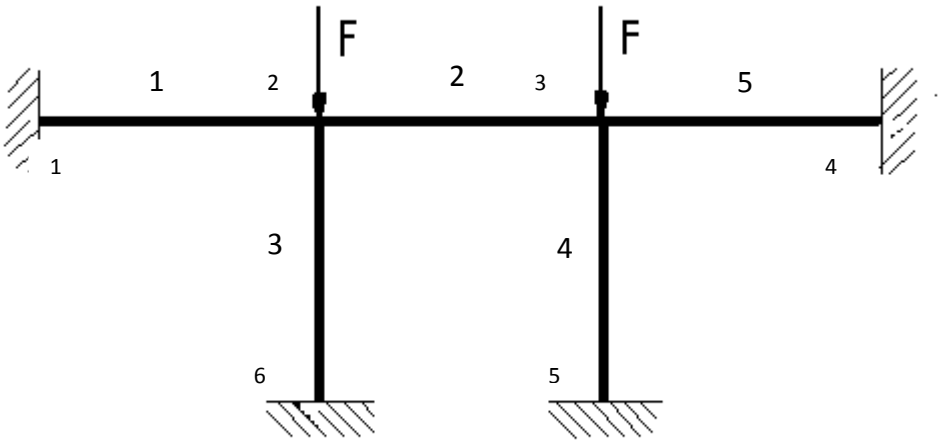


Figure 3.19 : Portique de 5 poutres

III.3.1.1 DONNÉES DE L'APPRENTISSAGE :

Fréquences naturelles (entrées)				Module de rigidité (sorties)				
F1	F2	F3	F4	E1	E2	E3	E4	E5
0.877	0.883	0.869	0.940	0.89	0.95	0.95	0.95	0.95
0.867	0.856	0.672	0.940	0.68	0.95	0.95	0.95	0.95
0.851	0.783	0.450	0.940	0.47	0.95	0.95	0.95	0.95
0.818	0.564	0.287	0.608	0.26	0.95	0.95	0.95	0.95
0.743	0.266	0.195	0.291	0.11	0.95	0.95	0.95	0.95
0.861	0.873	0.892	0.937	0.95	0.89	0.95	0.95	0.95
0.776	0.817	0.824	0.926	0.95	0.68	0.95	0.95	0.95
0.663	0.766	0.766	0.915	0.95	0.47	0.95	0.95	0.95
0.510	0.720	0.715	0.903	0.95	0.26	0.95	0.95	0.95
0.351	0.684	0.665	0.837	0.95	0.11	0.95	0.95	0.95
0.878	0.884	0.830	0.941	0.95	0.95	0.89	0.95	0.95
0.874	0.851	0.472	0.938	0.95	0.95	0.68	0.95	0.95
0.867	0.648	0.251	0.934	0.95	0.95	0.47	0.95	0.95
0.835	0.247	0.208	0.925	0.95	0.95	0.26	0.95	0.95
0.650	0.126	0.194	0.895	0.95	0.95	0.11	0.95	0.95
0.876	0.849	0.906	0.893	0.95	0.95	0.95	0.89	0.95
0.862	0.663	0.885	0.711	0.95	0.95	0.95	0.68	0.95
0.831	0.425	0.857	0.551	0.95	0.95	0.95	0.47	0.95
0.727	0.215	0.816	0.422	0.95	0.95	0.95	0.26	0.95
0.543	0.140	0.765	0.346	0.95	0.95	0.95	0.11	0.95
0.875	0.868	0.911	0.872	0.95	0.95	0.95	0.95	0.89
0.852	0.775	0.913	0.5845	0.95	0.95	0.95	0.95	0.68
0.816	0.640	0.902	0.282	0.95	0.95	0.95	0.95	0.47
0.748	0.424	0.595	0.190	0.95	0.95	0.95	0.95	0.26
0.632	0.185	0.311	0.189	0.95	0.95	0.95	0.95	0.11
0.453	0.016	0.150	0.188	0.95	0.95	0.95	0.95	0.005
0.875	0.878	0.823	0.940	0.84	0.95	0.95	0.95	0.95
0.864	0.845	0.617	0.940	0.63	0.95	0.95	0.95	0.95
0.829	0.638	0.327	0.711	0.32	0.95	0.95	0.95	0.95
0.751	0.804	0.808	0.923	0.95	0.63	0.95	0.95	0.95
0.553	0.731	0.727	0.906	0.95	0.32	0.95	0.95	0.95
0.873	0.827	0.389	0.937	0.95	0.95	0.63	0.95	0.95
0.865	0.557	0.235	0.933	0.95	0.95	0.42	0.95	0.95
0.853	0.349	0.215	0.928	0.95	0.95	0.32	0.95	0.95

0.856	0.608	0.879	0.668	0.95	0.95	0.95	0.63	0.95
0.844	0.746	0.909	0.509	0.95	0.95	0.95	0.95	0.63
0.770	0.493	0.699	0.192	0.95	0.95	0.95	0.95	0.32
0.571	0.071	0.107	0.092	0.005	0.95	0.95	0.95	0.95
0.421	0.109	0.185	0.781	0.95	0.95	0.005	0.95	0.95
0.766	0.255	0.828	0.451	0.95	0.95	0.95	0.32	0.95
0.201	0.592	0.125	0.073	0.95	0.005	0.95	0.95	0.95
0.345	0.108	0.692	0.283	0.95	0.95	0.95	0.005	0.95
0.866	0.715	0.891	0.755	0.95	0.95	0.95	0.74	0.95
0.766	0.255	0.828	0.451	0.95	0.95	0.95	0.32	0.95
0.871	0.786	0.322	0.936	0.95	0.95	0.58	0.95	0.95
0.798	0.176	0.204	0.923	0.95	0.95	0.21	0.95	0.95
0.873	0.763	0.896	0.821	0.95	0.95	0.95	0.79	0.95
0.795	0.305	0.844	0.482	0.95	0.95	0.95	0.37	0.95

III.3.1.2 DONNÉES DE VALIDATION :

Fréquences naturelles (entrées)				Module de rigidité (sorties)				
F1	F2	F3	F4	E1	E2	E3	E4	E5
0.873	0.873	0.775	0.941	0.79	0.95	0.95	0.95	0.95
0.571	0.071	0.108	0.092	0.005	0.95	0.95	0.95	0.95
0.724	0.791	0.794	0.921	0.95	0.58	0.95	0.95	0.95
0.463	0.709	0.701	0.898	0.95	0.22	0.95	0.95	0.95
0.876	0.874	0.657	0.939	0.95	0.95	0.79	0.95	0.95
0.860	0.456	0.223	0.931	0.95	0.95	0.37	0.95	0.95
0.421	0.109	0.185	0.781	0.95	0.95	0.005	0.95	0.95
0.851	0.553	0.872	0.627	0.95	0.95	0.95	0.58	0.95
0.677	0.183	0.802	0.395	0.95	0.95	0.95	0.21	0.95
0.864	0.825	0.911	0.731	0.95	0.95	0.95	0.95	0.79
0.788	0.546	0.802	0.196	0.95	0.95	0.95	0.95	0.37
0.870	0.866	0.725	0.943	0.74	0.95	0.95	0.95	0.95
0.842	0.859	0.874	0.934	0.95	0.84	0.95	0.95	0.95
0.629	0.754	0.753	0.912	0.95	0.42	0.95	0.95	0.95
0.872	0.726	0.278	0.935	0.95	0.95	0.53	0.95	0.95
0.866	0.715	0.890	0.755	0.95	0.95	0.95	0.74	0.95
0.766	0.255	0.828	0.451	0.95	0.95	0.95	0.32	0.95

III.3.1.3. DONNÉES DE TEST :

Fréquences naturelles (entrées)				Modules de rigidité (sorties)				
F1	F2	F3	F4	E1	E2	E3	E4	E5
0.838	0.698	0.357	0.812	0.37	0.95	0.95	0.95	0.95
0.822	0.844	0.856	0.931	0.95	0.79	0.95	0.95	0.95
0.593	0.743	0.741	0.909	0.95	0.37	0.95	0.95	0.95
0.201	0.591	0.125	0.073	0.95	0.005	0.95	0.95	0.95
0.871	0.786	0.322	0.936	0.95	0.95	0.58	0.95	0.95
0.798	0.176	0.203	0.921	0.95	0.95	0.21	0.95	0.95
0.873	0.763	0.896	0.801	0.95	0.95	0.95	0.79	0.95
0.795	0.305	0.845	0.482	0.95	0.95	0.95	0.37	0.95
0.345	0.108	0.692	0.283	0.95	0.95	0.95	0.005	0.95
0.836	0.714	0.908	0.434	0.95	0.95	0.95	0.95	0.58
0.845	0.746	0.401	0.911	0.42	0.95	0.95	0.95	0.95
0.695	0.778	0.782	0.917	0.95	0.53	0.95	0.95	0.95
0.877	0.881	0.745	0.939	0.95	0.95	0.84	0.95	0.95
0.873	0.808	0.901	0.846	0.95	0.95	0.95	0.84	0.95
0.816	0.363	0.849	0.516	0.95	0.95	0.95	0.42	0.95
0.858	0.801	0.911	0.657	0.95	0.95	0.95	0.95	0.74
0.826	0.682	0.906	0.357	0.95	0.95	0.95	0.95	0.53

III.3.2.CONCEPTION DU RÉSEAU :

III.3.2.1.DONNÉES DU RÉSEAU :

-Le réseau utilisé, est le perceptron multicouche à rétro propagation d'erreur avec apprentissage supervisé, ses données sont :

-Les entrées : les **4 premières fréquences naturelles**.

-Les sorties : les **5 modules de rigidité** des cinq poutres.

III.3.2.2.LES ÉTAPES DE L'APPRENTISSAGE :

De la même manière que pour des applications précédentes.

1- initialiser les poids et les biais avec des valeurs aléatoires faibles.

2- présenter le vecteur d'entrée ainsi que le vecteur de sortie désiré.

3- calculer la sortie du réseau.

4- calculer l'erreur entre la valeur désirée, et celle calculée par le réseau ensuite l'erreur est propagé vers les neurones de la couche cachée pour calculer le gradient des poids.

5- ajuster les poids et les biais du réseau.

6- présenter un autre vecteur et l'aller à l'étape 3.

7- fin.

Le processus d'apprentissage s'arrête. Lorsque l'erreur se stabilise à un niveau acceptable. Ou à un nombre d'itération fixé.

III.3.2.3.ARCHITECTURES :

La recherche de l'architecture optimale consiste à tester plusieurs architectures. Avec une ou plusieurs couches cachées. Et de prendre l'architecture qui à L'erreur minimum. -Pratiquement l'architecture se construit de la façon suivante :

On part d'une architecture de faibles neurones dans la couche cachée. On met le processus de l'apprentissage en marche. On fait plusieurs initialisations des poids en agissant en même temps sur les paramètres (α et η). On enregistre l'erreur donnée. On augmente le nombre de neurones dans la couche cachée. On répète la même procédure. Et à chaque fois on enregistre l'erreur. Parmi les architectures obtenues on choisit l'architecture qui donne l'erreur minimum.

Les tests sont faits à l'aide de '**nntool** '. le module (neural network Tools).

-Après le test de plusieurs architectures. On a choisi l'architecture suivante :

-Une couche d'entrée avec **4 neurones**.

-Une couche cachée avec **10 neurones**.

-Une couche de sortie avec **5 neurones**.

-La fonction d'apprentissage ' **traingdm** ', puisqu'elle est efficace lorsqu'on ne dispose pas d'une base de données riche.

Les paramètres de performance finals :

-Nombre d'itération : **1303** ; - Pas d'apprentissage : **0.15.**

-Max Fail: **5** ;

-Mc: **0.8** ;

-Min_grad : **1e-10** ;

-Show: **25.**

-Erreur : **0.0108** (Figure : 3.20) ;

-Biais et poids (voir tableau ci-après).

-Erreur désirée **: 0.001.**

TABLEAU DES POIDS ET BIAIS :

B1 {10}	W [4x10]	W [10x5] ^{T}	B2 {5}
-15.99 -2.76 9.05 0.05 -4.28 0.09 9.42 1.72 -8.83 0.30	11.01 5.97 -0.63 0.77 6.40 -6.17 -6.63 6.28 -4.09 -3.15 5.98 -8.09 7.62 -5.15 0.16 -8.10 0.51 11.61 -0.48 -2.28 9.37 -7.39 2.46 -4.94 -7.65 -0.75 1.98 -8.89 -1.40 -3.87 -7.95 8.01 7.95 3.22 7.14 6.30 7.75 1.54 -9.68 4.65	-0.75 2.75 1.05 0.58 2.94 -1.27 4.54 -0.39 3.6 2.97 1.15 -1.26 1.09 0.5 2.92 3.31 3.25 3.9 1.49 -2.51 2.43 -0.31 2.86 4.68 -1.67 1.89 4.36 1.60 -1.07 -1.96 -0.62 -2.07 -0.5 2.60 -1.55 -3.04 -1.54 -3.6 4.94 -0.28 1.55 1.85 0.5 -0.60 1.48 2.12 -3.09 -1.1 3.50 -2.43	0.76 1.96 1.08 -5.95 3.58

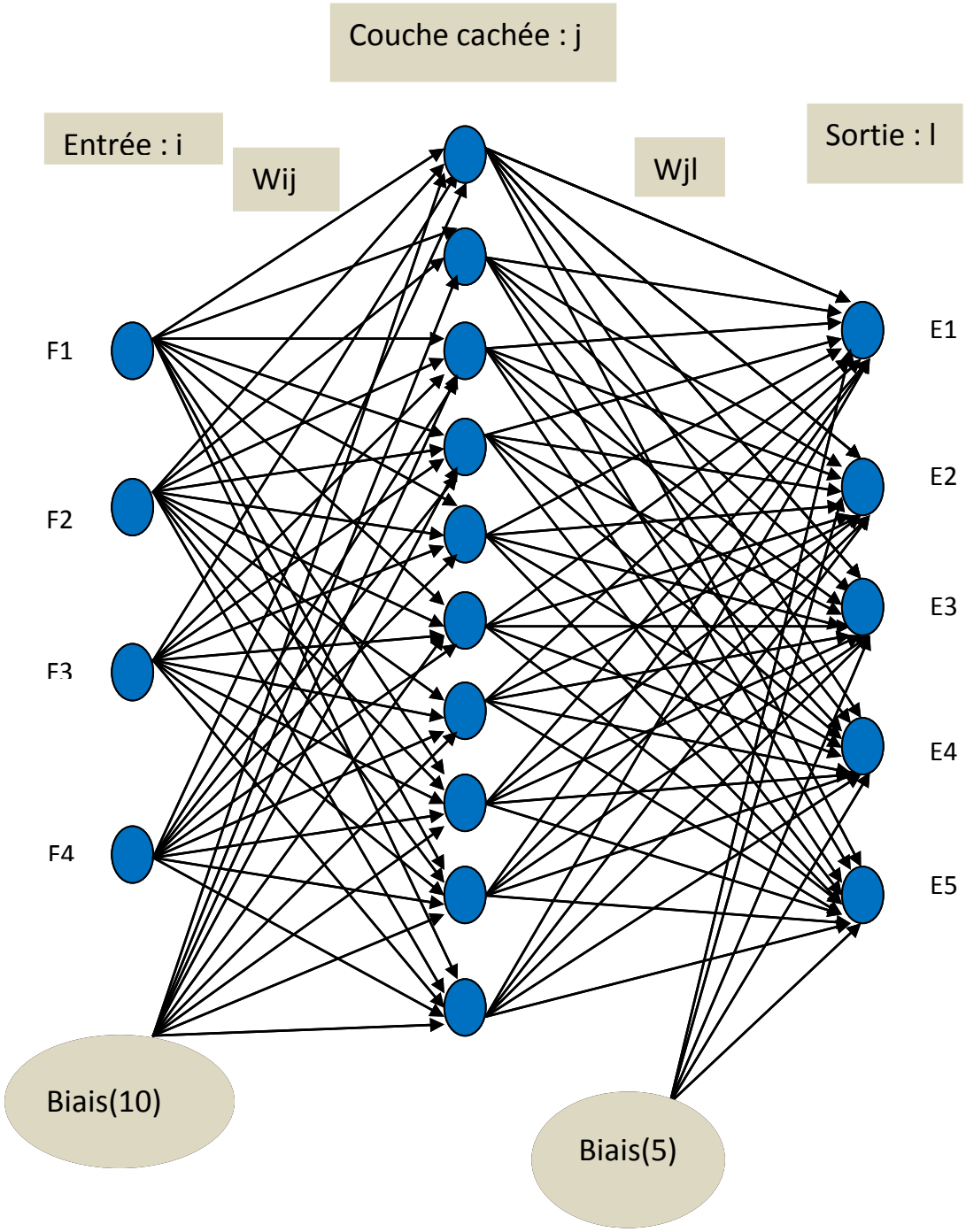


Figure 3.20 : Architecture du réseau

III.3.2.4. GRAPHE DE L'ERREUR :

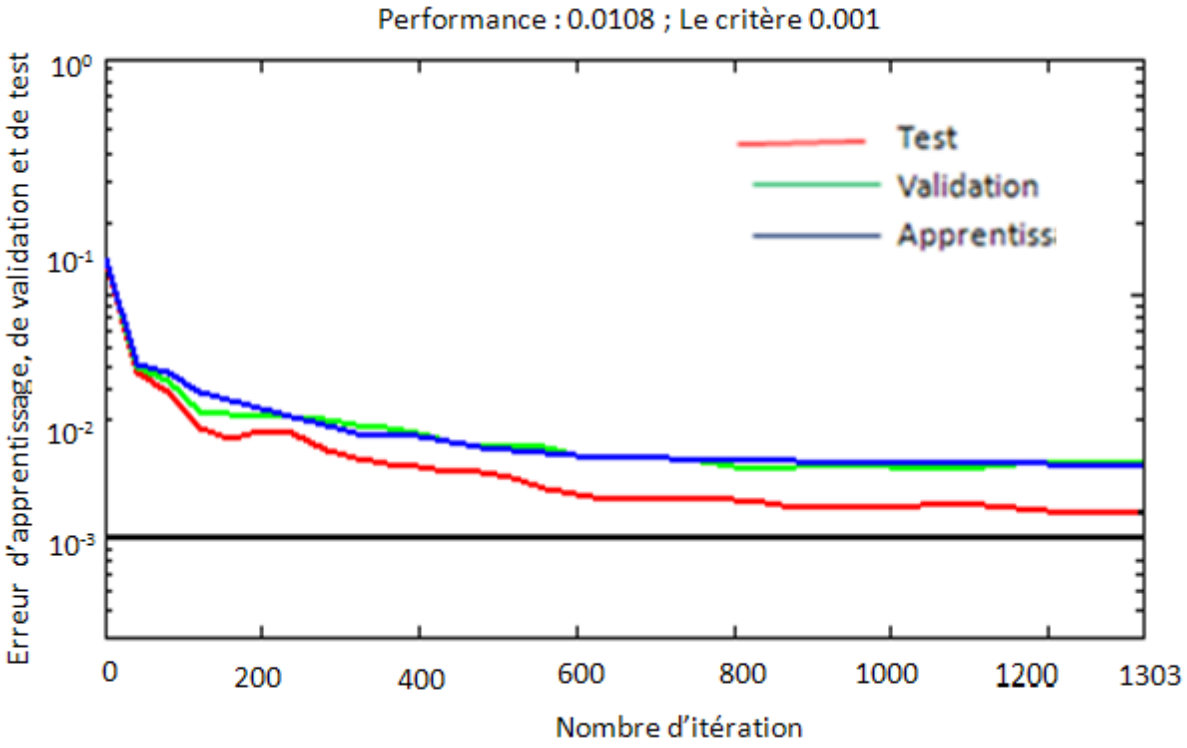


Figure 3.21 : Courbes d'erreur (Apprentissage, de validation et de test)

Le graphe de l'erreur a pu converger vers une erreur 0.0108 après 1303 itérations. Qu'on peut la considérer acceptable.

Les courbes ci-après définissent Le rapport entre les valeurs désirées en abscisses et les valeurs calculées par le réseau en ordonnées. On constate que toutes les données sont concentrées, sur la droite en pointillées avec un taux de classification de 96%, ce qui signifie que la tache de l'apprentissage a réussi.

III.3.2.5.COURBES DE RÉGRESSION :

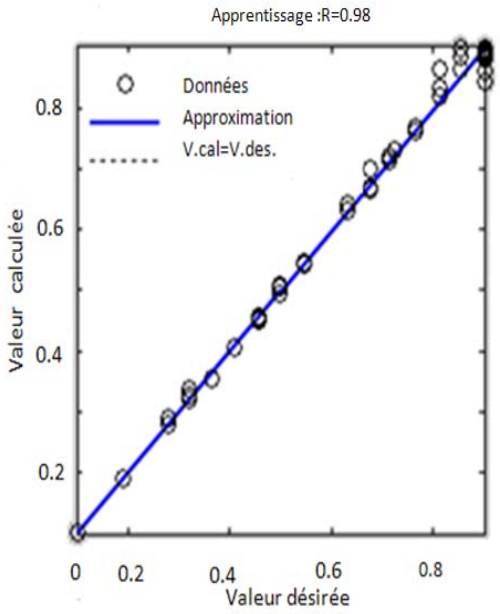


Figure 3.22 : Courbe de régression (Apprentissage)

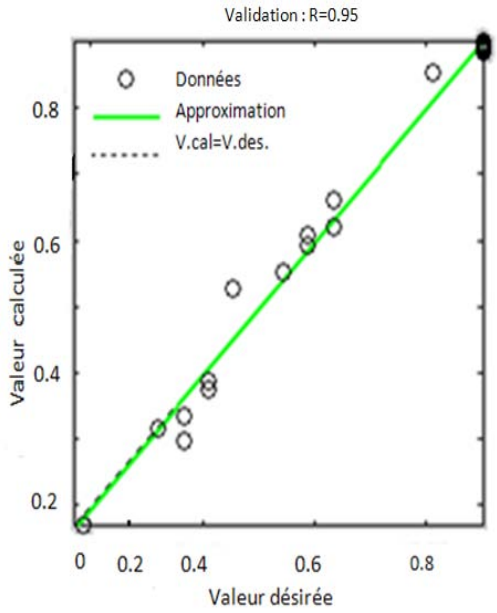


Figure 3.23 : Courbe de régression (Validation)

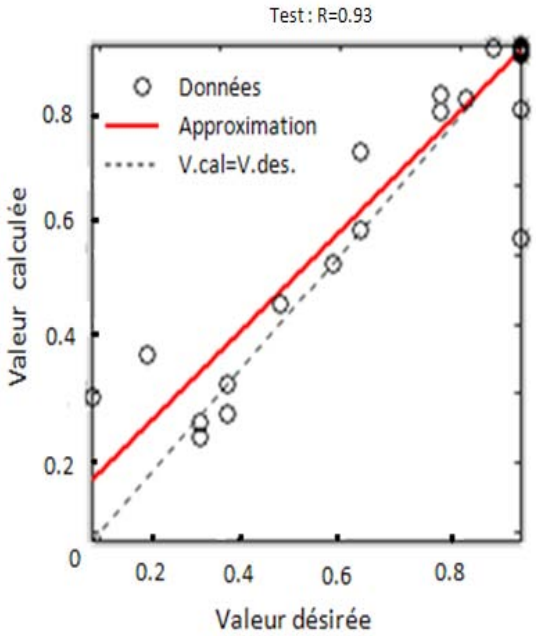


Figure 3.24 : Courbe de régression (Test)

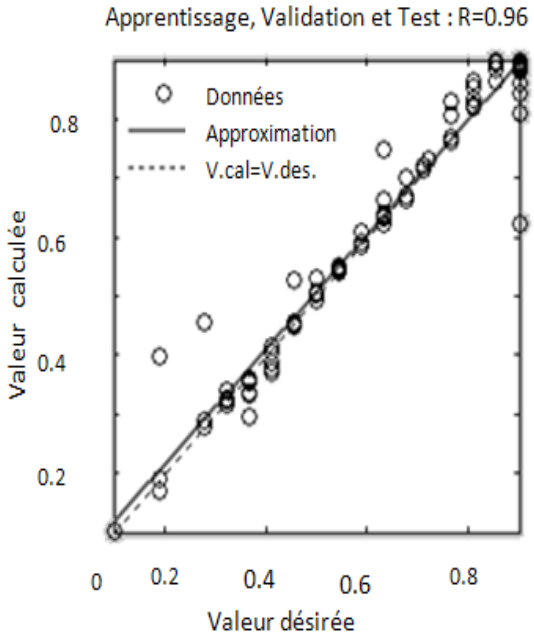


Figure 3.25 : Courbe de régression (l'ensemble)

III.3.3. COMPARISON DES RÉSULTATS :

Modules De Rigidité Obtenues par (EF)					Modules De rigidité Obtenues par (RN)				
E1	**E2**	**E3**	**E4**	**E5**	E1	E2	E3	E4	E5
0.95	0.95	0.95	0.95	0.42	0.97	0.96	0.93	0.91	0.48
0.95	0.95	0.95	0.95	0.84	0.98	0.95	0.97	0.99	0.87
0.95	0.95	0.95	0.53	0.95	0.94	0.97	0.98	0.50	0.93
0.95	0.95	0.74	0.95	0.95	0.99	0.99	0.76	0.97	0.95
0.95	0.74	0.95	0.95	0.95	0.93	0.76	0.96	0.96	0.96
0.53	0.95	0.95	0.95	0.95	0.60	0.93	0.98	0.93	0.97
0.21	0.95	0.95	0.95	0.95	0.25	0.97	0.91	0.90	0.89
0.58	0.95	0.95	0.95	0.95	0.54	0.90	0.98	0.91	0.99
0.95	0.95	0.95	0.95	0.21	0.91	0.92	0.99	0.98	0.94

-Les valeurs des modules de rigidité obtenues par la méthode des réseaux de neurone sont très proches des valeurs obtenues par éléments finis. Ce qui nous permet de dire que cette technique peut aider les ingénieurs à résoudre des problèmes complexes en exploitant des données qu'on peut les mesurer , en évitant ainsi des modélisations longues et compliquées, sans garantir leurs convergences.

CONCLUSION

CONCLUSION GÉNÉRALE

La méthode des réseaux de neurones présente de nombreux avantages par rapport aux méthodes classiques, utilisées en identification, diagnostique, l'évaluation et détermination de l'endommagement dans les structures mécaniques.

En effet la complexité de mise en œuvre de tels réseaux est bien très présente, Indépendante de la complexité des problèmes traités .en outre un réseau qui a bien appris peut servir de signature à une structure .il peut être réutilisé sans qu'il est besoin d'un nouvel apprentissage.par contre la méthode des réseaux de neurones, n'est pas systématique (aucune règle ne guide la détermination de l'architecture optimale à utiliser pour un problème donné).

La détermination de cette architecture optimale peut même devenir, à certaines occasions très fastidieuses. L'avancement des connaissances dans le domaine des réseaux de neurones appliquées à l'évaluation et à l'identification des structures, devrait par contre permettre à l'élaboration des règles précises guidant l'ingénieur vers l'architecture optimale à utiliser pour un problème donné.

Dans ce mémoire nous avons mis en lumière l'importance de l'emploi des réseaux de neurones, et leur capacité de résoudre des problèmes physiques, grâce un à modèle capable d'apprendre avec l'exemple, et déduire la relation existante entre les différents paramètres caractérisant ces problèmes, sans recourir à des formulations complexes qui pourraient être couteuses et gourmandes en temps.
Par manque de bases des données réelles, nous avons utilisé des données obtenues par éléments finis.

Nous avons validé notre approche par trois applications. **La première** consiste à appliquer le modèle de la rétro propagation, pour prédire la contrainte maximale

selon le critère de Von mises, ainsi que sa localisation et de comparer les résultats avec ceux, obtenue par la méthode des éléments finis, montrant à travers cet exemple que ce modèle peut remplacer fidèlement les méthodes conventionnelles, loin des formulations complexes, qui peuvent prendre beaucoup de temps.

- **La deuxième** consiste à détecter la position de l'excitation dans une poutre en mesurant la flèche dans des points fixes, cette application qui est inspirée d'un cas réel relevant de la détection des défauts, que peut présenter un arbre d'une machine en service, évitant par conséquent son endommagement, en la contrôlant par la mesure de sa flèche.

- **La troisième** application consiste à identifier l'endommagement des poutres d'un portique, en mesurant les fréquences naturelles, cette technique qui est largement utilisée pour détecter les zones endommagées, dans les ponts et les structures mécaniques pour des fins d'évaluation et de réparation.

Les résultats obtenus concernant les trois exemples sont satisfaisantes, ils nous permettent de conclure que l'application de cette méthode pour la prédiction, l'identification et le diagnostique est encourageante. Ce qui nous laisse confiant quant à son efficacité et sa capacité à résoudre des problèmes plus complexes.

BIBLIOGRAPHIE

BIBLIOGRAPHIE

Références:

[1] Barai S V, Pandey P C 1995a: Multilayer Perceptron in Damage Detection of Bridge Structures. Comput. Struct. 54: 597-608.

[2] Barai S V, Pandey P C; 'Vibration signatures analysis using artificial neural network'. Journal of computing in civil engineering. (9). 1995.

[3] Bishop C M 1998: Neural networks for pattern recognition (Oxford: Clarendon) .

[4] Casimir C. An introduction to Neural Computing .Swickley.1987

[5] Davalo E. et Naim P. Des réseaux de neurone. Deuxième Edition.1993

[6] Hajela P.; Berke L.; 'Neurobiological computational models in Structural analysis and design'. Computers structures. Vol. (4). 1995.

[7] Pernot S., Lamarque C.-H., Application of neural networks to the modelling of someconstitutive laws, Neural Networks, Vol.12, p.371-392, 1999.

[8] Bourret P. Reggia J. Réseaux neuronaux une approche connexionniste de l'intelligence artificielle . Teknea , Toulouse Marseille - Barcelone.1999]

[9] Wang S. Réseaux de Neurone multicouches Artificiels : Algorithmes d'apprentissage, implantation sur Hypercube.

[10] Watous R. L., Learning algorithms for connectionist networks: Applied gradient methodscof non lineair optimization. IEEE, First International Conference on Neural Networks, San Diego, California, 1987.

[11] Zurada J M, introduction to artificial neural system. West Publishing Company, USA.1992.

[12] Hornik K., Stinchcombe M., White H., Multilayer Feedforward Networks are Universal Approximators, Neural Networks, Vol.2, p.359-366, 1989

[13] Dayoff J.E., Neural Network Architectures: An introduction .Van Nostrand Reinhold, Newyork.1990.

[15] Rumelhart D., Hinton G., Williams R., Parallel Distributed Processing, MIT Press, Vol.1, Cambridge, 1986.

[16] Szewczyk Z P, Hajela P 1994 :
Damage detection in structures based on feature sensitive neural networks. ASCE J. Comput. Civil Eng. 8: 163-178

[17] Rosenblatt F., The Perceptron : a probabilistic model for information storage

[18] Mozolin M., Thill J.-C., Lynn Usery E., Trip distribution forecasting with multilayer perceptron neural networks : A critical evaluation, Transportation Research Part B, Vol.34, p.53-73, 2000.

[19] Davalo E., Naim P., Des Réseaux de Neurones, EYROLLES, Deuxième édition, 1993.

[20] Wu X., Ghaboussi J., Garrett J.H.Jr, "Use of Neural Networks in Detection of Structural Damage". Computers & Structures Vol.42, (4). Pergamon Press. 1992.

[21] Soudani A. Applications des réseaux de neurone aux mesures en écoulement turbulent. These de Doctorat, IN.G.P , France 1996.

[22] Bishop C., Neural Networks for Pattern Recognition, Oxford University Press, 1995.

[23] Rumelhart D E, Hinton G E, Williams R J 1986:
Learning internal representations by error back-propagation. Parallel Distrib. Process. Explor Microstruct. Cogn. 1: 318-362

[24] Nguyen D. et Widrow B, Neural Network for self-learning control system. Workshop on industriel applications of Neural Networks.1999.

[25] Franzini M A. "Speech recognition with back propagation. The ninth annual conference of the IEEE. Engineering in medicine and biology society". New York, pp 1702-1703, 1987.

[26] Guivier J. P. “Neuralworks, Networks II. Neural Wareinc”. Court Swickley, 1987.

[27] Nerrand O., Roussel-Ragot P., Personnaz L., Dreyfus G., Neural networks and nonlinear adaptive filter ing : unifying concepts and new algorithms, Neural Computation, Vol.5,p.165-199, 1993.

[28] Sejnowski T. Rosenberg C. “Parallel networks that learn to pronounce English TEXT. Complex systems”. Vol 1. 145-168, 1987.

[29] Chinchalkar S 2001: Determination of crack location in beams using natural frequencies.J.Sound Vibr. 247: 417-429 frequencies. J. Sound Vibr. 242: 577-596

[30] Sanayei M, Onipede O 1991:
Damage assessment of structures using static test data.

ANNEXES

Pergamon

0045-7949(94)00377-7

Computers & Structures Vol. 54, No. 4, pp. 597–608, 1995
Elsevier Science Ltd
Printed in Great Britain
0045-7949/95 $9.50 + 0.00

MULTILAYER PERCEPTRON IN DAMAGE DETECTION OF BRIDGE STRUCTURES

P. C. Pandey and **S. V. Barai**
Department of Civil Engineering, Indian Institute of Science, Bangalore-560 012, India

(*Received* 19 *October* 1993)

Abstract—Recent developments in artificial neural networks (ANN) have opened up new possibilities in the domain of structural engineering. For inverse problems like structural identification of large civil engineering structures such as bridges and buildings where the *in situ* measured data are expected to be imprecise and often incomplete, the ANN holds greater promise. The detection of structural damage and identification of damaged element in a large complex structure is a challenging task indeed. This paper presents an application of multilayer perceptron in the damage detection of steel bridge structures. The issues relating to the design of network and learning paradigm are addressed and network architectures have been developed with reference to trussed bridge structures. The training patterns are generated for multiple damaged zones in a structure and performance of the networks with one and two hidden layers are examined. It has been observed that the performance of the network with two hidden layers was better than that of a single-layer architecture in general. The engineering importance of the whole exercise is demonstrated from the fact that measured input at only a few locations in the structure is needed in the identification process using the ANN.

NOTATION

n	number of input nodes
m	number of hidden nodes
p	number of output nodes
i	1, n
j	1, m
k	1, p
$\{X_i\}$	training input vector
$[W_{ji}]$	weights between input and hidden layer
$\{out_j\}$	activation values for the hidden nodes
$[W_{kj}]$	weights between hidden and output layer
$\{o_k\}$	activation values for the output nodes or output vector
$[\Delta W_{kj}]$	error in weights between hidden and output nodes
η	learning parameter, here 0.9
$\{t_k\}$	target output vector
α	momentum parameter, here 0.7
$[\Delta_p W_{kj}]$	error $[\Delta W_{kj}]$ in previous pass
$[\Delta W_{ji}]$	error in weights between input and hidden nodes
$[\Delta_p W_{ji}]$	error $[\Delta W_{ji}]$ in previous pass
P	total number of samples
E	average system error
θ_j	threshold or bias parameter
θ_0	parameter for shape of the sigmoid function

INTRODUCTION

Generally in civil engineering practice, existing structures are inspected by experienced engineers who determine the location of the damaged zone in the structure and the extent of the damage. It is believed that system identification techniques [1, 2] can be extended to structures for systematic damage detection and evaluation. Structural identification is a process for constructing a mathematical description of a physical system when both the input and the corresponding output are known.

When a structure undergoes various degrees of damage, certain characteristics have been found to undergo changes. In order to identify those changes, during inspection a sequence of tests may be conducted and the resulting data such as load, displacements, strains, acceleration, etc. can be measured. From such data, mechanical properties, such as stiffness/strength, and dynamic characteristics, such as natural frequency and damping can be estimated and this can be dealt with by system identification techniques. Structural identification can be done both under static [3] and dynamic [1] conditions. These techniques have been demonstrated in the past in structural damage detection using conventional computing techniques. The algorithm adopted is generally complex and is not suited to the situations where measured data are imprecise or inadequate. The recent emergence of artificial neural networks (ANN) can be explored as an alternative tool for identification exercise in such situations. This paper presents the application of multilayer perceptron in identification of damage in trussed bridge structures.

ARTIFICIAL NEURAL NETWORK CONCEPT

Many good reviews on the neural network paradigms are available in the literature [4–8]. Here, only a brief conceptualization of neural nets and its computational counterpart is given.

An ANN is a software simulation or a hardware implementation of a structure derived from studying the physiology of groups of nerve cells or neurons.

Based on the current understanding of neurons, a computational model is developed.

NATURAL NEURON AND THE NETWORK (BRAIN)

A basic nerve cell or neuron is composed of a processing body, transmitting axons and receiving synapses and dendrites (Fig. 1). Let us consider enormous numbers of such cells interconnected in an additive manner. If a processing cell body receives a stimulus from say, an electrical or a chemical source or from a pressure or temperature change, this causes an activation potential to form internally. This potential rapidly spreads along all axons. It is delayed by distance, electrochemical state and synapse resistance at the terminus, where the axon joins another cell's body or its dendrites. The receiving cell is thus stimulated by all the transmitting axons connected to its body or its dendrite receptors. The signal received from any one axon or synapse or dendrite source is equivalent to the activation potential of the originating cell multiplied by a weight proportional to the time delay and resistance met along the axon, across the synapse and/or along the dendrite. The sum of all such weighted stimuli arriving within a short period becomes the input to the receiving nerve cell.

The state of that cell's internal processes determines the blending, squashing and thresholding performed before the signal is transmitted out of all its axons to the temporally next layer of nerve cells. The recent history of the receiving cell thus determines its excitability and its activation function. The outgoing stimulus travels down more axons and, weighted by time and resistance, is received by the next round of processing cells.

Our brain is understood to possess 10^{10} to 10^{11} neurons massively interconnected with 10^3 to 10^4 synaptic connections resulting in 10^{13} to 10^{15} connections. Even when we assume the neurons to take binary forms (firing or not firing), the total number of the degrees of freedom of the natural neural network (i.e. the brain) is truly astronomical, reaching $2^{10^{15}}$!

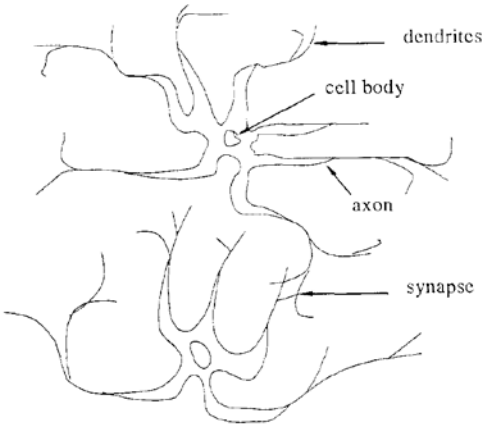


Fig. 1. Biological neuron.

ARTIFICIAL NEURON AND ARTIFICIAL NEURAL NETWORK (ANN)

Mathematically, a single nerve cell can be modelled as an artificial neuron (computational unit) consisting of receiving sites (synapses), receiving connections (dendrites), a processing element (cell body) and transmitting connections (axons). A typical architecture of an artificial neuron is shown in Fig. 2.

A neuron (or node) receives input stimuli from other neurons if they are connected to it or/and the external world. A neuron can have several inputs, but has only one output. This output, however, can be routed to the inputs of several other neurons.

Each neuron has certain constant parameters associated with it. These are its threshold, transfer function and the weights associated with its input. Each neuron performs a very simple arithmetic operation, i.e. it computes the weighted sum of its inputs, subtracts its threshold from the sum, and passes the result through its transfer function. The output of the neuron is the result obtained from this function. The output of a neuron is, therefore, a mathematical function of its inputs. The most common transfer functions used in neural-network literature are the hardlimiter, threshold, range and sigmoid nonlinearities.

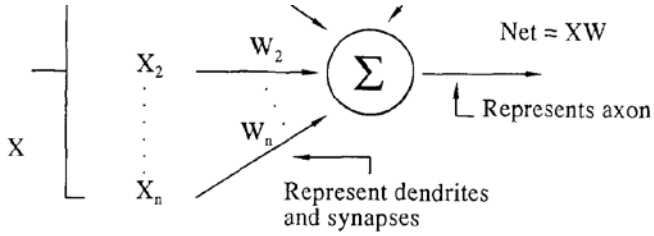
Artificial neural networks are modelled on the concept of present understanding of the functioning of the brain thus comprising a massively interconnected network of a large number of artificial neurons or computational units.

Neural network models are specified by the net topology, node (artificial neuron) characteristics and training or learning rules. The function of a neural network is determined by these parameters. The architecture of the network determines the inputs of each node. The node characteristics (threshold, transfer function and weights) determine the output of the node. The training or learning rules determine how the network will react when an unknown input is presented to it.

The large connectivity degree of the neurons and the massive parallelism as well as their nonlinear analogue response and learning capabilities are the basic factors which characterize the computational effectiveness of a neural network.

A computing device based on neural network has a greater fault tolerance than a classical sequential computer due to the increased number of locally connected processing nodes. Thus, some neurons or links out of order do not diminish considerably the performance of the network.

Neural networks hold the promise of providing a fast, data-driven modeling system with desirable robustness properties since their strengths and weak-



$$\text{Net} = X_1W_1 + X_2W_2 + \ldots\ldots + X_nW_n$$

Fig. 2. Artificial neuron.

nesses complement those of more traditional analytic techniques.

SUITABILITY OF ANN IN DAMAGE DETECTION OF STRUCTURES

A robust damage assessment methodology must be capable of recognizing patterns in the observed response of the structures resulting from individual member damage including the capacity of determining the extent of member damage. The area of damage assessment of structure is yet to evolve. A general damage assessment paradigm in the context of steel bridges has been presented by Pandey and Barai [9] discussing various issues. The difficulty is encountered because the data available from *in situ* measurements are often imprecise and inadequate. This problem can be tackled more effectively by the use of ANN which has also been demonstrated in recent papers [10, 11]. However, there is still a mystery surrounding the development of the architecture of networks. A systematic study on various aspects of networks simulation needs to be carried out in order to design a reliable network.

The present paper attempts to examine the suitability of the multilayer perceptron with reference to backpropagation learning algorithm.

MULTILAYER PERCEPTRON

A network architecture is a specification of the artificial neurons and their relationships. The multilayer perceptron is very popular ANN architecture and has performed well in a variety of applications in several domains including a few structural engineering applications published so far. A multilayer perceptron consists of an array of input neurons, known as the input layer, an array of output neurons, known as the output layer and a number of hidden layers. Each neuron receives a weighted sum from each neuron in the preceding layer and provides an input to every neuron of the next layer. The activation of each neuron is governed by a threshold function. In order to train the network, a popular training algorithm of the multilayer perceptron is the backpropagation method where the error calculated at the output of the network is propagated through the layers of neurons to update the weights. The learning algorithm is illustrated in the next section.

IMPLEMENTATION STEPS OF THE LEARNING ALGORITHM

Backpropagation algorithm/the generalized delta rule algorithm is very effective for learning examples of the behavioural phenomena. The derivation of the generalized delta rule [6, 12] is included in the Appendix for completeness. The schematic view of the network is shown in Fig. 3 and the steps involved for implementing the algorithm are given as follows.

Step 1. Select a number of input nodes (n), output nodes (p) and hidden nodes (m) and first training example (x_i)

$$\{X_i\} = \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix}. \tag{1}$$

Step 2. Initialize the weights using random number generator in the range of -0.5 to 0.5 [6]

$$[W_{ji}] = \begin{bmatrix} w_{11} & w_{12} & w_{13} & \cdot & \cdot & w_{1n} \\ w_{21} & w_{22} & w_{23} & \cdot & \cdot & w_{2n} \\ w_{31} & w_{32} & w_{33} & \cdot & \cdot & w_{3n} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ w_{m1} & w_{m2} & w_{m3} & \cdot & \cdot & w_{mn} \end{bmatrix}. \tag{2}$$

Step 3. Compute the value of $\{net_j\}$ for the hidden nodes [see eqn (A1)]

$$\{net_j\} = \begin{Bmatrix} net_1 \\ net_2 \\ \vdots \\ net_m \end{Bmatrix} = [w_{ji}]\{X_i\}. \quad (3)$$

Step 4. *Calculate the activation value* $\{out_j\}$ for the hidden nodes [eqn (A2)]. Here the sigmoidal function has been used (Fig. 4). The parameter θ_j is to shift the activation function to the left and right along the horizontal axis depending upon its positive or negative values, respectively. Similarly the θ_0 is used to modify the shape of the sigmoid (see Fig. 4)

$$\{out_j\} = \begin{Bmatrix} out_1 \\ out_2 \\ \vdots \\ out_m \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{Bmatrix} \left(\begin{Bmatrix} net_1 \\ net_2 \\ \vdots \\ net_m \end{Bmatrix} \right) = \begin{Bmatrix} f_1(net_1) \\ f_2(net_2) \\ \vdots \\ f_m(net_m) \end{Bmatrix} = \begin{Bmatrix} 1/(1+e^{(-net_1+\theta_1)/\theta_0}) \\ 1/(1+e^{(-net_2+\theta_2)/\theta_0}) \\ \vdots \\ 1/(1+e^{(-net_m+\theta_m)/\theta_0}) \end{Bmatrix}. \quad (4)$$

Step 5. Calculate the value of $\{net_k\}$ for the output node

$$\{net_k\} = \begin{Bmatrix} net_1 \\ net_2 \\ \vdots \\ net_p \end{Bmatrix} = [w_{kj}]\{out_j\}. \quad (5)$$

Step 6. Calculate the activation value $\{out_k\}$ for the output nodes

$$\{out_k\} = \begin{Bmatrix} o_1 \\ o_2 \\ \vdots \\ o_p \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \\ \vdots \\ f_p \end{Bmatrix} \left(\begin{Bmatrix} net_1 \\ net_2 \\ \vdots \\ net_p \end{Bmatrix} \right) = \begin{Bmatrix} f_1(net_1) \\ f_2(net_2) \\ \vdots \\ f_p(net_p) \end{Bmatrix} = \begin{Bmatrix} 1/(1+e^{(-net_1+\theta_1)/\theta_0}) \\ 1/(1+e^{(-net_2+\theta_2)/\theta_0}) \\ \vdots \\ 1/(1+e^{(-net_p+\theta_p)/\theta_0}) \end{Bmatrix}. \quad (6)$$

Step 7. Calculate error $[\Delta W_{kj}]$ [eqn (A24)]

$$[\Delta W_{kj}] = \eta\{t_k - o_k\}\{o_k\}\{out_j\}^T + \alpha[\Delta_p W_{kj}] \quad (7)$$

α and η are usually selected from experience.

Step 8. Compute the new values of weights between the hidden and output layers

$$[W_{kj}] = [W_{kj}] + [\Delta W_{kj}]. \quad (8)$$

Step 9. Calculate the $[\Delta W_{ji}]$ for input to hidden weights

$$[\Delta W_{ji}] = \eta\{out_j\}\{t_k - o_k\}\{o_k\} \times [W_{kj}]^T\{X_i\}^T + \alpha[\Delta_p W_{ji}]. \quad (9)$$

Step 10. Calculate the new values of the weights between input and hidden layer

$$[W_{ji}] = [W_{ji}] + [\Delta W_{ji}]. \quad (10)$$

The algorithm continues for all set until the average system error (ASE) [eqn (A7)] between the target output and computed output is close to the tolerance specified.

RECENT APPLICATIONS OF THE BACKPROPAGATION ALGORITHM IN STRUCTURAL DAMAGE DETECTION

Ghaboussi *et al.* [13] and Wu *et al.* [14] have demonstrated the use of backpropagation algorithm in structural applications. Hajela and Berke [15] have implemented neural networks paradigm in automated structural design. They have examined two distinct architectures, namely conventional layered architecture with input–output layers and hidden layers and modified flat networks termed as functional link nets. For both types of networks they adopted supervised learning with a backpropagation algorithm.

Wu *et al.* [11] used backpropagation neural networks architecture with single hidden layer to simulate damage states in a three storey frame. The structure was subjected to earthquake base acceleration and the transient response was computed in time domain. The Fourier spectra of the computed relative acceleration time histories of the top floor for various members were used in training the neural networks. The member damage was defined as a reduction in the member stiffness. A neural networks architecture consisting of 200 input units, three output units and a hidden layer with ten nodes was selected by trial and error. The input represented the amplitudes of the Fourier spectra and the output represented the damage condition of each member. They used a total of 43 training cases consisting of 42 frequency spectra of relative accelerations recorded at top floor level along with the information that indicates which member is damaged and the extent of the damage. In the course of training the network they observed that

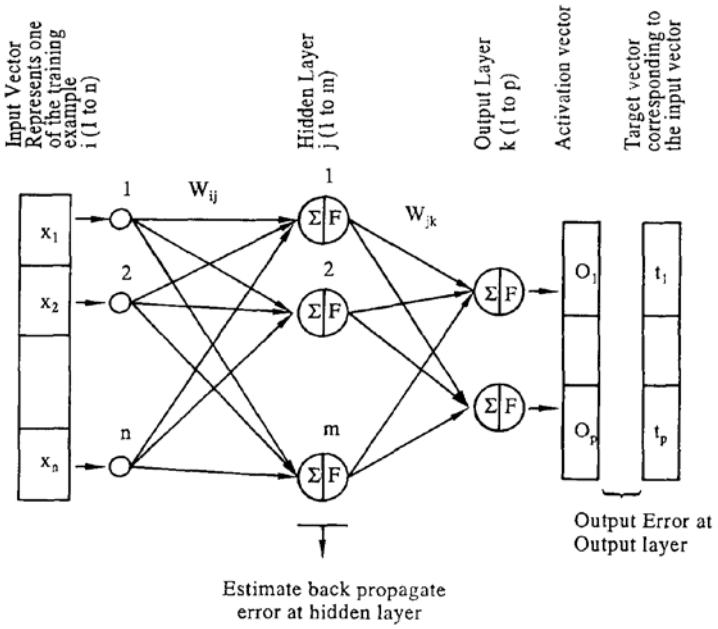


Fig. 3. Backpropagation in single hidden layer of artificial neural networks.

if the hidden layer was too small, the network would not converge. On the other hand, if the network was too large, it would not converge either. Obviously the performance of the network which depends on the hidden layer is problem dependent and needs to be researched further.

Szewczyk and Hajela [10] have considered the damage detection of structures as an inverse problem. They modeled the damage as a reduction in the stiffness of structural elements which were associated with observed static displacements under prescribed loads. To generate this reverse mapping between stiffness of individual member of the structure and global static displacements, an improved counter propagation neural network was used. They performed simulation of a frame structure with nine bending elements and 18 degrees of freedom (X–Y displacements and rotation at each node). In training the network 3600 randomly generated damage patterns were used. The size of the network architecture was governed by a control parameter also called as resolution parameter. In this investigation the resolution parameter used was 0.9. They observed from the exercise on the frame structure that the network performance was generally precise with gradual deterioration in presence of noisy and incomplete measurements. They also concluded that the resolution parameter plays an important role in designing the size of the network and is highly problem dependent.

Elkordy *et al.* [16] adopted backpropagation neural networks to model damage states of a five storey steel frame. Three neural networks were used. All the networks were designed with four input nodes, two

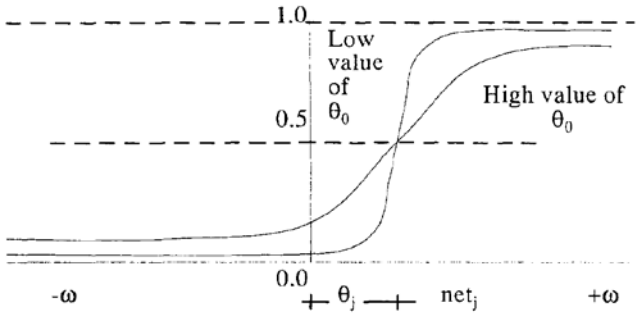


Fig. 4. Typical sigmoidal function.

output nodes and 14 hidden nodes chosen by trial and error. These networks trained with analytically generated states of damage, were used to diagnose damage states obtained experimentally from a series of shaking tests. Although the results were promising, they concluded that the relation between the number of damage patterns required for training the network to perform satisfactorily and the degree of simplification of the model needs to be investigated further.

From the literature survey, it is obvious that the design of a reliable ANN is as yet an unresolved issue. A more detailed treatment on the architectural aspect of the multilayer perceptron using backpropagation learning is given in this paper. The efficiency of the training paradigm and associated learning parameters are also looked at with special reference to bridge truss structures.

NETWORK ARCHITECTURE: IMPLEMENTATION ISSUES

The successful application of neural networks to a specific problem depends on two factors, namely representation and learning.

(a) *Problem representation: layers and nodes*

Choosing a topology for the network is a difficult task. If the number of hidden units is too small, then the network may not be sufficient to develop the required internal representation of the problem and therefore may not be able to perform the necessary recognition task. On the other hand if the number of hidden units is too large, then the network can learn to give the correct classification for all the data in the training set but its performance would degrade in the testing set. This is the case of 'overfitting', where the network learns the 'noise' presented in the training data and not the required general pattern. The appropriate selection of layers and nodes is problem dependent and the optimum layout can be arrived at only after extensive computational experimentation in the application domain.

(b) *Learning*

Learning rate (η). The choice of learning parameters (η, α) is a tricky task in the backpropagation learning algorithm. The learning rate (η) has to be chosen as high as possible to allow fast learning without leading to oscillations. Generally a value of 0.9 is recommended in the literature [6].

Momentum (α). The role of the momentum term (α) is to increase the speed of learning without leading to oscillations. Effectively the momentum term filters out high frequency variations of the error surface in the weight space, since it adds the effect of past weight changes on the current direction of movement in the weight space. So far, the value has been chosen by trial and error.

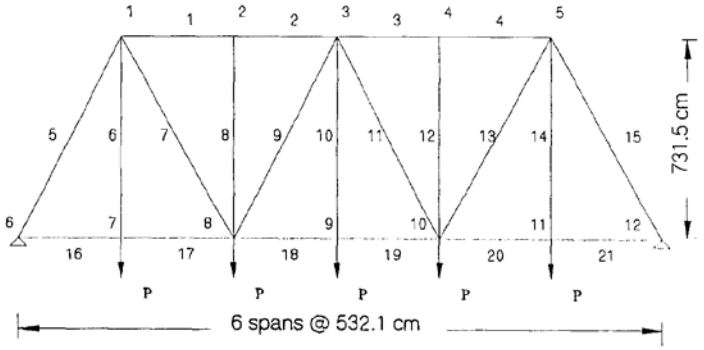
Error tolerance. The system error tolerance also known as average system error [see eqn (A7)] has direct influence on the convergence of the network during iterative learning. Smaller the value of ASE, more number of iterations and CPU time requires for convergence. This is also dependent on the training data.

(c) *Convergence*

For a network to converge to the required solution it is required to use a higher learning rate (η). But it is found that when a large η is used, the network oscillates. On the other hand, choosing a small η often a large number of iterations are required to reach the solution.

A proper selection of the learning rate (η), the momentum value (α), and the system error tolerance are needed for efficient learning and the design of a stable network.

Considering the governing issues, several networks were designed to model bridge truss problems, out of which only a few were selected from training, convergence and CPU considerations.



P = 100 kg.

E = 2 x 10^6 kg/cm^2

Fig. 5. Bridge-truss configuration.

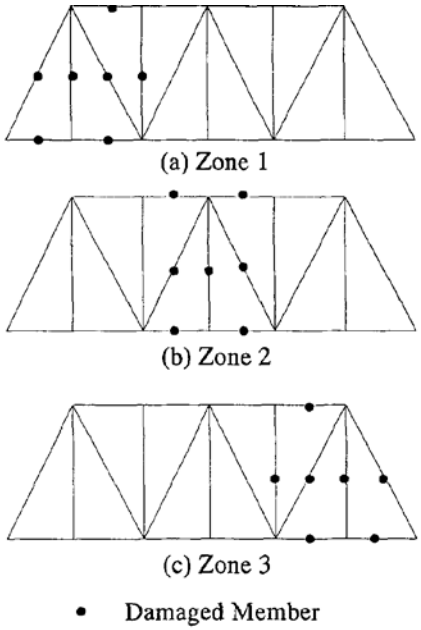


Fig. 6. Identification of damaged zone.

GENERAL ARCHITECTURE OF THE NETWORK FOR THE BRIDGE TRUSS

An extensive neural networks investigation was carried out on the problem of 21-bar bridge truss as shown in the Fig. 5 with three distinct zones assumed as damaged (Fig. 6). The bottom chord nodes (7, 8, 9, 10 and 11) were considered for measuring the response of the structure under given static load as shown in Fig. 5. The purpose of the exercise was to identify the members in the damaged zone and the reduction in their stiffnesses (here a function of cross-sectional areas) from the response data. The final two architectures (Figs 7 and 8) were arrived at after an extensive network parametric study on the bridge truss. In this network the input nodes represent the measured parameters (vertical displacements at the nodes, U_7, U_8, U_9, U_{10}, and U_{11}) and the output nodes are the identified parameters (cross-sectional areas $a_1, a_2, \ldots, a_{21}$ of the members representing the stiffness in assumed damage state) for the neural networks. The detailed study was carried out with the following two architectures: (1) the single hidden layer architecture 5–21–21 (Fig. 7) with five input nodes, 21 hidden nodes and 21 output nodes. And (2) the two hidden layer architecture 5–21–21–21 (Fig. 8) with five input nodes, 21 output nodes and 21 hidden nodes per layer.

In both the architectures the learning parameter η was assumed as 0.9, and the momentum parameter α was assumed as 0.7 [6]. Several trial runs were made with ASE of 0.1, 0.01 and 0.001. Finally compromise between the CPU time required for training, Number of iteration required for convergence and accuracy in the results, the value of ASE was selected as 0.01.

TYPICAL TRAINING PATTERNS

In the present study the bridge structure has been identified having three distinct damaged zones where members are assumed to be damaged in turn (Fig. 6).

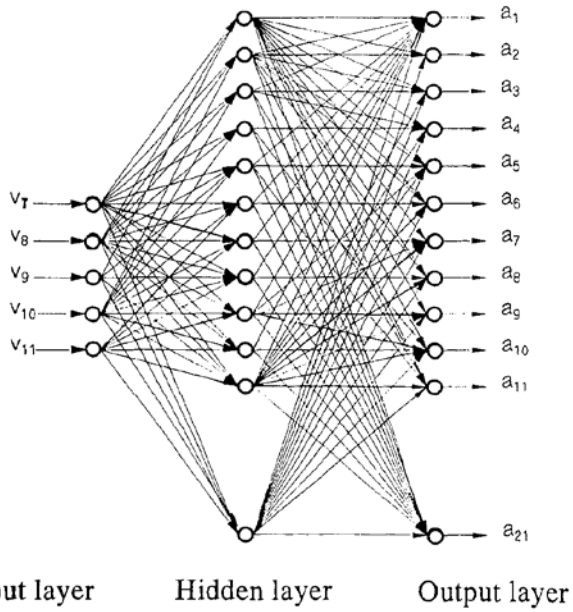


Fig. 7. One hidden layer architecture (5–21–21).

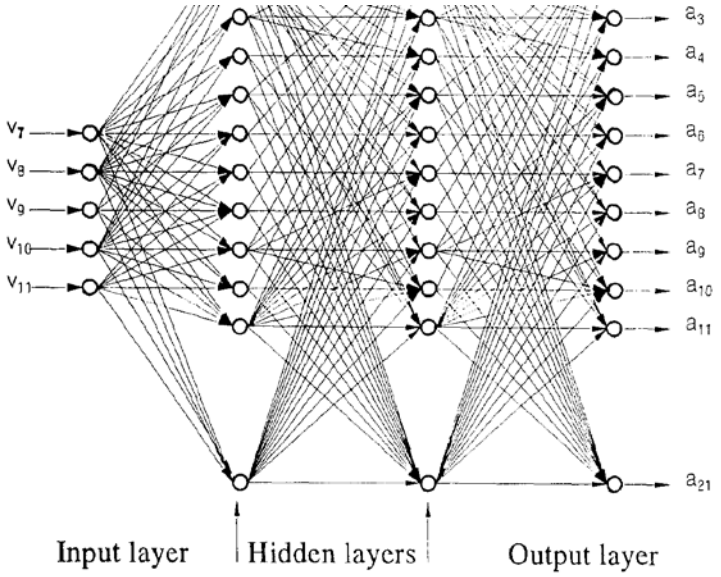


Fig. 8. Two hidden layer architecture (5–21–21–21).

Table 1. Typical samples of zone 1

Member number	FEM solution ($\times 10^2$)	Results obtained from trained NN architecture 5–21–21		Results obtained from trained NN architecture 5–21–21–21	
		ASE = 0.01 ($\times 10^2$)	Error (%)	ASE = 0.01 ($\times 10^2$)	Error (%)
(*a*) *Sample* 1					
1	0.1078400	0.1212100	12.3979959	0.1124380	4.2637234
5	0.1110600	0.1241090	11.7495041	0.1146220	3.2072716
6	0.1142800	0.1289780	12.8613920	0.1199050	4.9221230
7	0.1174900	0.1267840	7.9104567	0.1236480	5.2412987
8	0.1207100	0.1335600	10.6453495	0.1308620	8.4102373
16	0.1239200	0.1158040	6.5493860	0.1211300	2.2514558
17	0.1271400	0.1191840	6.2576675	0.1244750	2.0961130
(*b*) *Sample* 2					
1	0.1400100	0.1460560	4.3182611	0.1471300	5.0853500
5	0.1432200	0.1494910	4.3785710	0.1498310	4.6159678
6	0.1464400	0.1532160	4.6271544	0.1540010	5.1632056
7	0.1496500	0.1566420	4.6722436	0.1555850	3.9659295
8	0.1528700	0.1563420	2.2712111	0.1528210	0.0320501
16	0.1560900	0.1323770	15.1918812	0.1356780	13.0770788
17	0.1593000	0.1327140	16.6892624	0.1329840	16.5197754
(*c*) *Sample* 3					
1	0.1008200	0.1146810	13.7482662	0.1041760	3.3287070
5	0.1040400	0.1174420	12.8815842	0.1062620	2.1357186
6	0.1072500	0.1201010	11.9822845	0.1116200	4.0745959
7	0.1104700	0.1190420	7.7595773	0.1156440	4.6836281
8	0.1136800	0.1274720	12.1323023	0.1250150	9.9709778
16	0.1169000	0.1113230	4.7707429	0.1170480	0.1266089
17	0.1201200	0.1154710	3.8702950	0.1218040	1.4019336
(*d*) *Sample* 4					
1	0.1329800	0.1412610	6.2272463	0.1400810	5.3399000
5	0.1362000	0.1445910	6.1607985	0.1426660	4.7474313
6	0.1394100	0.1481450	6.2656918	0.1471530	5.5541215
7	0.1426300	0.1508360	5.7533545	0.1492170	4.6182423
8	0.1458500	0.1519910	4.2104850	0.1486300	1.9060614
16	0.1490600	0.1292300	13.3033705	0.1329850	10.7842484
17	0.1522800	0.1301670	14.5212736	0.1315170	13.6347580

Table 2. Typical samples of zone 2

FEM	Results obtained from trained NN architecture 5–21–21		Results obtained from trained NN architecture 5–21–21–21		
(a) *Sample* 1					
2	0.1078400	0.1253950	16.2787457	0.1233450	14.3777828
3	0.1110600	0.1284610	16.6681089	0.1259670	13.4224701
9	0.1142800	0.1321330	15.6221609	0.1286040	12.5341225
10	0.1174900	0.1302710	10.8783741	0.1294790	10.2042761
11	0.1207100	0.1350460	11.8764019	0.1365700	13.1389332
18	0.1239200	0.1176430	5.0653667	0.1198660	3.2714672
19	0.1271400	0.1220380	4.0129004	0.1229860	3.2672677
(b) *Sample* 2					
2	0.1400100	0.1505770	7.5473146	0.1537950	9.8457289
3	0.1432200	0.1534550	7.1463456	0.1570160	9.6327229
9	0.1464400	0.1566570	6.9769158	0.1605650	9.6455917
10	0.1496500	0.1587660	6.0915532	0.1627390	8.7464094
11	0.1528700	0.1613820	5.5681334	0.1687890	10.4134235
18	0.1560900	0.1370220	12.2160311	0.1478650	5.2694011
19	0.1593000	0.1371250	13.9202757	0.1475670	7.3653460
(c) *Sample* 3					
2	0.1721700	0.1687690	1.9753711	0.1782820	3.5499763
3	0.1753800	0.1714290	2.2528298	0.1821420	3.8556263
9	0.1786000	0.1742150	2.4552042	0.1864520	4.3964181
10	0.1818200	0.1795310	1.2589440	0.1891210	4.0155029
11	0.1850300	0.1803710	2.5179684	0.1939130	4.8008428
18	0.1882500	0.1508280	19.8788853	0.1700120	9.6881847
19	0.1914700	0.1476180	22.9028053	0.1666370	12.9696531
(d) *Sample* 4					
2	0.1008200	0.1186030	17.6383667	0.1159090	14.9662819
3	0.1040400	0.1216940	16.9684753	0.1184240	13.8254538
9	0.1072500	0.1254680	16.9864826	0.1208580	12.6881142
10	0.1104700	0.1226470	11.0229073	0.1212460	9.7546902
11	0.1136800	0.1279310	12.5360670	0.1284840	13.0225182
18	0.1169000	0.1123460	3.8956339	0.1129210	3.4037621
19	0.1201200	0.1178280	1.9080918	0.1167890	2.7730589

Considering each damaged zone independently and separately, a total of 40 training patterns have been generated with the help of finite element software and stored in files and an additional 10 testing patterns have been generated and stored separately for verification of the trained network after proper normalization. The normalization is essential while using the Sigmoid function, in order to have the input and training patterns values between 0 and 1. These normalized training patterns presented to the network help in faster convergence. For normalization of the input–output pair an interface program has been developed. In this study the backpropagation algorithm has been implemented in C on VAX8810. The networks were trained for the three distinct damaged zones separately and then tested for 10 testing patterns. Only four typical testing patterns have been presented here for the three zones separately (Tables 1–3). The observations of this study are discussed in the next section.

OBSERVATIONS

The results obtained for the computational experimentation were quite promising. Some of the typical results for the three zones have been given in Tables 1–3. However, the following observations are made based on all ten tested patterns:

1. Trained networks were able to identify the damaged zone for all the testing patterns.
2. Trained networks had provided reasonable value of cross sectional areas of all the members for the testing patterns.
3. The performance of the 5–21–21–21 architecture was better than that of 5–21–21 except in the case of zone 3 for a few samples (Tables 3a–c). The percentage errors in the 5–21–21–21 case is higher than 5–21–21 in the case of zone 3. This can be either due to the reason of overgeneralization during training or else the testing patterns might have not fallen in the class of the training space.
4. The learning parameter as 0.9 and momentum parameter as 0.7 recommended by Pao [6] was quite appropriate in the current context.
5. The iterations taken by networks for various zonal identifications are given in the Table 4. It is usually found that the iterations taken by two hidden layers networks were more than those by one hidden layer networks.

Table 3. Typical samples of zone 3

Member number	FEM solution ($\times 10^2$)	Results obtained from trained NN architecture 5–21–21		Results obtained from trained NN architecture 5–21–21–21	
		ASE = 0.01 ($\times 10^2$)	Error (%)	ASE = 0.01 ($\times 10^2$)	Error (%)
(*a*) *Sample* 1					
4	0.1078400	0.1263630	17.1763668	0.1290590	19.6763725
12	0.1110600	0.1297060	16.7891178	0.1327190	19.5020657
13	0.1142800	0.1327690	16.1786861	0.1369070	19.7996101
14	0.1174900	0.1315870	11.9984655	0.1320800	12.4180803
15	0.1207100	0.1344400	11.3743725	0.1391450	15.2721405
20	0.1239200	0.1132720	8.5926447	0.1228120	0.8941238
21	0.1271400	0.1189660	6.4291348	0.1250040	1.6800431
(*b*) *Sample* 2					
4	0.1400100	0.1457230	4.0804234	0.1643140	17.3587627
12	0.1432200	0.1490650	4.0811305	0.1678790	17.2175617
13	0.1464400	0.1525330	4.1607456	0.1716270	17.1995354
14	0.1496500	0.1563530	4.4791212	0.1652250	10.4076233
15	0.1528700	0.1604190	4.9381843	0.1680720	9.9443979
20	0.1560900	0.1310860	16.0189629	0.1436540	7.9671993
21	0.1593000	0.1328240	16.6202106	0.1409030	11.5486526
(*c*) *Sample* 3					
4	0.1721700	0.1596240	7.2869854	0.1903970	10.5866280
12	0.1753800	0.1629260	7.1011534	0.1936970	10.4441776
13	0.1786000	0.1666740	6.6774888	0.1968790	10.2346029
14	0.1818200	0.1744030	4.0793138	0.1894660	4.2052550
15	0.1850300	0.1793830	3.0519397	0.1886610	1.9623822
20	0.1882500	0.1438900	23.5644150	0.1582390	15.9420967
21	0.1914700	0.1425360	25.5570049	0.1517690	20.7348404
(*d*) *Sample* 4					
4	0.1008200	0.1211220	20.1368847	0.1199360	18.9605236
12	0.1040400	0.1244520	19.6193829	0.1235690	18.7706642
13	0.1072500	0.1274030	18.7906837	0.1278010	19.1617756
14	0.1104700	0.1249700	13.1257353	0.1234060	11.7099705
15	0.1136800	0.1275090	12.1648493	0.1313680	15.5594645
20	0.1169000	0.1084510	7.2275414	0.1171190	0.1873416
21	0.1201200	0.1151350	4.1500144	0.1205670	0.3721321

6. The CPU time required for training the networks on VAX8810 was less than 60 min in all the exercise undertaken here.

GUIDELINES FOR ARCHITECTURE BASED ON THE EXERCISE

Input nodes and output nodes: number of facts per training example generally decide the number of input nodes and corresponding output parameter would give the number of output nodes.

Hidden layers and hidden nodes: for choosing the number of hidden layers, it is always advisable to go for more than one hidden layer provided the number of connections is not constant. For the number of hidden nodes per hidden layer, make the hidden layer size as either input or output layer, whichever is maximum.

Learning parameter η and momentum parameter α: select learning parameter η as large as possible. The learning parameter η as 0.9 and momentum parameter α as 0.7 is quite appropriate in such applications.

Average system error: after pilot study on various values of average system error, keeping the balance between CPU time required for training and accuracy in the results, decide the suitable ASE.

Table 4. Number of iterations taken to train the networks

	Neural network architecture	
	5–21–21	5–21–21–21
Zone 1	1502†	2181†
Zone 2	1314†	1015†
Zone 3	387†	525†

† CPU time taken by networks during training was less than 60 min on a VAX8810

Learning parameter $\eta = 0.9$.
Momentum parameter $\alpha = 0.7$.
Average system error = 0.01.
Total number of training sets used for each zone = 40.

CONCLUSIONS

A multilayer perceptron architecture with back-propagation learning algorithm has been adopted to model a typical bridge truss with simulated damaged states. The training patterns were generated using general structural analysis program with assumed zones of damage in the structure. The working of the network was demonstrated by comparing the output with the algorithmically generated performance parameters not considered in the training. The issues

related to the performance of the network have been examined. From the observation it is concluded that the perceptron model is quite appropriate for the structural damage identification. For the case illustrated, the performance of the architecture with two hidden layers was better than that of a single layer. The engineering significance of the investigation is reflected from the fact that the measured data at only a few locations in the structure is needed to train the network for the identification exercise.

REFERENCES

1. J. T. P. Yao, *Safety and Reliability of Existing Structures*. Pitman (1985).
2. P. Hajela and F. J. Soeiro, Recent developments in damage detection based on system identification method. *Struct. Optimiz.* 1–10 (1990).
3. M. Sanyayei and O. Onipede, Damage assessment of structures using static test data. *AIAA Jnl* **29**, 1174–1179 (1991).
4. R. P. Lippman, An introduction to computing with neural nets. *IEEE ASSP Magazine* 4–22, April (1987).
5. T. Kohonen, State of the art in neural computing. *IEEE ICNN* **I**, 79–90 (1987).
6. Yoh-Han Pao, *Adaptive Pattern Recognition and Neural Networks*. Addision-Wesley, Reading, MA (1989).
7. P. D. Wasserman, *Neural Computing: Theory and Practice*. Van Nostrand Reinhold, New York (1990).
8. J. E. Dayoff, *Neural Networks Architecture: An Introduction*. Van Nostrand Reinhold, New York (1990).
9. P. C. Pandey and S. V. Barai, Damage assessment of steel bridges. *J. Struct. Engng* **20**, 9–21 (1993).
10. Z. P. Szewczyk and P. Hajela, Neural networks based damage detection in structures. Technical Report, RPI, Troy (1992).
11. X. Wu, J. Ghaboussi and J. H. Garrett, Use of neural networks in detection of structural damage. *Comput. Struct.* **42**, 649–659 (1992).
12. D. E. Rumelhart, G. E. Hinton and R. J. Williams, Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*. Vol. 1: *Foundations* (Edited by D. E. Rumelhart and J. L. McCelland), pp. 318–362. MIT Press, Cambridge, MA (1986).
13. J. Ghaboussi, J. H. Garrett and X. Wu, Knowledge-based modeling of material behaviour with neural networks. *J. Engng Mech., ASCE* **117**, 132–153 (1991).
14. X. Wu, J. H. Garrett and J. Ghaboussi, Representation of material behaviour: neural networks-based models. *IEEE IJCNN* **I**, 229–234 (1990).
15. P. Hajela and L. Berke, Neurobiological computational models in structural analysis and design. *Comput. Struct.* **41**, 657–667 (1991).
16. M. F. Elkordy, K. C. Chang and G. C. Lee, Neural Networks trained by analytically simulated damaged states. *J. Computing in Civil Engng, ASCE* **7**, 130–145 (1993).

APPENDIX I: GENERALIZED DELTA RULE ALGORITHM [6, 12]

The net input to a node in layer j is given by (Fig. 3)

$$net_j = \sum_{i=0}^{m} w_{ji} o_i \tag{A1}$$

and the output of node j will be

$$o_j = f(net_j). \tag{A2}$$

Here f is the activation function and in this study following given sigmoidal function has been used

$$o_j = \frac{1}{1 + e^{-(net_j + \theta_j)/\theta_0}}. \tag{A3}$$

Now the input to the nodes of layer k is

$$net_k = \sum w_{kj} o_j \tag{A4}$$

and its respective outputs are

$$o_k = f(net_k). \tag{A5}$$

In the training process of neural networks, for the input pattern $x_p = i_{pi}$ the weights adjustment will take place in the links of the neural networks for desired output t_{pk} at the output nodes. After achieving this first adjustment the network will pick up another pair of x_p and t_{pk}, and will again adjust weights for new pair. Similar way the process will go on till all the input–output pairs get exhausted. Finally network will have a single set of stabilized weights satisfying all the input–output pairs.

Usually the outputs o_{pk} will not be the same as the desired output values t_{pk}. For each input–output pattern, the square of the error can be given by

$$E_p = \frac{1}{2} \sum_k (t_{pk} - o_{pk})^2 \tag{A6}$$

and the average system error by

$$E = \frac{1}{2P} \sum_p \sum_k (t_{pk} - o_{pk})^2. \tag{A7}$$

Avoiding the p subscript in the eqn (A6) for convenience, then the expression will be

$$E = \frac{1}{2} \sum_k (t_k - o_k)^2. \tag{A8}$$

In a true gradient search for a minimum system error one has to compute the derivative of the error function E. with respect to any weight in the network and then change the weights according to the rule

$$\Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}}, \tag{A9}$$

where η is learning parameter.

The partial derivative $\partial E/\partial w_{kj}$ can be given by using chain rule

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial w_{kj}}. \tag{A10}$$

Using eqn (A4)

$$\frac{\partial net_k}{\partial w_{kj}} = \frac{\partial}{\partial w_{kj}} \sum w_{kj} o_j = o_j \tag{A11}$$

now δ_k can be defined by

$$\delta_k = \frac{\partial E}{\partial net_k} \tag{A12}$$

therefore

$$\Delta w_{kj} = \eta \delta_k o_j. \tag{A13}$$

The weights on each line should be changed by an amount proportional to the product of the term δ_k, available to the unit receiving input along that line and the activation o_j

along that line. The determination of δ_k is a recursive process. To compute $\delta_k = -\partial E/\partial net_k$, the chain rule can be used to express in terms of two factors. First the rate of change of error with respect to the output o_k and second, the rate of change of the output of the node k with respect to input to that same node. Therefore

$$\delta_k = -\frac{\partial E}{\partial net_k} = -\frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial net_k}. \tag{A14}$$

Now these factors can be computed as

$$\frac{\partial E}{\partial o_k} = -(t_k - o_k) \tag{A15}$$

$$\frac{\partial o_k}{\partial net_k} = \tilde{f}_k(net_k). \tag{A16}$$

Using expressions (A15) and (A16), we have

$$\delta_k = (t_k - o_k)\tilde{f}_k(net_k). \tag{A17}$$

For any output layer k, Δw_{kj} will be given by

$$\Delta w_{kj} = \eta(t_k - o_k)\tilde{f}_k(net_k)o_j = \eta\delta_k o_j. \tag{A18}$$

Similarly for the internal units

$$\Delta w_{ji} = \eta\delta_j o_i \tag{A19}$$

$$\delta_j = \tilde{f}_j(net_j)\sum_k \delta_k w_{kj}. \tag{A20}$$

The application of the backpropagation algorithm involves two phases. In the first phase the input is presented and propagated forward through the network to compute the output value of each unit. In the backward phase the δs for all the units are computed. Once these two phases are complete, one can compute for each weight the Δws.

In summary here we add one more subscript p to denote the pattern number, we have

$$\Delta_p w_{ji} = \eta\delta_{pj} o_{pi}. \tag{A21}$$

If j are the nodes of the output layer then

$$\delta_{pj} = (t_{pj} - o_{pj})\tilde{f}_j(net_{pj}) \tag{A22}$$

or if j are nodes of internal or hidden units then

$$\delta_{pj} = \tilde{f}_j(net_{pj})\sum_k \delta_{pk} w_{kj}. \tag{A23}$$

The backpropagation is basically a gradient descent algorithm. In multilayer networks, the error surfaces will be complex with several local minima. It is possible that the gradient descent procedure may not reach the global minimum, but get trapped in one of the many local minima.

One way to increase the learning rate without leading to oscillation is to modify the backpropagation algorithm by including the momentum term α as below

$$\Delta w_{ji}(n+1) = \eta(\delta_j o_i) + \alpha\Delta w_{ji}(n), \tag{A24}$$

where n is the presentation number and the α the constant that determines the effect of the previous weight changes on the current direction of movement in the weight space. This provides a kind of momentum in the weight space that effectively filters out the high frequency variations of the error surface in the weight space.

Vol.3, Issue 2(2011)30-36
International Journal of Engineering & Applied Sciences (IJEAS)

A NEW METHOD FOR DAMAGE DETECTION IN SYMMETRIC BEAMS USING ARTIFICIAL NEURAL NETWORK AND FINITE ELEMENT METHOD

F. Nazari[1], and S. Baghalian[2]*

[1]Mechanical Engineering Department, Islamic Azad University, Hamedan Branch, Young Researchers Club, Hamedan, Iran.

[2] Civil Engineering Department, Islamic Azad University, Hamedan Branch, Young Researchers Club, Hamedan, Iran.
[*]E-mail address: f.nazari@iauh.ac.ir

Received Date: 02 January 2011; Accepted Date: 11 February 2011

Abstract

*In this paper a new method for crack detection in symmetric beams is presented. Natural frequency is frequently used as a parameter to detect cracks in structures. In symmetric structures, it isn't possible to identify the location and the depth of a crack using only the natural frequencies. This is due to the fact that the natural frequencies for any symmetric position of a crack with respect to symmetry plane of the structure are the same. In this research it is assumed that the structure is a rectangular beam which is fixed at both ends. **F**inite **E**lement **M**ethod (**FEM**) was used to obtain natural frequencies of beam in different conditions of cracks. Then assumed the crack is located at the right side of the structure. Based on data were obtained from FEM, two distinct **A**rtificial **N**eural **N**etworks (**ANN**s) were trained for crack location and depth detection in some different conditions and then were tested. As it was assumed that the crack is at the right side of the beam, two symmetric positions could exist for a crack. Finally using an algorithm based on first vibrational mode shape of structure, locations and depths of cracks have been identified with good approximations.*

Keywords: Crack Detection; Symmetric Beam; Mode Shape; Natural Frequency; Artificial Neural Network

1. Introduction

Crack is one of the common faults that if develops, may cause catastrophic damages in structures. A lot of studies have been done on non-destructive estimation methods. The non-destructive methods used so far can be divided in four groups. The first group includes methods that determine if there is a specific fault in the structure or not [1, 2]. The second group includes methods not only capable of identifying the fault but also locating it [3-5]. The third group includes methods capable of specifying more information about the fault, like the depth [6-9], and the fourth group contains methods that can even estimate the effect of the fault on the structure. In recent years investigators have shown great interest in vibration analysis method and so there are a lot of investigations in this area. Dimarogonas reviewed methods of investigating cracked structures in 1996 [10]. Crack causes a local flexibility in the structure which affects the dynamic behaviour. For example it reduces the natural frequencies and changes the mode shapes. Analyzing these effects can be used for crack detection [11]. Dimarogonas et al. modeled a crack using local flexibility and calculated the equivalent stiffness utilizing fracture mechanics [12, 13]. Adams et al. developed an experimental technique to estimate the location and the depth of a crack based on the changes of the natural frequencies [14]. In another investigation Dimarogonas presented methods which relate the depth of the crack to the changes of the natural frequencies when the crack location is known [15]. These methods can be used to identify cracks in different structures. Gudmunson presented a method to predict the changes of the natural

frequencies caused by faults such as cracks, notches, etc [16]. Masoud et al. investigated vibrational characteristics of a fixed-fixed beam with a symmetric crack considering coupling effect of crack depth and axial load [17]. Shen et al. presented a method based on minimizing the difference between the measured data the data obtained from an analytical data to identify cracks in an Euler-Bernoulli beam [18]. In this paper the parameter used to identify the fault is natural frequency. This is because of the fact that measuring natural frequency is cost effective [20], and can be done accurately in most structures [11].

A new technique used frequently for damage identification in recent years is neural network. Wu et al. used back propagation neural network to identify the fault location in a simple frame [21]. Kao and Hung presented a two step method for identifying cracks based on neural network. First step was to identify damaged and undamaged system situations. Second step was fault detection in structures. In this step a trained neural network was used to produce free vibration response of the system and finally a comparison was made between the results to evaluate changes in amplitude and periods [22]. Chen et al. worked on using neural network for fault detection in engineering structures in case the excitation signal is not available [23].
In this paper back error propagation neural network is used to estimate natural frequencies in different crack locations and crack depths of symmetric beams.

2. Modal Analysis

The structure investigated in this paper is a beam fixed at both ends with an open crack (Fig. 1).

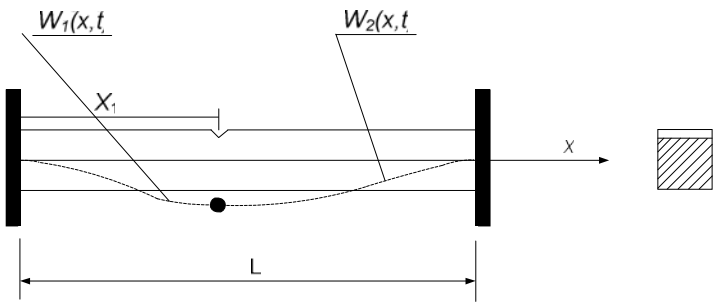


Fig. 1. Schematic of clamped-clamed beam with an open crack

Table 1 gives the beam's material and geometrical characteristics. Since the structure is symmetric with respect to a plane passing through the center and normal to the *x* direction, for a crack located at a distance of *x* from any of the two supports, there are two possible conditions with the same value of natural frequencies. In this section it is assumed that the crack is located at the right side of the beam. Three natural frequencies of the structure are calculated for different conditions of crack locations and depths using finite element method. The natural frequencies of the considered cracked beam were calculated by using modal analysis. In this article the 2D elements have been used for modal analysis. This element is defined by eight nodes which have two degrees of freedom at each node include translations in the perpendicular directions. A meshed view of the typical structure in crack region has been illustrated in Fig. 2.

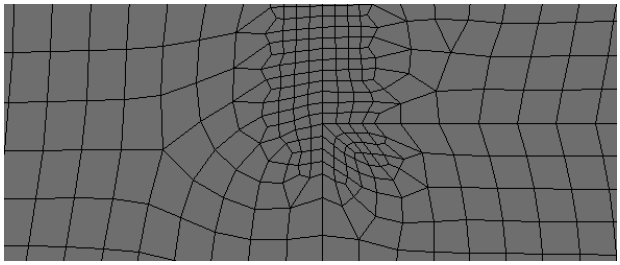


Fig. 2. Typical finite element discretisations.

Table 1. Beam Characteristics

Density (Kg/m3)	Poisson Ratio	Elasticity Modulus (GPa)	Length (mm)	Thickness (mm)	Depth (mm)
7860	0.3	210	175	12.5	25

3. Artificial Neural Network

Nowadays Artificial neural networks are used in many engineering applications. The most popular type of neural network is the **M**ulti-**L**ayer **F**eed **F**orward (**MLFF**). A schematic diagram of a typical MLFF neural network architecture is depicted in Fig. 3.

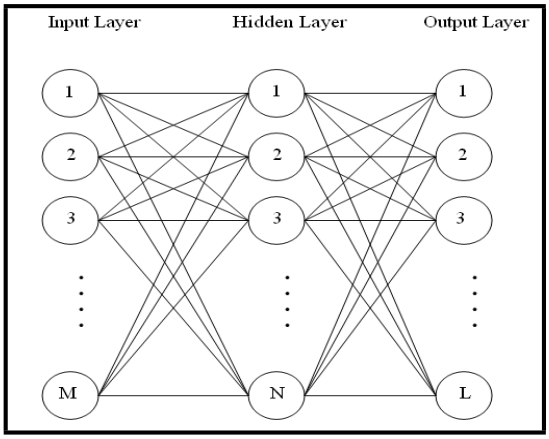


Fig. 3. Schematic diagram of typical MLFF neural network

The network usually consists of an input layer, some hidden layers and an output layer. Usually knowledge is stored as a set of connection weights. The process of modifying the connection weights, in some orderly fashion, uses a suitable learning method called training.

The back-error propagation is the most widely used learning algorithm. It is one of the most powerful learning algorithms in neural networks. The back-propagation neural network was proposed by McClelland and Rumelhart [25] in a ground-breaking study originally focused on cognitive computer science. In this paper the structure of neural network includes three layers: the input layer, hidden layer, and output layer. The variable M shows the total neuron number in the input layer, variable N shows the total neuron number in the hidden layer, and the variable L shows the total neuron

number in the output layer. Values w_{MN} are the weights between the input and the hidden layer. Values w_{LN} are the weights between the hidden and the output layer. The operation of back error propagation is consisting of three steps:

1- Feed-forward step:

$$v_j \quad w_{LN}(n)u_{j\ 1}(n); \tag{1}$$

$$o_j(n) \quad (v_j(n)) \quad \frac{2}{1 \quad \exp(\ v_j(2n))}; \tag{2}$$

Where, o_j is output, u_j is input, $u_{j\ 1}$ is output of hidden layer and is a transfer function.

2- Back-propagation step:

$$_j(n) \quad e_j(n).\ '(v_j(n)) \quad (d_j(n) - o_j(n))o_j(n)(1 - o_j(n)); \tag{3}$$

Where, $_j$ represents the local gradient function, e_j shows the error function, o_j means the actual output and d_j is desired output.

3- Adjust weighted value:

$$w_{NM}(n \quad 1) \quad w_{NM}(n) \quad w_{NM}(n) \quad w_{NM}(n) \quad _j(n).o_j(n); \tag{4}$$

where is the learning rate. Repeating these three steps results to the value of the error function will be zero or a constant value.

In this paper two distinct neural networks are employed for prediction of locations and depths of cracks. Networks consist of an input layer, a hidden layer and an output layer with 4, 20 and 1 neurons, respectively. In each network, transfer functions for neurons of hidden and output layers are Tansig and Purline, and are defined as equation (5) and (6) respectively.

$$f(x) \quad \frac{2}{(1 \quad \exp(-2x)) - 1}. \tag{5}$$

$$f\ (x) \quad x. \tag{6}$$

4. Crack Detection Procedure

In this section three different natural frequencies from two different crack conditions are used to train the network as the input and the corresponding locations and depths are obtained as the output. These data are based on the assumption that the crack is located at the right side of the beam. It should be noted that a crack located at the left side of the beam gives the same output. In order to determine the original location of the crack, beam deflection is calculated at two arbitrary symmetrical points located at the right and the left side of the beam. The original crack location is at the side which has the bigger value of deflection. It happens because the crack lead to local flexibility in its surrounding [11], therefore in the same distances from two end of symmetric beam, the side that cracked is there,

has more deflection rather than other side. This matter has been proved with FEM analysis of considered beam that is tabulated in Table 3.

5. Results

Table 2 compares first four natural frequencies of structure that obtained from modal analysis of present study and the analytical results that presented in reference [24]. In should be mentioned that, in Table 2, f_i represent the i^{th} natural frequency of cracked beam.

Table 2. Comparison of the natural frequencies obtained from present study and Ref. [24].

Crack Location (mm)		35	70
Crack Depth (mm)		12.5	12.5
Present Work (HZ)	f_1	3823.1	3483.4
	f_2	8776.6	8787.8
	f_3	15785	15700
	f_4	23203	21366
Reference [24] (HZ)	f_1	3819.1	3395.3
	f_2	8743.6	8679.5
	f_3	14691.1	15474.0
	f_4	22849.1	21518.7
Error (%)	f_1	0.1	2.6
	f_2	0.4	1.2
	f_3	7.4	1.4
	f_4	1.5	0.7

Table 3 shows the deflection values of structure in the considered point at first vibrational mode shape of beam that was obtained for two different crack locations. The corresponding predicted values of crack locations and depths are compared with the actual values in Table 4. As has been shown in this Table, the predicted crack characteristics are in good agreement with actual data, which the maximum errors were 1.3% and 1.2% for depth and location of crack, respectively. Therefore it can be concluded that the presented method of this paper is suitable for crack detection in symmetric beams. Also it should be mentioned that the presented method can be applied to many other symmetric cracked structures.

Table 3. Deflections of two symmetric points

Number	1	2
Depth (mm)	1.1607	1.5
Location (1) (mm)	122.5	122.5
Deflection (1) (mm)	1.8099	1.6284
Location (2) (mm)	52.5	52.5
Deflection (2) (mm)	1.6123	1.7878

Table 4. Comparison of predicted and actual depth and location of crack

Number	1		2	
Parameter	Depth	Location	Depth	location
Predicted Results (mm)	8.86	122.41	3.0	51.87
Actual Results (mm)	8.75	122.5	3	52.5
Errors (%)	1.3	0.7	0	1.2

6. Conclusion

This paper presented a new method for crack identification in symmetric beams. In this study, first of all, it was determined that identifying the crack location in a symmetric beam with an open crack is not possible knowing only the natural frequencies. Two distinct ANNs were employed for prediction of locations and depths of cracks in half of beam. Then an algorithm based on calculating the beam deflections at two arbitrary symmetric points on first vibration mode shape was presented to identify the original location of the crack. Then it was showed that the predicted crack characteristics were in good agreement with actual data, which the maximum errors were 1.3% and 1.2% for depth and location of crack, respectively. Finally it was concluded that the presented method of this study is suitable and useful for crack detection in symmetric beams.

7. References

[1] Vandive, J.K., Detection of structural failures on fixed platforms by measurement of dynamic responses, Proceedings of the 7th Annual Offshore Technology Conference, Houston, Texas, 1975.
[2] Crohas, H. and Lepert, P., Damage detection monitoring method for offshore platforms is field tested, Oil& Gas J., 80, 94-103, 1982.
[3] Cawley, P. and Adams, R.D., The location of defects in structures from measurements of natural frequencies, J. Strain Anal., 14, 49-57, 1979.
[4] Pandey, A.K., Biswas, M. And Samman, M.M., Damage detection from changes in curvature mode shapes, J. Sound Vib, 145, 321-332, 1991.
[5] Chance, J., Tomlinson, G.R. and Worden, K., A simplified approach to the numerical and experimental modeling of the dynamics of a cracked beam, Proceedings of the 12th International Modal Analysis Conference, Honolulu, Hawaii, 1994.
[6] Stubbs, N. And Osegueda, R., Global non-destructive damage evaluation in solids, Int. j. analyt. exp. modal anal., 5, 67-79, 1990.
[7] Wu, X., Ghaboussi, J. and Garrett, J. H., Use of neural networks in detection of structural damage, Comput. Struct., 42, 649-659, 1992.
[8] Kaouk, M. and Zimmerman, D.C., Structural damage assessment using a generalized minimum rank perturbation theory, AIAA J., 32, 836–842, 1994.
[9] Kim, J.T. and Stubbs, N., Model uncertainty and damage detection accuracy in plate-girder bridges, J. Struct. Eng., 121, 1409-1417, 1995.
[10] Dimarogonas, A.D., Vibration of cracked structures: a state of the art review, Eng. Fract. Mech., 55, 831-857, 1996.
[11] Douka, E., Loutridis, S. and Trochidis, A., Crack identification in beams using wavelet analysis, Int J Solids Struct., 40, 3557-3569, 2003.
[12] Dimarogonas, A.D., Vibration Engineering, West Publishers, St Paul, Minesota, 1976.
[13] Paipetis, S.A. Dimarogonas, A.D., Analytical method in rotor dynamics, Appl. Sci. London,1986.

[14] Adams, A.D. and Cawley, P., The location of defects in structures from measurements of natural frequencies, J. Strain Anal., 14, 49-57, 1979.
[15] Chondros, T.G. and Dimarogonas, A.D., Identification of cracks in welded joints of complex structures, J. Sound Vib., 69, 531-538, 1980.
[16] Goudmunson, P., Eigenfrequency change of structures: a state of the art review, Eng. Frac. Mech., 55, 831-857, 1982.
[17] Masoud, A., Jarrad, M.A. and Al-Maamory, M., Effect of crack depth on the natural frequency of a prestressed fixed-fixed beam, J. Sound Vib., 214, 201-212, 1998.
[18] Shen, M.-H. and Taylor, J.E., An identification problem for vibrating cracked beams, J. Sound Vib., 84, 150-457, 1991.
[19] Liang, R.Y., Hu, J. and Choy, F., Theoretical study of crack-induced eigenfrequency changes on beam structures, J. Eng. Mech., 118, 384-396, 1992.
[20] Kim, J.-T. and Stubbs, N., Crack detection in beam-type structures using frequency data, J. Sound Vib., 259, 145–160, 2003.
[21] Wu, X., Ghaboussi, J. and Garret, J.H., Use of neural networks in detection of structural damage, Comput. Struct., 42 , 649-659, 1992.
[22] Kao, C.Y. and Hung, S.L., Detection of structural damage via free vibration responses generated by approximating artificial neural networks, Comput. Struct., 81, 2631-2644, 2003.
[23] Chen, Q., Chan, Y.W. and Worden, K., Structural fault diagnosis and isolation using neural networks based on response-only data, Comput. Struct., 81, 2165-2172, 2003.
[24] Khaji, N. Shafiei, M. And Jalalpour, M., Closed-form solutions for crack detection problem of Timoshenko beams with various boundary conditions, Int. J. Mech. Sci., 51, 667-681, 2009.
[25] McClelland, J.L. and Rumelhart, D.E., Parallel distributed processing: Explorations in the Microstructure of Cognition, Volumes I and II, MIT Press, Cambridge, Massachusetts, USA, 1986.

Damage identification in laboratory cantilever beam using neural networks

Artur Borowiec* and Leonard Ziemiański
Department of Structural Mechanics, Rzeszów University of Technology
W. Pola 2, 35-959 Rzeszów, Poland
email: artur.borowiec@prz.edu.pl, ziele@prz.edu.pl

Abstract

This paper presents the application of Artificial Neural Networks (ANN) in the identification of damage in a simple laboratory beam structure. The application of ANNs extends the nondestructive damage identification method by adding parameter to the structure. Damage is identified based on the variations in the dynamic parameters without knowledge of the initial values of the undamaged structure. In the experimental examples ANNs are applied for the analysis of the dynamic response of beam, in order to locate the damage and its extent. The assessment of the state of the structure relies on comparison of the structure eigenfrequencies obtained from the systems with additional masses placed at different nodes.

Keywords: beams, damage, dynamics, inverse problems, neural networks

1. Introduction

Nowadays knowledge of the condition of the structure is considered to be more and more important. The state of the structure and its safety depend to a great extent on the degradation of the structure elements (beams, connections, etc.). Nondestructive methods predict the location and extent of damage in existing engineering structures. Publications on the identification of damage present mainly the approach which implies that the eigenfrequencies and eigenmodes of the undamaged structure are known. Damage is identified based on the variations in the dynamic parameters with respect to the initial values [1]. Some methods require the introduction of external perturbations to the structure. The detection method which provides the global assessment of damage is usually not sensitive to the extent of damage. In the paper by Dems [2] an additional parameter is introduced (e.g concentrated elastic or rigid support, additional mass elastically or rigidly attached to the structure, boundary constraint) in order to increase the accuracy of identification. In the paper by Zhong [3] response-only method for damage detection of beam-like structures using high accuracy frequencies with auxiliary mass spatial probing is presented. This method does not require knowledge of the initial values of the undamaged structure to located damage.

The present paper is intended to provide the analysis of eigenfrequencies with respect to an additional mass and the application of ANNs to damage identification.

1.1. Damage identification proceedings

The scheme of damage identification proceedings is shown in Fig. 1. Basically, we can distinguish two main stages. In the first stage a numerical model (FEM) of the structure considered was built. Then for the selected locations of damage (l_c) and for the selected extent of damage (h_c) with mass M added at different nodes, the dynamic parameters of the structure (${}^m\boldsymbol{f}_n$) were computed. The results obtained were used for training ANNs. At this stage the number and the best location of mass (l_m) as well as the best eigenfrequencies ($\boldsymbol{f}_n$) were analyzed to improve the identification results. In the second stage the response of the structure (${}^m\tilde{\boldsymbol{f}}_n$) was measured. The vibrations of the structure were caused by an impact. The results obtained were used as the input vector $X = \{{}^m\tilde{\boldsymbol{f}}_n\}$ for ANNs trained on numerical results. The output vector $Y = \{l_c, h_c\}$ was described by the location and extent of the damage.

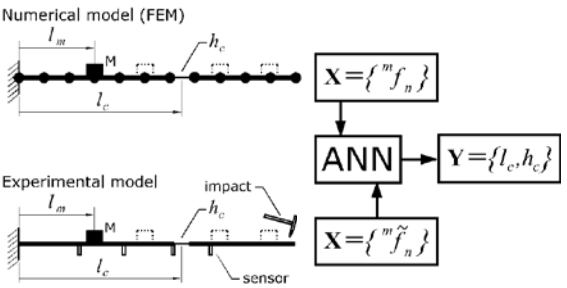


Figure 1: Scheme of damage identification proceedings

In this paper only the data measured has been considered. That approach verifies the quality of proper neural prediction with the experimental data. The model only one type of damage at a time was considered. Changes of eight eigenfrequencies were observed after adding mass and reducing stiffness. The data measured was used for learning and testing ANNs.

2. Experimental model of beam

The model of cantilever beam was measured. The model was made of flat steel (S235JRG2) 10 mm × 40 mm and 1200 mm length. The damage of the beam was done by the notch 1.2 mm wide and 1.0 mm to 8.0 mm deep. After measuring the notch was welded and ground. The additional mass M = 0.2 kg represented about 5% of the total mass of the models. The number and the best location of the mass were analyzed to improve the results with an acceptable minimal level of error. Vibrations of the models were caused by an impact. The response of the beam was measured in the range of 0.25 Hz to 1024.0 Hz with the step 0.25 Hz. The measurements were done using eight PCB accelerometers attached to the models connected to a multichannel analyser Scadas III with LMS software.

The model was formally divided into 24 parts (Fig. 2). Each part was 50 mm long. Eight of them (triangles A–H) were cut in

*Financial support by the Polish Science Committee, Grant N N501 134336 is gratefully acknowledged.

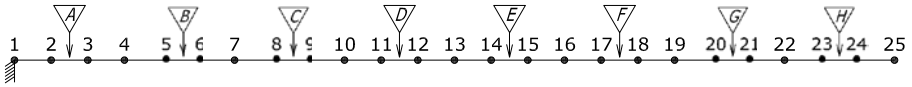


Figure 2: Scheme of notch (triangles A–H) on the laboratory model

the middle. For two notches (A, B) the beam was broken after the deepest cut (8.0 mm) had been made. On the whole 62 cases of damages were studied. The additional mass was attached at 23 points on the beam in turn (nodes 2–24 on Fig. 2). The experiment involved 1426 measurements. As a result 62 patterns to train ANN were obtained.

3. Application of neural networks

All the neural networks computations were performed using Neural Network Toolbox for Matlab. The input vector $X = \{{}^{m}\tilde{f}_n\}$ consisted of the dynamic responses of the structure with an additional mass. The output vector $Y = \{l_c, h_c\}$ consisted of the location and extent of damage.

The ANN with one hidden layer was applied and the Levenberg-Marquardt method was used in the training procedure. In each case 30% of the patterns were selected as testing ones, the remaining 70% were regarded as training ones. In all the cases, comparison of MSE (Mean Square Error) of testing was used to improve the results with acceptable minimal level of error.

4. Results

The combinations of a set of one, two or three best eigenfrequencies (f_n) with one, two or three best positions of mass (l_m) were analyzed. In all the combinations the identification of extent was better than the identification of location. The best results were obtained for ANN simulations using the input vector $X = \{{}^{7}f_3, {}^{21}f_3, {}^{22}f_3, {}^{7}f_7, {}^{21}f_7, {}^{22}f_7\}$.

The results for the identification of the location of damage are shown in Fig. 3. The figure shows the values predicted from the networks versus the target values. In the case of the proper neural prediction the points are on the diagonal. The results from learning (circle) and testing (triangle) are shown. The Mean Square Errors and the correlation coefficient are also shown.

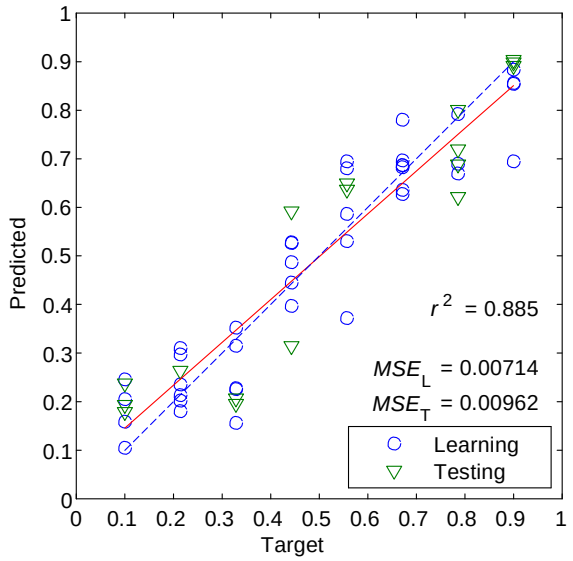


Figure 3: Prediction of the location of damage

The results for the identification of the extent of damage are shown in Fig. 4. These results were not excellent but very promising due to used resolutions of measured (0.25 Hz).

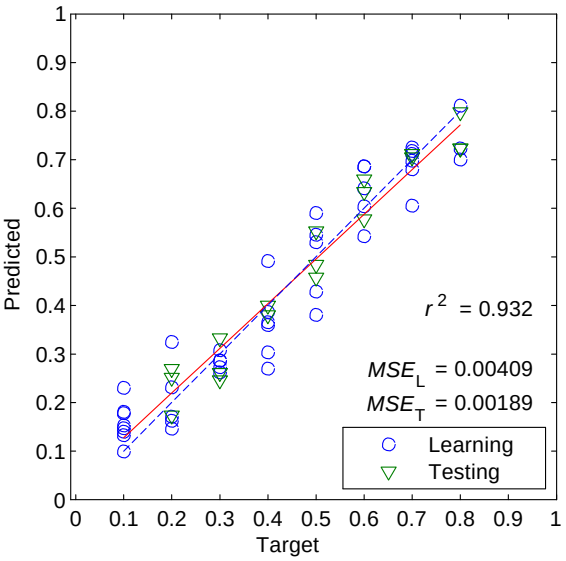


Figure 4: Prediction of the extent of damage

However using the response-only method [3] of the location of damage only one deepest type of damage (Fig. 2 point C) was definitely located.

5. Conclusions

In the proposed method the damage is identified on the basis of the variations in the dynamic parameters without knowledge of the initial values of the undamaged structures. The application of ANNs improves the nondestructive damage identification method the method uses an additional parameter introduced to the structure which increases the identification accuracy. The results obtained show that it is possible to identify damage using the dynamic responses of structures. Artificial Neural Networks are able to identify the location and extend of damage in structures.

References

[1] S.W. Deobeling, C.R. Farrar, M.B. Prime, D.W. Sheritz, *Damage identification and health monitoring of structural and mechanical systems from changes in their vibration characteristic: a literature review*, Los Alamos Natl. Lab, 1996.

[2] K. Dems, Z. Mróz, Identification of damage in beam and plate structure using parameter-dependent frequency changes. *Engineering Computations*, 18, pp. 96–120, 2001.

[3] S. Zhong, S. O. Oyadij, K. Ding. Response-only method for damage detection of beam-like structures using high accuracy frequencies with auxiliary mass spatial probing. *Journal of Sound and Vibration*, 311, pp. 1075–1099, 2008.