A la mémoire de mon père.

Remerciements

Les travaux de recherche présentés dans ce mémoire ont été réalisés au laboratoire EOLE du département de génie civil de Tlemcen et sous la direction de Monsieur F. GHOMARI. Je tiens à le remercier vivement pour m'avoir fait l'honneur de diriger ce mémoire, pour la confiance qu'il m'a témoignée et l'intérêt qu'il a porté à ces travaux.

Monsieur A. BEKKOUCHE m'a fait l'honneur d'accepter de présider le jury de ma soutenance, je lui présente mes remerciements les plus distingués.

Mes remerciements s'adressent également à Messieurs D. KERDAL, A. MEGNOUNIF et A. DJEDID pour avoir accepté d'examiner ce mémoire. Qu'ils trouvent ici l'expression de ma profonde reconnaissance.

Je remercie tout particulièrement mon mari, qui a initié cette recherche et qui, par ses conseils, ses encouragements, sa compréhension et sa disponibilité a très largement contribué à son développement et à ses conclusions.

Je remercie également toute ma famille pour sa patience durant ces derniers mois et pour ses encouragements.

Résumé

Les réseaux de neurones artificiels (RNA) ou réseaux neuromimétiques se veulent le modèle mathématique du système nerveux humain. Il s'agit de systèmes de traitement de l'information qui comportent certaines caractéristiques issues de l'étude des réseaux de neurones biologiques. Les deux grandes classes de problèmes pour lesquelles on utilise les RNA sont : les problèmes de classification et de reconnaissance de formes, et les problèmes d'interpolation. Leur domaine d'application dans le génie civil est très vaste. On peut les employer, par exemple, pour interpoler les lois de comportement de matériaux ou d'éléments de structures obtenues à partir d'essais expérimentaux ou suite à des développements théoriques. Le but est de les incorporer ensuite dans des codes de calcul numérique des structures. Dans ce mémoire de magister, nous voulons les utiliser pour étudier, à titre d'exemple, la flexion simple des poutres de Timoshenko en élasto-plasticité. Le travail consiste à proposer une stratégie d'interpolation des relations constitutives (moment - courbure) élasto-plastiques, à chercher les réseaux de neurones nécessaires à sa mise en œuvre et enfin, à la programmer dans un code de calcul en éléments finis. La comparaison des résultats calculés par le programme élaboré à ceux trouvés dans la littérature et à ceux fournis par le logiciel CAST3M a permis de valider le programme et la stratégie développés. Cette étude a également mis en évidence les grandes capacités d'apprentissage et de généralisation des RNA.

Mots-clés : *Réseaux de neurones artificiels, méthode des éléments finis, non linéarité matérielle, flexion simple, poutre de Timoshenko.*

Abstract

Artificial Neural Networks (ANN) or neuro mimetic networks are the mathematical model of human nervous system. It's about systems of data processing with some characteristics derived from biological neural networks. These ANN are used in two major problem classes: Problems related to the classification and the recognition of shapes, and interpolation problems. In civil engineering field, they can be widely used. We can use them to interpolate materials or structural elements behaviour laws obtained from experimental tests or from theoretical development, with the aim to incorporate them in numerical codes for structures. In this work, we use them to study the elasto-plastic bending of Timoshenko beam. First, we propose an interpolation strategy of the non linear constitutive relations (moment-curvature relationships), then we look for neural networks for its implementation and finally, we program it in a finite element software. To validate the program and the strategy developed, results obtained from the elaborated program are compared with those found in literature and those obtained from CAST3M software. We found very good agreement. This study, also, has displayed the great capacities of ANN learning and generalization.

Keywords: *Artificial neural networks, Finite element method, material non linearity, simple bending, Timoshenko beam.*

Table des Matières

Liste des Figures

Liste des Tableaux

Notations.

Les principales notations utilisées sont présentées ci-dessous, organisées par chapitres. Les autres symboles précisant des détails ou introduits pour les besoins d'une démonstration sont présentés dans le cours du texte. De toute façon, toutes les notations sont définies lors de leur première apparition dans le texte.

Chapitre 2

X_i, Y_i, Z_i	: Cellules de réseaux de neurones.
x_i, y_i, z_i	: Activations des neurones.
w_{ij}	: Poids ou coefficient synaptique de la connexion ij.
P	: Signal d'entrée du neurone Y.
F, f	: Fonctions d'activation.
E	: Erreur quadratique.
t	: Représente l'étape d'apprentissage.
n, m	: nombre de neurones par couche.

Chapitre 3

x, y	: Coordonnées cartésiennes.
$[\sigma]$	: Matrice représentant le tenseur des contraintes.
σ_x	: Composante normale de la contrainte suivant l'axe x.
σ_y	: Composante normale de la contrainte suivant l'axe y.
τ_{xy}	: Composante de la contrainte de cisaillement dans le plan *(xy)*.
N	: Effort normal.
T	: Effort tranchant.
M	: Moment fléchissant.
S	: Section de poutre.
p	: Force par unité de longueur.
p_x, p_y	: Composantes de force sur les axes de coordonnées.

$\theta(x)$: Rotation d'une section droite par rapport à sa position initiale.
y_0 : Position de la ligne moyenne.
$u(x)$: Le déplacement axial de la ligne moyenne.
$\bar{u}(x,y)$: Le déplacement suivant x d'un point (x,y) quelconque.
v : Le déplacement latéral ou flèche de la ligne moyenne.
$\bar{v}$: Déplacement latéral d'un point de la section droite.
ε_x : Déformation longitudinale.
γ_{xy}, β : Distorsion.
E : Module de Young.
ν : Coefficient de Poisson.
G : Module de cisaillement.
χ : La courbure.
I : Moment d'inertie de la section.
σ_0 : Limite élastique d'un matériau à comportement élasto-plastique parfait.
σ_p : Valeur de la contrainte à la rupture.
ε_p : Valeur de la déformation à la rupture.
ε_0 : Limite élastique de la déformation (correspondant à σ_0).
p : Paramètre de ductilité.
ψ : Position de l'axe plastique.
S_{inf} : Zone de contraintes positives de la section totalement plastifiée.
S_{sup} : Zone de contraintes négatives de la section totalement plastifiée.
M_p : Moment plastique.
W_{pl} : Le module de flexion plastique.
η : Variable sans dimension associée à l'ordonnée y.
η_0 : Variable sans dimension associée à l'ordonnée y_0.
ξ : Variable sans dimension associée à l'abscisse z.
e : Variable sans dimension associée à la déformation.
s : Variable sans dimension associée à la contrainte.
e_0 : Variable sans dimension associée à la limite de la déformation élastique.
s_0 : Variable sans dimension associée à la limite élastique de la contrainte.
α : Variable sans dimension associée au module de Young.
n : Variable sans dimension associée à l'effort normal.
m : Variable sans dimension associée au moment fléchissant.
k : Variable sans dimension associée à la courbure.
I_z : Inertie par rapport à l'axe Oz.

Chapitre 4

N_1, N_2 : Fonctions d'interpolation.
$p^{(e)}$: Vecteur force élémentaire apporté par les efforts internes.
$f^{(e)}$: Vecteur force élémentaire apporté par les forces extérieures.
$\varepsilon_f^{(e)}$: Déformation flexionnelle élémentaire.
$B_f^{(e)}$: Matrice reliant les déformations flexionnelles aux DDL élémentaires.
$\varphi^{(e)}$: Vecteur des degrés de liberté élémentaires.
$\varepsilon_s^{(e)}$: Déformation due au cisaillement ou distorsion de l'élément.
$B_s^{(e)}$: Matrice reliant les déformations de cisaillement aux DDL élémentaires.
K_f : Matrice de rigidité flexionnelle.
K_s : Matrice de rigidité de cisaillement.
Δf : Incrément de charge.
$(EI)_T$: Raideur flexionnelle tangentielle.
K_T : Matrice de rigidité tangente.
$\Delta\varphi$: Incrément des degrés de liberté.
$\Delta\chi$: Incrément de courbure.
Δk : Incrément de moment en variable adimensionnelle.
$\Delta\beta$: Incrément de distorsion.
ψ : Vecteur des forces résiduelles.
ΔM : Incrément de moment.
Δm : Incrément de moment en variable adimensionnelle.

Chapitre 1

Introduction générale

1.1 Contexte

Partant de la constatation que les neurones biologiques ne réalisent que des tâches élémentaires alors que le cerveau est doué d'intelligence et d'une fantastique capacité de traitement de l'information, on peut imaginer la perspective qui s'offrirait à celui qui pourrait modéliser leur fonctionnement. Les réseaux de neurones artificiels (RNA) ou réseaux neuromimétiques se veulent justement le modèle mathématique du système nerveux humain. Il s'agit d'une reproduction plus ou moins réaliste des neurones biologiques.

Un RNA est un système de traitement de l'information qui comporte certaines caractéristiques issues de l'étude des réseaux de neurones biologiques. Le traitement de l'information a lieu au sein d'éléments simples appelés neurones ou cellules. Les signaux transitent d'un neurone à un autre par l'intermédiaire d'un lien de communication. Chacun de ces liens possède un coefficient synaptique qui pondère le signal transmis. Chaque neurone applique une fonction d'activation au signal reçu pour déterminer le signal de sortie.

Un réseau est caractérisé par son architecture et par la méthode de mise à jour des coefficients synaptiques. L'architecture fixe le nombre des cellules et leur organisation, définit les liens entre les neurones, et affecte à chaque cellule une fonction d'activation. Lors de l'apprentissage du réseau, ce sont les poids ou coefficients synaptiques qui subissent des changements afin de reproduire le message désiré. La méthode retenue pour optimiser ces coefficients est l'autre caractéristique du RNA.

La complexité du système nerveux naturel fait qu'il ne peut être parfaitement représenté mathématiquement. Plusieurs modèles sont alors nécessaires pour reproduire ses différents aspects, et chaque modèle doit être adapté au type de problème posé. Le choix du type de réseaux neuromimétiques, en fonction du problème à étudier, est alors très important. Les deux grandes classes de problèmes pour lesquelles on utilise les RNA sont : les problèmes de classification et de reconnaissance de formes, et les problèmes d'interpolation. Leur domaine d'application est très vaste. On peut citer l'industrie, les finances, les télécommunications, l'informatique, la médecine, la robotique, …, et le génie civil.

Dès le début des années 90, les ingénieurs civils se sont intéressés aux RNA. Ils les ont utilisés, par exemple, pour déterminer le niveau d'endommagement des membrures de bâtiments ou de ponts réalisées en béton armé ou en acier [13, 14]. On a également employé les RNA à l'identification structurale [14], ou à la prédiction du mouvement des pentes [16]. Au département de génie civil de l'université A. Belkaid de Tlemcen, l'intérêt pour les RNA a commencé à la fin des années 90. On les a alors appliqués pour générer un accélérogramme artificiel compatible avec le spectre de réponse réglementaire des RPA 99 [18] ou pour estimer le risque lié à l'effet de site ou encore pour générer un spectre de réponse à la surface libre [6].

Le domaine d'application des réseaux de neurones artificiels dans le génie civil est donc très vaste. Ils peuvent être utilisés dans tout problème nécessitant une classification ou une interpolation. On peut, en particulier, envisager de les employer pour interpoler les lois de comportement de matériaux ou d'éléments de structures à partir d'essais expérimentaux pour les utiliser dans des codes de calcul numérique des structures. On peut également entrevoir l'éventualité de faire appel à leur grande capacité d'apprentissage pour « apprendre » des formulations théoriques compliquées. L'objectif dans ce cas est de réduire le temps de calcul qui reste l'un des soucis majeurs en calcul et modélisation des structures malgré le développement continu des méthodes et des moyens informatiques. Le fait d'apprendre des solutions théoriques compliquées peut également être vu comme une étape utile précédant l'interpolation de lois expérimentales. En effet, les solutions théoriques étant bien définies, elles peuvent servir comme référence pour arrêter la méthodologie d'utilisation des réseaux de neurones dans les logiciels de calcul des structures. C'est précisément dans ce cadre que s'inscrit ce travail de magister.

1.2 Objectifs

Dans ce mémoire de magister, nous voulons prouver que les réseaux de neurones artificiels peuvent être utilisés efficacement dans un code d'éléments finis en tant qu'outil d'interpolation des lois de comportement. Nous les utiliserons pour étudier la flexion simple des poutres en élasto-plasticité. Le choix des poutres en flexion simple comme exemple d'éléments structuraux n'est théoriquement qu'une étape. L'objectif final est d'arrêter une méthodologie à généraliser aux cas plus compliqués. Ce choix est justifié par le fait que dans la théorie de flexion des poutres, les équations d'équilibre sont exprimées en fonction de résultantes des contraintes, appelées efforts internes ou efforts de réduction, et qui sont le moment de flexion et l'effort tranchant. Les équations constitutives doivent être exprimées en fonctions de ces efforts. Elles prennent alors la forme d'une relation moment – courbure (M-χ), qui est le résultat de l'intégration de la courbe contrainte – déformation (σ-ε). Les expressions (M-χ) théoriques peuvent alors être relativement compliquées même dans le cas de relations (σ-ε) très simples. Leur insertion dans un programme numérique peut s'avérer laborieuse et leur utilisation fastidieuse d'où, l'idée de les utiliser comme exemple à notre objectif en les faisant apprendre à des réseaux de neurones artificiels, réputés pour leur grande capacité d'apprentissage.

La méthode numérique de résolution d'équations différentielles retenue est la méthode des éléments finis dont l'efficacité à traiter les problèmes les plus compliqués n'est plus à démontrer. Parmi les théories de flexion des poutres, nous voulons utiliser celle de Timoshenko. Elle présente l'avantage de prendre en considération la distorsion et permet, par conséquent, d'étudier aussi bien les poutres élancées que les poutres épaisses. Nous ferons

néanmoins une approximation en utilisant les lois constitutives qui négligent toute interaction entre les différents efforts internes, ce qui est bien entendu inexacte.

Le travail consiste donc à proposer une stratégie d'interpolation des relations constitutives, à chercher les réseaux de neurones nécessaires à sa mise en œuvre et enfin, à la programmer dans un code d'éléments finis.

1.3 Plan du mémoire

Le contexte et les objectifs du mémoire étant fixés, nous avons articulé ce mémoire en six chapitres. Le deuxième est une revue bibliographique sur les réseaux de neurones et leurs utilisations en génie civil. Le troisième chapitre rappelle les relations constitutives (M-χ) élasto-plastiques de deux types de sections de poutres. Le quatrième présente la stratégie adoptée et sa mise en œuvre pratique dans un programme d'éléments finis. Le cinquième donne des exemples de validation du programme élaboré et le dernier est la conclusion du travail.

Dans le deuxième chapitre, nous avons commencé par présenter le modèle mathématique des réseaux de neurones artificiels, les différentes fonctions d'activation usuelles, ainsi que les différentes règles d'apprentissage. La règle de la rétropropagation du gradient de l'erreur est la règle d'apprentissage utilisée dans ce travail, nous avons alors exposé son algorithme en détail. Nous avons ensuite passé en revue quelques travaux scientifiques dans le génie civil, élaborés en utilisant les réseaux neuromimétiques.

Le chapitre trois commence par rappeler brièvement la théorie de flexion des poutres en présentant les hypothèses, les relations traduisant l'équilibre, les équations de compatibilité et les relations constitutives. Ce dernier groupe d'équations est directement lié à la géométrie de la section à étudier et à la loi de comportement considérée. Nous faisons alors l'hypothèse d'un matériau à comportement élasto-plastique parfait, assimilable à de l'acier. Ensuite, nous présentons le développement des lois constitutives théoriques des poutres à sections triangulaires et en T dans le cas de la flexion simple.

Pour montrer le travail réalisé, nous avons alors décidé de commencer le quatrième chapitre par une présentation, tirée de la bibliographie, de la méthode des éléments finis appliquée aux poutres de Timoshenko en flexion simple. Nous présentons ensuite comment utiliser les relations (M-χ) théoriques dans un code de calcul numérique. On peut soit, en faire l'évaluation et la résolution exacte soit, utiliser les réseaux de neurones. Nous exposons alors la solution que nous proposons pour utiliser les réseaux de neurones artificiels comme outils d'interpolation de la loi de comportement. La dernière partie de ce chapitre explique la mise en œuvre pratique de la solution proposée, de la recherche des réseaux de neurones adéquats, à la programmation de la stratégie adoptée et son insertion dans un code d'éléments finis.

Dans le cinquième chapitre sont proposées différentes applications effectuées à l'aide du programme élaboré. Pour le valider et vérifier l'opportunité de la stratégie retenue, des exemples simples sont étudiés et les résultats comparés à d'autres trouvés dans la littérature ou obtenus par des approches différentes.

Nous terminons ce mémoire par quelques conclusions sur l'intérêt du travail et les résultats obtenus. Nous dressons ensuite les perspectives offertes à courts et moyens termes par les approches introduites.

Le code source et certains points techniques sont reportés en annexe, les renvois sont indiqués dans le cours du texte à chaque fois que jugée utile.

Chapitre 2

Les réseaux de neurones

2.1 Introduction

Le fonctionnement des réseaux de neurones appelés aussi réseaux neuromimétiques est basé sur celui des réseaux de neurones biologiques. Il est donc important de bien comprendre le fonctionnement du système nerveux humain pour pouvoir le reproduire.

La complexité du système nerveux naturel fait qu'il ne peut être parfaitement représenté mathématiquement. Plusieurs modèles sont alors nécessaires pour reproduire ses différents aspects, et chaque modèle doit être adapté au type de problème posé. Le choix du type de réseaux neuromimétiques en fonction du problème à étudier, est alors très important. Deux grandes classes pour lesquelles on utilise les réseaux de neurones sont : les problèmes de classification et de reconnaissance de formes, et les problèmes d'interpolation. Leur domaine d'application est très vaste. On peut citer : l'industrie, les finances, les télécommunications, l'informatique, la médecine, …, et le génie civil.

Dans ce qui suit, nous allons commencer par exposer d'une part, les réseaux de neurones naturels, voir leurs compositions, ainsi que leurs fonctionnements et nous introduirons d'autre part, les réseaux de neurones artificiels, leur modèle mathématique, les différentes fonctions d'activation usuelles, ainsi que les différentes règles d'apprentissage. La règle delta généralisée ou règle de la rétropropagation du gradient de l'erreur est utilisée dans cette étude. Son algorithme va alors être exposé en détail.

Pour finir nous exposerons quelques travaux élaborés en utilisant les réseaux neuromimétiques dans le génie civil depuis les premiers succès.

2.2 Les réseaux de neurones naturels

Le cerveau est composé d'un grand nombre de cellules appelées les neurones de l'ordre de 10^{11}. Ces neurones sont interconnectés par des jonctions appelées les synapses de l'ordre de

10^4 par neurones. Les dendrites, le soma, l'axone et l'arborisation terminale constituent le neurone (figure 2.1). [1, 11]

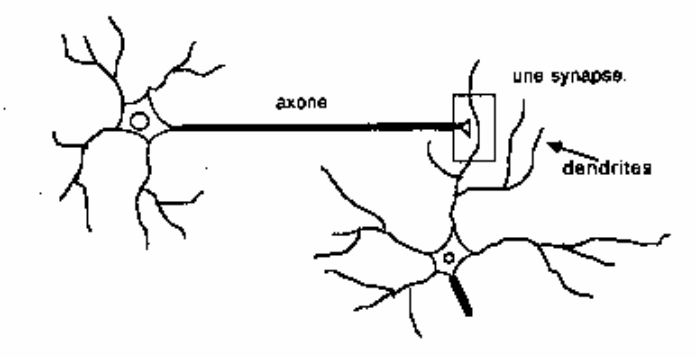


Figure 2. 1 : Anatomie d'un neurone typique.

Les dendrites, qui constituent l'organe d'entrée du neurone, recueillent l'information sous forme d'influx nerveux, arrivant des autres neurones. Elle est ensuite acheminée vers le soma. Ce dernier rassemble toutes les informations reçues par les dendrites et procède à une sommation dite *spatio-temporelle* :

Spatio parce que l'arbre dendritique fait converger sur le soma des signaux venant d'un grand domaine spatial autour du neurone.

Temporelle parce que la transmission des signaux le long d'une dendrite est caractérisée par un retard, une atténuation et un effet de filtrage.

Lorsque la sommation dépasse une valeur seuil, il y a émission d'un potentiel d'action, ***le spike*** (figure 2.2). Le neurone s'active et transmet le message le long de l'axone vers l'arborisation terminale, dans laquelle des peptides sont synthétisées et stockées à l'intérieur des vésicules qui, libérés dans l'espace synaptique, servent de médiateurs chimiques à l'influx nerveux pré-synastique. Le neurotransmetteur en se liant à son récepteur spécifique situé sur la dendrite voisine engendre, soit une dépolarisation de sa membrane dans le cas où le potentiel post-synaptique est excitateur PPSE, soit une surpolarisation dans le cas où le potentiel post-synaptique est inhibiteur PPSI. Le résultat final vient de l'addition algébrique des états d'excitation et d'inhibition des synapses intéressées. [11]

En résumé, l'anatomie du cerveau sous-entend un câblage permettant la transmission de l'information, qui est modulée et multipliée par la présence de sélectionneurs neurochimiques permettant une organisation en réseau complexe et autorisant une plasticité fonctionnelle. Cette modulation des influx prend place au niveau des synapses qui, dépendant de l'état d'excitation des neurones pré et post-synaptiques, modifient leur perméabilité face à un neurotransmetteur de façon à amplifier ou atténuer l'influx qu'ils transmettent.

La richesse du système nerveux réside dans sa capacité d'adaptation et dans le grand nombre de neurones qui le compose.

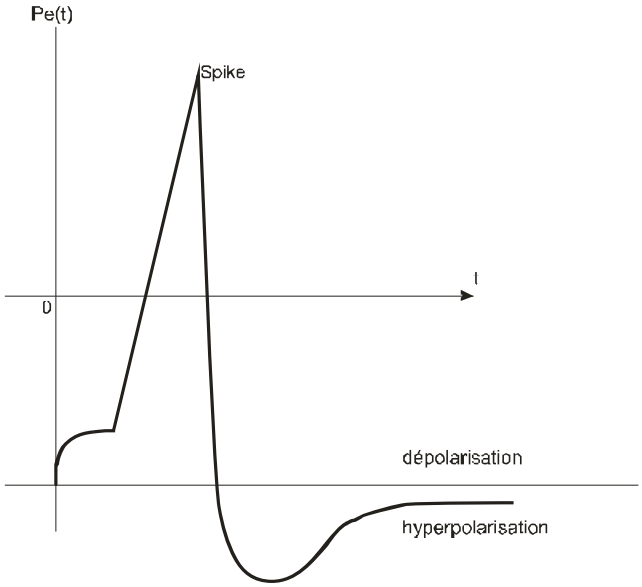


Figure 2. 2 : Potentiel d'action

2.3 Les réseaux de neurones artificiels.

2.3.1 Définition

Un réseau de neurones artificiel (RNA) ou réseau neuromimétiques est un système de traitement de l'information qui comporte certaines caractéristiques issues de l'étude des réseaux de neurones biologiques.

Ils ont été développés en tant que généralisation de modèles mathématiques de la pensée humaine, sur la base des hypothèses suivantes :

- Le traitement de l'information a lieu au sein d'éléments simples appelés neurones.
- Les signaux transitent d'un neurone à un autre par le biais d'un lien de communication.
- Chacun de ces liens possède un coefficient synaptique qui pondère le signal transmis.
- Chaque neurone applique une fonction d'activation au signal reçu pour déterminer le signal de sortie.

Les neurones, appelés aussi cellules ou nœuds, d'un même réseau sont reliés entre eux par des liens de communication. Ces derniers sont affectés de coefficients synaptiques ou poids, qui modélisent les connexions synaptiques des systèmes biologiques. Lors de l'apprentissage du réseau, ces coefficients, peuvent subir des changements pour obtenir le message désiré. Ces derniers agissent en augmentant le signal reçu (entrée excitatrice) ou bien en le diminuant (entrée inhibitrice). Ces coefficients sont des réels qui représentent la part d'information utilisée par chaque cellule pour mener correctement l'interprétation du message.

Un réseau est caractérisé par son architecture et par la méthode de mise à jour des coefficients synaptiques. Chaque neurone possède un état interne appelé activation ou niveau d'activité, qui est représenté par une fonction de l'information reçue.

2.3.2 Modèle mathématique

Soit le réseau de neurones de la figure 2. 3. Il comporte trois cellules d'entrées X_1, X_2, X_3 et une sortie Y. Le neurone Y reçoit des stimuli des neurones X_1, X_2, X_3.

Notons :

x_i : activation du neurone X_i

w_i : poids ou coefficient synaptique associé à la connexion reliant les cellules X_i et Y

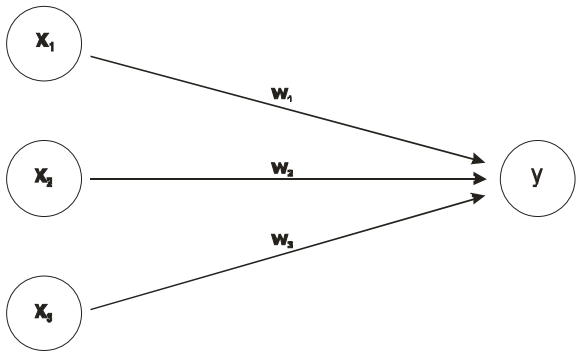


Figure 2. 3 : Exemple de réseau de neurones de base

Le signal d'entrée P du neurone Y est la somme pondérée des signaux transmis par X_1, X_2, X_3 et est donné par :

$$P = w_1x_1 + w_2x_2 + w_3x_3 = \sum w_ix_i \tag{2.1}$$

L'activation y du neurone Y est donnée par une fonction de son entrée :

$$y = f(P) \tag{2.2}$$

Où f peut être une fonction linéaire, sigmoïde ou autre.

Le neurone de la figure 2. 3 peut alors être représenté comme montré par la figure 2.4 :

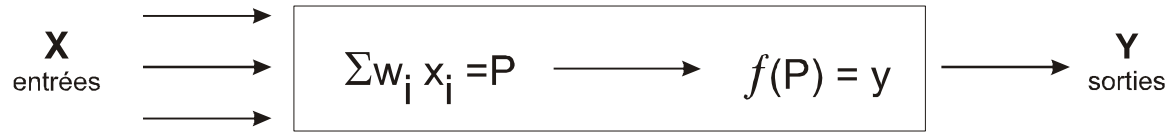


Figure 2. 4 : Schématisation du réseau de neurones

On suppose maintenant, que le neurone Y soit connecté à d'autres neurones Z_1 et Z_2. Les coefficients synaptiques des connexions sont notés respectivement v_1 et v_2. Un tel élément de traitement est appelé cellule cachée. Celle-ci envoie son signal y à chacun des neurones Z_i.

Les signaux reçus par Z_1 et Z_2 seront souvent différents car l'information est multipliée par le poids approprié v_1 ou v_2.

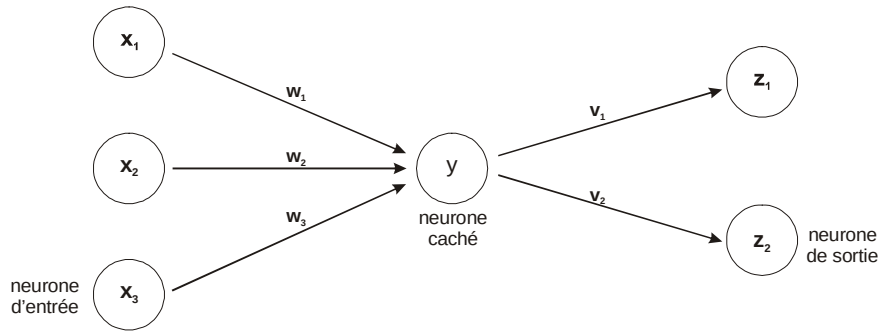


Figure 2. 5 : Réseau de neurones comportant une unité cachée

2.3.3 Les fonctions d'activation usuelles

Le signal de sortie d'un neurone artificiel est obtenu par application de sa fonction d'activation à la somme pondérée des signaux d'entrée. Parmi les fonctions d'activation les plus utilisées, on peut citer : la fonction identité, la fonction de Heaviside, et la fonction sigmoïde. Dans de nombreux réseaux, l'activation des cellules d'entrée est semblable au signal provenant du milieu extérieur. Dans ce cas la fonction est la fonction identité (Purelin) donnée par :

$$f(x) = x, \forall x \tag{2.3}$$

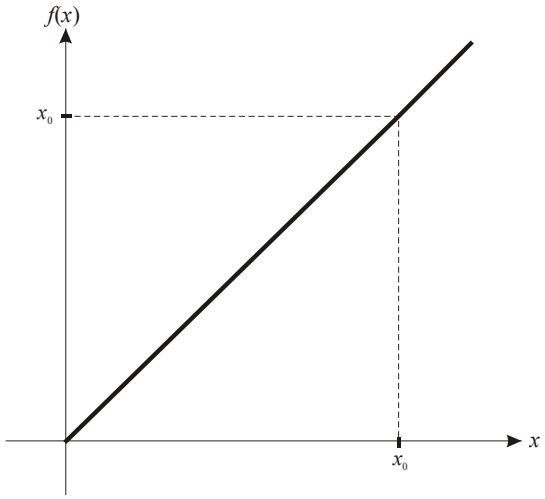


Figure 2. 6 : La fonction identité (Purelin)

Les réseaux à une couche utilisent souvent une fonction en escalier pour convertir le signal d'entrée en une sortie binaire ou bipolaire.

La fonction binaire en escalier, avec une valeur seuil θ en dessous de laquelle le signal ne se propage pas, est l'une des plus courantes, et est appelée fonction de Heaviside avec seuil θ.

$$\begin{aligned} f(x) &= 1 \qquad \text{si } x \rangle \theta \\ f(x) &= 0 \qquad \text{si } x \langle \theta \end{aligned} \tag{2.4}$$

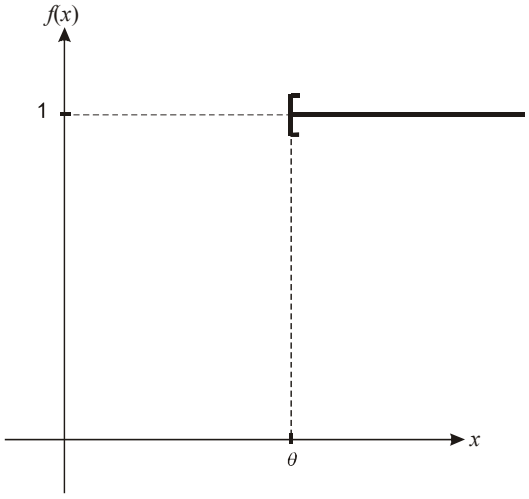


Figure 2. 7 : La fonction de HEAVISIDE

Parmi les fonctions sigmoïdes, on peut citer la fonction tangente hyperbolique et la fonction logistique (Logsig). Cette dernière est une sigmoïde prenant ses valeurs entre 0 et 1, elle est souvent utilisée lorsque la valeur de sortie souhaitée du signal est soit binaire, soit comprise entre 0 et 1.

$$f(x) = \frac{1}{1+e^{-\sigma x}} \tag{2.5}$$

$$\frac{df(x)}{dx} = \sigma\, f(x) \times \left[1 - f(x)\right] \tag{2.6}$$

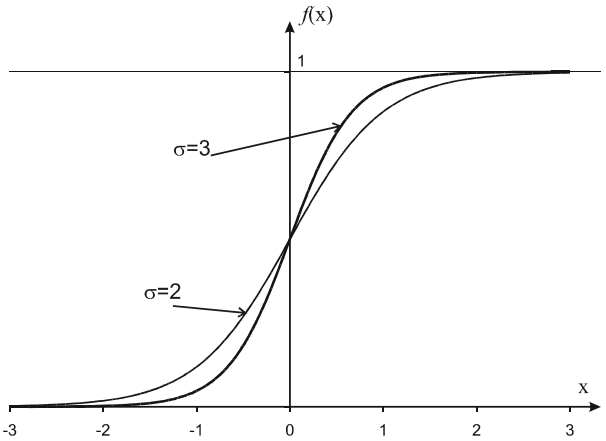


Figure 2. 8 : Fonctions sigmoïdes logistiques pour 2 valeurs du paramètre de raideur σ

La fonction sigmoïde binaire peut être modifiée de façon à couvrir n'importe quel intervalle [*a*, *b*], *a* et *b* étant des réels quelconques. On définit alors les paramètres :

$$\begin{cases} \gamma = b - a \\ \eta = a \end{cases} \tag{2.7}$$

La fonction sigmoïde définie par l'équation ci-dessous est la fonction recherchée :

$$g(x) = \gamma\, f(x) - \eta \tag{2.8}$$

Sa dérivée par rapport à x est :

$$\frac{dg(x)}{dx} = \frac{1}{\gamma}\left[\eta + g(x)\right]\left[\gamma - \eta - g(x)\right] \tag{2.9}$$

L'intervalle le plus courant étant [-1, 1], la fonction sigmoïde bipolaire (γ=2, η=1), représentée par la figure 2.8, est définie par la relation suivante :

$$g(x) = 2f(x) - 1 = \frac{2}{1+e^{-\sigma x}} - 1 = \frac{1-e^{-\sigma x}}{1+e^{-\sigma x}} \tag{2.10}$$

$$\frac{dg(x)}{dx} = \frac{\sigma}{2}\left[1 + g(x)\right]\left[1 - g(x)\right] \tag{2 11}$$

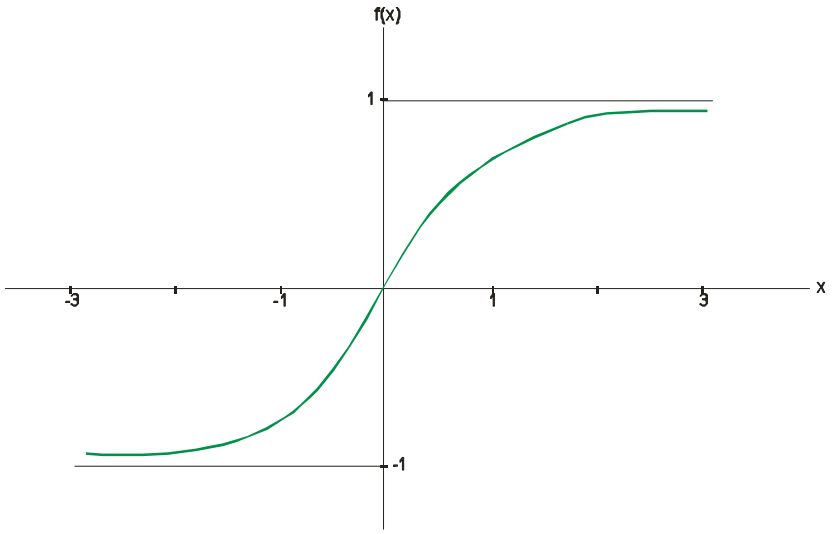


Figure 2. 9 : Fonction sigmoïde bipolaire pour σ=1

Lorsque les sorties souhaitées sont comprises entre –1 et 1, la fonction utilisée est la fonction tangente hyperbolique donnée par la relation ci-dessous :

$$h(x) = \frac{e^{\sigma_x} - e^{-\sigma_x}}{e^{\sigma x} + e^{-\sigma x}} = \frac{1 - e^{-2\sigma x}}{1 + e^{-2\sigma x}} \tag{2.12}$$

Sa dérivée par rapport à *x* est :

$$\frac{dh(x)}{dx} = \sigma[1 + h(x)][1 - h(x)] \tag{2.13}$$

La relation entre la fonction tangente hyperbolique et la fonction sigmoïde bipolaire est :

$$h(x) = g(2x) \tag{2.14}$$

2.3.4 Les méthodes d'ajustement des coefficients synaptiques [12].

La méthode d'ajustement des coefficients synaptiques est une importante caractéristique des différents réseaux de neurones. On distingue deux types d'apprentissage :

- Apprentissage libre ou non supervisé.
- Apprentissage sous surveillance ou supervisé.

2.3.4.1 Apprentissage non supervisé.

L'apprentissage non supervisé ne nécessite qu'un vecteur de données d'entrée (pattern), le réseau modifie ses paramètres en tenant compte des informations locales. Ces méthodes n'ont pas besoin de sorties désirées préétablies.

2.3.4.2 Apprentissage supervisé.

L'apprentissage supervisé est réalisé par présentation d'une séquence de vecteurs d'entrée (pattern), auxquels sont associés des vecteurs de sortie (cible) correspondants. Les coefficients synaptiques sont alors ajustés en référence à un algorithme d'apprentissage.

2.3.5 Les règles d'apprentissage.

La méthode d'ajustement des coefficients synaptiques pendant l'apprentissage du réseau peut être choisie parmi les règles suivantes :

2.3.5.1 La règle de HEBB.

En 1949, HEBB a proposé le premier mécanisme d'évaluation des synapses. Une interprétation de cette règle pour les réseaux de neurones est la suivante : si deux neurones connectés entre eux sont activés en même temps, la connexion qui les relie doit être renforcée, dans le cas contraire elle n'est pas modifiée. Sa formulation est la suivante :

$$w_{ij}(t+1) = w_{ij}(t) + \eta S_i S_j \tag{2.15}$$

Avec :

$w_{ij}(t)$ le poids de la connexion reliant les neurones S_i et S_j.

η Un nombre compris entre 0 et 1, représentant le taux d'apprentissage.

T représente l'étape d'apprentissage.

2.3.5.2 La règle du perceptron.

Dans le cas du perceptron, la fonction d'activation est une fonction discrète. Les sorties prennent des valeurs binaires (0 ou 1). La règle d'apprentissage du perceptron est la suivante :

$$\begin{cases} w_{ij}(t+1) = w_{ij}(t) + \eta\, x_i & \text{Si la sortie actuelle est égale à 0 et doit être égale à 1} \\ w_{ij}(t+1) = w_{ij}(t) - \eta\, x_i & \text{Si la sortie actuelle est égale à 1 et doit être égale à 0} \\ w_{ij}(t+1) = w_{ij}(t) & \text{Si la sortie est correcte.} \end{cases} \quad (2.16)$$

Où,

η représente le coefficient d'apprentissage,

t l'étape d'apprentissage,

x_i l'entrée du neurone i,

w_{ij} la connection synaptique ou poids entre le neurone i et le neurone j.

2.3.5.3 La règle de WIDROW-HOFF ou la règle Delta.

WIDROW et HOFF ont étudié l'algorithme d'apprentissage du perceptron en considérant une fonction d'activation continue et dérivable. Cette règle est connue sous le nom de la méthode des moindres carrés ou encore la règle Delta. Le principe de cette règle est donné dans ce qui suit :

- Calculer l'erreur quadratique :

$$E = \sum_{j=1}^{n} \left(d_j - y_j\right)^2 \qquad \text{avec} \quad y_j = \sum_{i=1}^{m} x_i w_{ji} \quad (2.17)$$

- Minimiser cette erreur en modifiant les poids de chaque neurone :

$$w_{ji}(t+1) = w_{ji}(t) + \eta\, x_i \left(d_j - y_j\right) \quad (2.18)$$

où

n le nombre de neurones à la sortie,

d_j la sortie désirée,

y_j la sortie calculée,

x_i l'entrée i du neurone j,

m le nombre de neurones à l'entrée,

η le coefficient d'apprentissage.

2.3.5.4 La règle Delta généralisée ou règle de la rétropropagation.

La règle Delta généralisée appelée règle de la rétropropagation du gradient est une généralisation de la règle de WIDROW-HOFF. Elle s'applique aux réseaux multicouches utilisant un apprentissage supervisé. L'idée de base est simple, le réseau apprend en essayant de diminuer son erreur à chaque itération. Il le fait en changeant l'intensité des connexions en sens inverse du signal d'erreur. Son nom provient du fait que les corrections sont apportées de la couche de sortie vers la couche d'entrée. La phase d'apprentissage est réalisée en trois étapes successives :

- Propagation des vecteurs constituant le pattern d'entrée,
- Calcul des erreurs théorie/réalité, puis rétropropagation de celles ci (dans le sens de la sortie vers l'entrée),
- Mise à jour des coefficients synaptiques, couche après couche, dans cette même direction.

2.4 La rétropropagation du gradient de l'erreur [4, 14].

Un réseau à rétropropagation de l'erreur se compose d'une première couche de cellules d'entrée, d'une couche (ou plusieurs couches) de cellules intermédiaires reliées à la couche précédente par des connexions modifiables par apprentissage, et d'une couche de sortie.

Dans ce qui suit, nous allons exposer l'algorithme de mise à jour des coefficients synaptiques. La notation illustrée à la figure 2.10 sera utilisée.

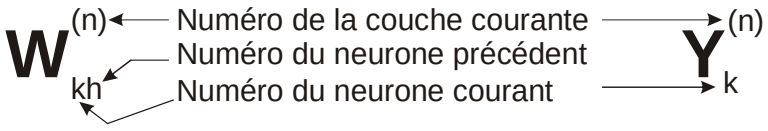


Figure 2.10 : Notation utilisée.

2.4.1 Propagation des données d'entrée.

Chaque unité d'entrée (X_i, i=1…, n) reçoit un stimulus du milieu extérieur et le transmet aux cellules de la deuxième couche (k=1).

$$y_1^{(1)} = F(x_1) = \frac{1}{1+e^{-x_1}}. \qquad (2.19)$$

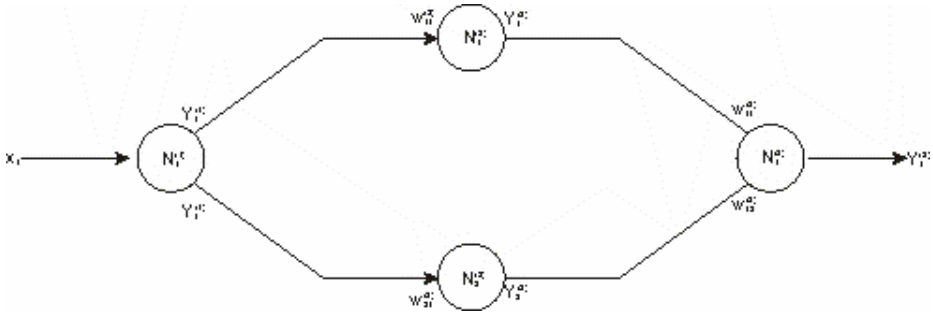


Figure 2. 11 : Schéma de la propagation des données d'entrée.

Au niveau des neurones de la deuxième couche, les influx sont pondérés et de nouveau insérés dans la fonction sigmoïde.

$$y_1^{(2)} = F\left(y_1^{(1)}\right) = \frac{1}{1+e^{-w_{11}^{(2)} y_1^{(1)}}}. \tag{2.20}$$

$$y_2^{(2)} = F\left(y_1^{(1)}\right) = \frac{1}{1+e^{-w_{21}^{(2)} y_1^{(1)}}}. \tag{2.21}$$

Les influx se concentrent au niveau de la couche 3, et on obtient :

$$y_1^{(3)} = F\left(y_1^{(2)}, y_2^{(2)}\right) = \frac{1}{1+e^{-w_{11}^{(3)} y_1^{(2)} - w_{12}^{(3)} y_2^{(2)}}} .. \tag{2.22}$$

En supposant que la connectivité du réseau est complète, la sortie du neurone k de la couche c s'écrit :

$$y_k^{(c)} = F\left(y_1^{(c-1)}, ..., y_n^{(c-1)}\right) = \frac{1}{1+e^{\left(-p_k^{(c)}\right)}} \tag{2.23}$$

$$\left\{p_k^{(c)}\right\} = \left[w_{kh}^{(c)}\right]\left\{y_h^{(c-1)}\right\} \tag{2.24}$$

$$\left\{y_k^{(c)}\right\} = \left\{F\left(p_k^{(c)}\right)\right\} \tag{2.25}$$

Lors de la phase d'apprentissage, les sorties du réseau sont comparées avec les valeurs attendues. L'écart entre ces valeurs est ensuite utilisé pour modifier les poids sur l'ensemble du réseau.

2.4.2 Rétropropagation du gradient de l'erreur.

L'écart entre les valeurs calculées et les valeurs attendues n'est pas nul. La rétropropagation de l'erreur, pour calibrer les poids du réseau, peut entraîner des problèmes de convergence et de stabilité. Pour remédier à ce problème, le gradient instantané de l'erreur est

substitué à l'erreur proprement dite. Lorsque le point d'erreur minimum est atteint, le gradient de celle ci sera nul et le réseau aura convergé malgré une erreur absolue non nulle (figure 2.12).

La rétropropagation du gradient de l'erreur se fait en suivant le même réseau que celui utilisé pour la rétropropagation des données d'entrée. Ce réseau est parcouru en sens inverse. L'erreur quadratique entre la sortie calculée ($y_1^{(3)}$) et la sortie attendue (d_1) se calcule de la façon suivante :

$$E = \frac{1}{2}\left(y_1^{(3)} - d_1\right)^2 \tag{2.26}$$

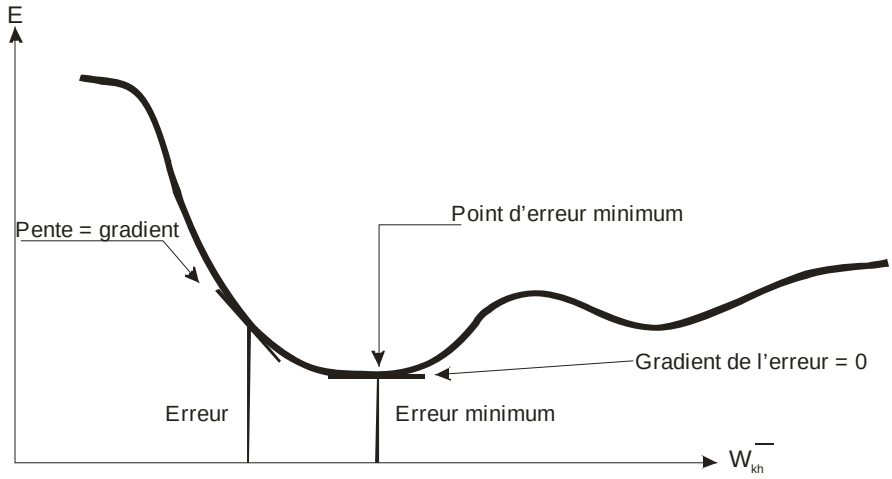


Figure 2.12 : Erreur vs. Gradient de l'erreur.

Le gradient de cette erreur par rapport au poids $w_{kh}^{(2)}$ est :

$$\frac{\partial E}{\partial w_{kh}^{(2)}} = \sum \frac{\partial E}{\partial p_i^{(3)}} \frac{\partial p_i^{(3)}}{\partial w_{kh}^{(2)}}. \tag{2.27}$$

Le premier terme de la sommation se décompose selon le même principe,

$$\frac{\partial E}{\partial p_i^{(3)}} = \frac{\partial E}{\partial y_i^{(3)}} \frac{\partial y_i^{(3)}}{\partial p_i^{(3)}}. \tag{2.28}$$

En tenant compte de l'expression de *E*, on tire des relations (2.23) et (2.28),

$$\frac{\partial E}{\partial p_i^{(3)}} = \left(y_i^{(3)} - d_i\right) F\left(p_i^{(3)}\right) = \left(y_i^{(3)} - d_i\right) F\left(\sum w_{ij}^{(3)} y_j^{(2)}\right) \tag{2.29}$$

Le second terme de l'équation (2.27) s'écrit, en fonction de la relation (2.24),

$$\frac{\partial p_i^{(3)}}{\partial w_{kh}^{(2)}} = \frac{\partial}{\partial w_{kh}^{(2)}}\left[\sum w_{ij}^{(3)} y_j^{(2)}\right]. \tag{2.30}$$

Il est évident que les éléments de $w_{ij}^{(3)}$ sont indépendants de $w_{kh}^{(2)}$. De plus, le seul élément du vecteur $y_j^{(2)}$ étant lié à $w_{kh}^{(2)}$ est celui obtenu pour j=h. Ainsi,

$$\frac{\partial p_i^{(3)}}{\partial w_{kh}^{(2)}} = w_{ik}^{(3)} F\left(p_k^{(2)}\right) y_h^{(1)} \tag{2.31}$$

En combinant les relations (2.28) et (2.31), on obtient la forme explicite de l'équation (2.27).

$$\frac{\partial E}{\partial w_{kh}^{(2)}} = \sum \left(y_i^{(3)} - d_i\right) F\left(p_i^{(3)}\right) w_{ik}^{(3)} F\left(p_i^{(2)}\right) y_h^{(1)}. \tag{2.32}$$

Définissons un facteur $\delta_k^{(c)}$ comme une mesure de contribution d'un neurone à l'erreur totale.

$$\delta_k^{(c)} = \frac{\partial E}{\partial p_k^{(c)}} \tag{2.33}$$

On a alors, pour la couche de la sortie, le facteur de contribution pour le neurone i.

$$\delta_i^{(3)} = \frac{\partial E}{\partial p_i^{(3)}} = \left(y_i^{(3)} - d_i\right) F\left(p_i^{(3)}\right) \tag{2.34}$$

Le gradient de l'erreur par rapport au poids associé au lien entre le neurone h de la couche 1 et le neurone k de la couche 2 est donné par :

$$\frac{\partial E}{\partial w_{kh}^{(2)}} = \left[\sum \delta_i^{(3)} w_{ik}^{(3)}\right] F\left(p_k^{(2)}\right) y_h^{(1)} \tag{2.35}$$

On peut déterminer le facteur de contribution $\delta_i^{(c)}$ pour une couche intermédiaire. Cette entreprise est plus complexe que pour la couche de sortie puisque la fonction d'erreur n'est pas connue au niveau des couches intermédiaires. Les couches intermédiaires ayant un effet sur l'erreur, cette fonction d'erreur dépend des potentiels d'action de chacune des couches. Ainsi, $E=E(p^{(1)},p^{(2)},p^{(3)})$. La règle de dérivation en chaîne permet d'écrire,

$$\frac{\partial E}{\partial y_k^{(c)}} = \sum \frac{\partial E}{\partial p_i^{(c+1)}} \frac{\partial p_i^{(c+1)}}{\partial y_k^{(c)}} = \sum \frac{\partial E}{\partial p_i^{(c+1)}} \frac{\partial}{\partial y_k^{(c)}} \sum w_{ih}^{(c+1)} y_h^{(c)} \tag{2.36}$$

Le seul terme de la dernière sommation qui est fonction de $y_k^{(c)}$ est pour h=k, on a alors,

$$\frac{\partial E}{\partial y_k^{(c)}} = \sum \frac{\partial E}{\partial p_i^{(c+1)}} w_{ik}^{(c+1)} = \sum \delta_i^{(c+1)} w_{ik}^{(c+1)} \tag{2.37}$$

En utilisant la relation (2.28) et en reportant dans l'équation (2.33), on a finalement :

$$\delta_k^{(c)} = F\left(p_k^{(c)}\right) \sum \delta_i^{(c+1)} w_{ik}^{(c+1)} \tag{2.38}$$

En reportant cette dernière relation dans l'équation (2.35), on obtient l'expression suivante du gradient de l'erreur en fonction du poids $w_{kh}^{(2)}$,

$$\frac{\partial E}{\partial w_{kh}^{(2)}} = \delta_k^{(2)} y_h^{(1)} \tag{2.39}$$

Avec

$$\delta_k^{(2)} = \left[\sum \delta_i^{(3)} w_{ik}^{(3)}\right] F\left(p_k^{(2)}\right) \tag{2.40}$$

La variation d'un poids, proportionnelle au gradient de l'erreur par rapport à ce poids, s'écrit donc,

$$\Delta w_{kh}^{(c)} = \alpha \frac{\partial E}{\partial w_{kh}^{(c)}} = \alpha \delta_k^{(c)} y_h^{(c-1)} \tag{2.41}$$

Les termes $\delta_k^{(c)}$ de cette relation sont déterminés par l'équation (2.34) pour la dernière couche et par l'équation (2.40) pour les couches intermédiaires. Le terme α est le facteur d'apprentissage qui gouverne la vitesse de convergence.

Cette règle d'adaptation est rarement utilisée telle quelle car elle repose sur la minimisation d'un critère d'erreur instantanée, sur un exemple d'apprentissage en particulier. La convergence est dans ce cas, très lente et nécessite un pas d'adaptation α petit pour éviter l'instabilité. L'amélioration la plus couramment utilisée est d'adjoindre à la règle d'adaptation des poids, donnée par l'équation (2.41), un terme de filtrage sur les incréments d'adaptation. Ce terme, souvent appelé momentum, est une fraction de la validation des poids pour l'exemple d'apprentissage précédent. On obtient ainsi la règle suivante :

$$\Delta w_{kh}^{(c)}(n) = \alpha \delta_k^{(c)}(n) y_h^{(c-1)}(n) + \gamma \Delta w_{kh}^{(c)}(n-1) \tag{2.42}$$

Où le terme γ est le facteur de momentum ou facteur d'oubli. Les facteurs d'apprentissage α et de momentum γ sont positifs et inférieurs à 1. La détermination de ces facteurs est intuitive.

Il existe deux façons de mettre en œuvre cette règle d'adaptation. Les poids peuvent être adaptés après chacun des exemples d'apprentissage ou pour une itération complète sur l'ensemble de la base d'apprentissage. Dans le premier cas, la règle donnée par l'équation (2.42) est appliquée. Cette règle s'appelle la méthode Delta généralisée. Dans le second cas, la

règle Delta cumulative, les produits $\delta_k^{(c)} y_h^{(c-1)}$ sont conservés pour chaque exemple d'apprentissage et les poids sont mis à jour à la fin de chaque itération sur la base d'apprentissage en fonction de la moyenne des produits $\delta_k^{(c)} y_h^{(c-1)}$. Le terme de momentum, dans ce cas, se réfère à la mise à jour précédente des poids.

$$\Delta w_{kh}^{(c)} = \alpha \frac{1}{n} \sum_{i=1}^{n} \delta_k^{(c)}(i) y_h^{(c-1)}(i) \tag{2.43}$$

2.5 Comprendre vs. apprendre par cœur. [14]

Un réseau doit établir des relations entre des données d'entrée et des données de sortie. Ces relations, après apprentissage, doivent représenter très précisément les exemples présents dans la base de données d'apprentissage. Elles doivent aussi être capables de représenter des problèmes absents de l'apprentissage ; le réseau doit être capable de généraliser. Lorsque le réseau représente bien les exemples de base de données d'apprentissage mais qu'il est incapable de représenter tout autre problème, on dit que le réseau a appris par cœur. De façon à savoir à quel moment le réseau a le mieux compris le problème sans apprendre par cœur, on ajoute à l'apprentissage des exemples de validation. Ces derniers sont indépendants des exemples d'apprentissage et ne sont pas utilisés pour la mise à jour des coefficients synaptiques. L'erreur calculée sur ces exemples montre les capacités de généralisation du réseau.

Ainsi, au lieu de conserver les poids donnant une erreur d'apprentissage minimum, on conservera les poids donnant une erreur de validation minimum. La figure 2.13 montre une courbe typique d'erreur moyenne d'apprentissage et de validation en fonction du nombre d'itérations sur la base de données d'apprentissage.

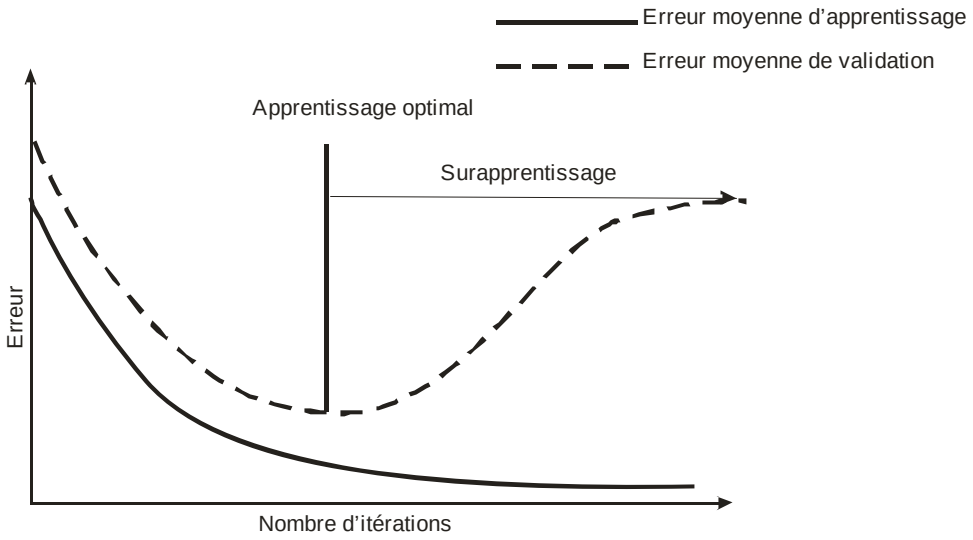


Figure 2.13 : Courbes d'erreurs moyennes d'apprentissage et de validation en fonction du nombre d'itérations sur la base de données d'apprentissage.

L'erreur moyenne d'apprentissage correspond ici à la moyenne des erreurs sur les sorties des exemples d'apprentissage. De la même façon, l'erreur moyenne de validation est la moyenne des erreurs sur les sorties des exemples de validation.

2.6 Utilisation des réseaux de neurones dans le génie civil

2.6.1 Introduction

Les premiers réseaux de neurones artificiels ont vu le jour en 1943. Les chercheurs Mc CULLOTH et PITTS ont élaboré un réseau capable de modéliser des processus physiologiques tels que la perception de la chaleur et du froid.

En 1949, HEBB a mis au point une loi générique d'apprentissage de ces réseaux, ce qui donna naissance aux premières simulations informatiques en 1956.

En 1958, ROSENBLATT développa de nouveaux systèmes, il mis au point un nouvel algorithme puissant, amélioré par la suite par WIDROW et qui est connu sous le nom de la règle de DELTA. Elle a pour objectif de minimiser l'erreur entre les réponses obtenues à la sortie des réseaux et celles désirées en utilisant la méthode des moindres carrées.

En 1982, HOPFIELD élabora des réseaux capables de résoudre des problèmes d'optimisation.

En 1990, les premiers articles sur des problèmes de génie civil ont été publiés, notamment les résultats des travaux de Wu et col. Par la suite, plusieurs travaux, touchant aux différentes spécialités de la discipline, ont été entrepris par plusieurs chercheurs. Dans ce qui suit, nous allons en présenter quelques uns.

2.6.2 Les travaux de Wu et col.

En 1990, WU et col, cités dans [14], ont déterminé le niveau d'endommagement des membrures d'un bâtiment à trois étages, en utilisant un réseau neuromimétiques à rétropropagation, avec une et deux couches intermédiaires. Ils ont effectué une série d'analyses dynamiques d'un bâtiment soumis à une sollicitation sismique pour différents niveaux d'endommagement de la structure. Comme données d'entrée du réseau, ils ont utilisé 200 mesures prises sur le spectre de la réponse de la structure, les sorties étant le niveau d'endommagement à chaque palier de la structure sur une échelle de 0 à 1 : 1 représente l'état non endommagé. Le réseau considéré était constitué de 200 nœuds à la couche d'entrée, 10 nœuds dans la couche intermédiaire, et 3 nœuds dans la couche de sortie (200-(10)-3). Son apprentissage comportait 42 exemples.

Ils ont aussi étudié la combinaison de données d'entrée provenant de mesures d'accélération au deuxième et troisième étage de la structure. Le réseau utilisé comportait 400 données d'entrée, 2 couches intermédiaires de 8 nœuds chacune, et d'une couche de sortie de 3 nœuds (400-(8)-(8)-3).

Ils conclurent que l'utilisation des réseaux de neurones artificiels pour l'évaluation de l'endommagement des structures est un axe de recherche prometteur.

2.6.3 Les travaux de Hajela et Berke.

En 1991, Hajela et Berke, cités dans [14], ont publié un article présentant les possibilités qu'apportent les réseaux de neurones dans des 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 membrures du treillis, les 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 ou 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 donne de meilleurs résultats que le réseau avec couches intermédiaires.

2.6.4 Les travaux de Barai et Pandey.

En 1995, Barai et Pandey, cités dans [14], ont publié les résultats de leurs recherches sur la détection de l'endommagement dans les membrures d'un pont en treillis à l'aide de réseaux de neurones. Ils simulent alors la réponse dynamique d'un pont sollicité par une charge mobile ponctuelle progressant à vitesse constante.

Les données d'entrée du réseau neuromimétiques consistaient en 69 points de la réponse 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 intermédiaires des 21 membrures du treillis.

Dans le premier cas, un seul point de mesure a été utilisé, le réseau ayant alors 69 données d'entrées pour une architecture (69-(21)-(21)-21). Pour le deuxième et troisième cas 3 et 5 points de mesures ont été utilisés. Aux 69 données d'entrées s'ajoutaient alors 3 et 5 nœuds dont la valeur associée déterminant à quel point de mesure les 69 autres données d'entrées étaient associées. L'architecture des réseaux était alors (72-(21)-(21)-21) pour trois points de mesure, et (74-(21)-(21)-21) pour cinq points de mesure.

L'apprentissage des réseaux comportait 16 exemples d'apprentissage et 5 exemples test pour chaque point de mesure. 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 mesure et d'exemples d'apprentissage ne donnent pas nécessairement de meilleurs résultats qu'un petit nombre.

2.6.5 Les travaux de Henchi, Fafard et Langis.

En 1997, Henchi, Fafard et Langis [14] ont utilisé les réseaux de neurones à l'identification structurale. Ils ont étudié deux cas théoriques qui constituent une première incursion dans l'utilisation de réseaux neuromimétiques pour l'évaluation de l'endommagement localisé dans une structure en utilisant des modèles tridimensionnels. Par la suite, ils ont présenté les résultats des analyses concernant un cas réel soit la quantification du niveau d'endommagement des différentes membrures d'un pont (le pont de Senne terre) (le pays).

Dans le premier cas, il s'agissait de déterminer l'état d'endommagement dans l'âme d'une poutre en T de béton armé en se basant sur des analyses statiques ou des analyses en fréquences.

Le réseau a été entraîné à partir de 33 exemples d'apprentissage pour des analyses statiques. Chacun des exemples est constitué d'une combinaison de modules élastiques dans la zone endommagée de l'âme de la poutre et des déflexions correspondantes. En plus des exemples d'apprentissage, 3 exemples de validation et 1 exemple test ont été ajoutés à la base de données de l'analyse.

Plusieurs architectures de réseaux ont été étudiées. Celui à une couche intermédiaire comportant 20 mesures s'est avéré le choix offrant une meilleure convergence. L'architecture du réseau était (8-(20)-7) pour 8 points de mesures.

Pour vérifier la précision des calculs, une analyse par éléments finis à été effectuée avec les modules élastiques obtenus du réseau de neurones. Les déflexions calculées par cette nouvelle analyse sont comparées aux déflexions de l'exemple test. On remarque des erreurs maximales inférieures à 1%.

Le réseau de neurones pour l'analyse en fréquence a été entraîné à partir de 73 exemples d'apprentissage. Chacun des exemples est constitué d'une combinaison de modules élastiques dans la zone endommagée de l'âme de la poutre aux quelle s'ajoutent les huit premières fréquences naturelles correspondantes à cette combinaison. Six exemples de validation et un exemple test ont été ajoutés à la base de données de l'analyse.

Le réseau donnant les meilleurs résultats était composé de 2 couches intermédiaires de 4 et 7 neurones. Son architecture était (4-(4)-(7)-7). D'autres architectures ont été testées, les réseaux à une couche intermédiaire offrent des résultats de qualité nettement inférieurs aux réseaux à deux couches intermédiaires.

Les déflexions obtenues des analyses en fréquence sont comparées aux déflexions théoriques, aussi sont comparées les fréquences obtenues à partir des modules élastiques donnés par les analyses statiques. L'erreur maximale retenue est inférieure à 0.5%.

Ils ont ensuite vérifié s'il était possible de déterminer les constantes matérielles d'une pièce pour un cas plus réaliste où seule une partie de l'âme est endommagée. Pour ceci, seules des analyses en fréquences ont été effectuées.

L'apprentissage du réseau neuromimétiques comportait 36 exemples d'apprentissage, 3 exemples de validation, et 1 exemple test.

Le réseau utilisé comportait une couche intermédiaire, son architecture était (5-(15)-9).

De façon à vérifier les performances du réseau, une analyse par éléments finis a été effectuée avec les modules élastiques obtenus précédemment. Les fréquences obtenues sont comparées aux fréquences de l'exemple test. L'erreur maximale est inférieure à 0.1%.

Ils conclurent que la précision des résultats obtenus des réseaux de neurones est excellente.

2.6.6 Les travaux de Ko, Sun & Ni.

L'objectif de ces travaux publiés en 2001 [13], est de développer un schéma à trois phases pour la détection des dommages d'un pont suspendu (Kap Shui Mun) à partir de mesures des caractéristiques modales. La stratégie de diagnostic a pour but de détecter l'occurrence, la position et l'étendu de l'endommagement. Le premier niveau du schéma est une nouvelle technique de filtrage utilisant un réseau de neurones ayant pour but de signaler le dommage (une sorte d'alarme). La deuxième étape de la stratégie a pour but de localiser la zone

endommagée. Et le troisième niveau devrait préciser la localisation en identifiant l'élément concerné. Un réseau de neurones de type perceptron multicouches est alors utilisé dans ce cas.

2.6.7 Les travaux de Beirow & Osterrieder.

Dans cette étude [3], et afin de faire face à une éventuelle demande d'extension des capacités de transmission, par l'augmentation des dimensions et du nombre des antennes, les méthodes de calcul statique et dynamique des tours pour transmission TV ont été analysées en prenant comme exemple la tour de Cottebuss (Allemagne). Le travail comporte une étude expérimentale en deux parties dont l'une est réalisée en soufflerie sur un modèle réduit, et l'autre exploite les mesures réalisées par les instruments de surveillance installés sur la tour. On mesure alors les directions du vent et les vitesses de vibration. La partie numérique de la recherche a été réalisée par les méthodes stochastiques et déterministes usuelles dans le calcul dynamique des structures soumises au vent. Une troisième méthode est également utilisée, elle est qualifiée d'hybride car elle couple les méthodes numériques et les réseaux de neurones pour essayer d'approcher la réalité d'une meilleure manière. Un réseau de neurones est alors éduqué pour calculer la réponse dynamique de la structure sous des vitesses de vent arbitraires en utilisant les mesures expérimentales.

2.6.8 Les travaux de Vassileva.

Ce travail [21] présente les résultats d'une recherche effectuée dans le domaine de la prédiction du mouvement sismique du sol en utilisant les réseaux de neurones artificiels. Dans cette étude les enregistrements de 100 tremblements de terre sont utilisés. Ils sont tous mesurés dans une zone géographique raisonnablement petite en Californie. Ils ont été analysés afin d'en extraire des caractéristiques qui seraient capables de représenter les séismes d'une façon plus synthétique que l'enregistrement complet. Un réseaux de neurones effectuant la compression des enregistrements a été alors proposé. On a aussi cherché un deuxième réseau pour relier les sismographes aux spectres de réponse.

2.6.9 Les travaux de Mayoraz, Cornu & Vulliet.

Cette recherche [16] se rapporte à la prédiction des mouvements de pentes. On y propose une approche basée sur les réseaux de neurones artificiels en remplacement des méthodes couramment utilisées dans ces cas, et qui sont basées sur des modèles mécaniques simulant l'écoulement de la masse de sol selon diverses lois de comportement. Le but de la démarche est de détecter les changements de vitesse de la masse en mouvement à l'aide de données climatologiques et physiques telles que les pluies, les débits et les pressions interstitielles. Le réseau fonctionne comme une alarme et constitue un outil d'aide à la décision. Les données sont fournies par deux sites suivis en continu : La Chenaula en suisse et Salledès, près de Clermont-Ferrand en France. Le type de réseau choisi est un perceptron à deux couches. Le modèle permet d'estimer la vitesse de glissement à trois jours, sur la base de deux réseaux en cascade. Les pressions interstitielles estimées par le premier réseau à l'aides des précipitations nettes et des précédentes valeurs des pressions sont réintroduites dans un second réseau pour estimer la vitesse de glissement, avec également les précédentes valeurs de ce paramètre.

2.6.10 Quelques travaux effectués au département de génie civil de Tlemcen.

2.6.10.1 Nouvelle approche de génération d'accélerogramme artificiel compatible avec un spectre de réponse en utilisant les réseaux de neurones

En 2002, RAHMOUN.Z [18] a utilisé les réseaux de neurones pour générer un accélérogramme artificiel compatible avec un spectre de réponse. Il fait apprendre à un réseau de neurones le chemin inverse directement à partir d'un enregistrement d'un accélérogramme et son spectre de réponse. Pour cela deux réseaux ont été développés. Le premier réseau est développé comme instrument de compression de données. Il compresse le vecteur de la transformée discrète de Fourier des accélérogrammes dans un vecteur de dimension réduite. Le second apprend la relation entre un spectre de réponse et un spectre de Fourier compressé par le premier réseau de neurones.

Le premier réseau a été entraîné à partir de 21 accélérogrammes pour la procédure d'apprentissage, et le second à partir de 12 accélérogrammes.

A la fin de cette étude, un accélérogramme artificiel a été généré à partir d'un spectre de réponse réglementaire. L'accélérogramme généré ressemble aux accélérogrammes du groupe d'apprentissage.

2.6.10.2 Estimation du risque lié a l'effet de site et génération d'un spectre de réponse a la surface libre

En 2004, DERRAS.B [6] a utilisé les réseaux de neurones pour estimer le risque lié à l'effet de site et générer un spectre de réponse à la surface libre. Pour cela il a développé deux systèmes de réseaux de neurones. Il fait apprendre au premier réseau le modèle linéaire équivalent, et au second les événements réels enregistrés en surface et en profondeur. Le premier réseau a été entraîné pour estimer le niveau de risque lié à l'effet de site à partir des facteurs d'amplification spectrale. Le second pour tracer le spectre de réponse en surface à partir d'un enregistrement sur un site de référence, en tenant compte des propriétés élastiques du profil par une classification de site basée sur la vitesse de cisaillement moyenne sur trente mètres de profondeur et la fréquence caractéristique du sol.

Quatre exemples ont été traités pour valider les deux systèmes neuronaux. Il en résulte que, le premier réseau peut estimer le niveau de risque pour des sollicitations faibles à modérées. Pour des sollicitations fortes, il signale une sous estimation du risque. Les spectres de réponse générés à la surface libre par le second réseau de neurones peuvent donner des approximations intéressantes.

2.7 Conclusion

Dans ce chapitre, nous avons commencé par présenter les réseaux de neurones biologiques, car il est important de bien comprendre le fonctionnement du système nerveux humain pour pouvoir le reproduire. Nous avons ensuite défini les réseaux de neurones artificiels ou réseaux neuromimétiques, leur modèle mathématique, les différentes fonctions d'activation usuelles, ainsi que les différentes règles d'apprentissage. La règle delta généralisée ou règle de la

rétropropagation du gradient de l'erreur est la règle d'apprentissage utilisée dans ce travail, nous avons alors exposé son algorithme en détail. Pour finir, nous avons passé en revue quelques travaux élaborés en utilisant les réseaux neuromimétiques dans le génie civil.

Nous rappelons que nous voulons utiliser les réseaux de neurones artificiels pour étudier la flexion simple des poutres en élasto-plasticité. Dans le chapitre qui suit, nous allons alors exposer la théorie de flexion des poutres, les relations traduisant l'équilibre, les équations de compatibilité et les équations constitutives. Ces dernières sont directement liées à la géométrie de la section et à la loi de comportement du matériau considéré. Nous présentons alors, les lois constitutives des poutres à sections triangulaires et en T.

Chapitre 3

Formulation des relations Moment – Courbure élasto-plastiques des poutres fléchies

3.1 Introduction.

Dans ce mémoire de magister, nous voulons utiliser les réseaux de neurones artificiels, que nous venons de présenter dans le chapitre précédent, pour étudier la flexion simple des poutres en élasto-plasticité. Dans la théorie de flexion, les équations d'équilibre sont exprimées en fonction de résultantes des contraintes, appelées efforts internes ou efforts de réduction, et qui sont l'effort normal, le moment de flexion et l'effort tranchant. Les équations constitutives doivent alors être exprimées en fonctions de ces efforts. Elles peuvent être déterminées théoriquement à partir des relations contraintes - déformations dites, lois de comportement. C'est ce travail théorique que nous allons exposer dans ce chapitre.

La théorie de flexion des poutres est brièvement rappelée en présentant les hypothèses, les relations traduisant l'équilibre et les équations de compatibilité. Quant au dernier groupe d'équations qui sont les relations constitutives, elles sont directement liées à la géométrie de la section à étudier et à la loi de comportement considérée. Nous faisons alors l'hypothèse d'un matériau à comportement élasto-plastique parfait, assimilable à de l'acier. Nous présentons, ensuite, les lois constitutives des poutres à sections triangulaires et en T. Dans le cas de la flexion simple, elles prennent la forme de relations moment – courbure [15].

3.2 Théorie de la flexion des poutres.

La mécanique des milieux continus en petites déformations prend comme hypothèses générales les points suivants :

- Le matériau considéré est continu, homogène et à température uniforme constante,

- Le chargement est statique, c'est-à-dire que les actions sont appliquées progressivement,
- Les déformations sont petites,
- La loi de comportement du matériau est isotrope et indépendante du temps.

La théorie simplifiée de la flexion des poutres droites planes, par des hypothèses supplémentaires, transforme le problème tridimensionnel de la mécanique des milieux continus en un problème unidimensionnel où, la variable indépendante « principale » est l'abscisse x mesurée sur un axe parallèle à la ligne droite définie par l'ensemble des centres de gravité de sections droites, appelée ligne moyenne de la poutre.

Dans ce qui suit, nous rappelons les trois catégories d'équations qui régissent le comportement de ces éléments de structures. Il s'agit des équations d'équilibre, des équations de compatibilité et des relations constitutives [8].

3.2.1 Équations d'équilibre

Si on considère que la poutre fléchit dans le plan (xy), le tenseur symétrique des contraintes se réduit à :

$$[\sigma]=\begin{bmatrix}\sigma_x & \tau_{xy}\\ \tau_{xy} & \sigma_y\end{bmatrix} \text{ Ou bien sous forme de vecteur, } \{\sigma\}=\begin{Bmatrix}\sigma_x\\ \sigma_y\\ \tau_{xy}\end{Bmatrix} \tag{3.1}$$

Dans cette théorie, on estime que σ_y est négligeable devant σ_x. On écrit alors que :

$$\sigma_y = 0 \tag{3.2}$$

On introduit ensuite la notion d'efforts internes, qui sont des résultantes obtenues par intégration de ces contraintes sur la section S de la poutre. Il s'agit de l'effort normal N, du moment de flexion M (autour d'un axe normal au plan xy et passant par la ligne moyenne positionnée à y_0) et de l'effort tranchant T.

$$N(x)=\iint_S \sigma_x(x,y)\cdot dS \tag{3.3}$$

$$M(x)=\iint_S (y-y_0)\cdot\sigma_x(x,y)\cdot dS \tag{3.4}$$

$$T(x)=\iint_S \tau_{xy}(x,y)\cdot dS \tag{3.5}$$

La figure (3.1) représente ces différentes grandeurs et la convention de signes adoptée.

Pour écrire les équations d'équilibre en fonction des efforts internes, on considère un tronçon de poutre de longueur dx, compris entre les abscisses x et x+dx, soumis à une force p

par unité de longueur, de composantes p_x et p_y sur les axes de coordonnées, tel que montré dans la figure (3.2).

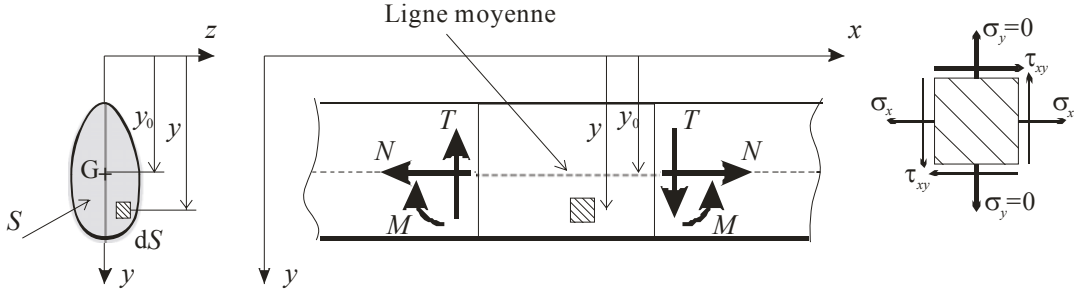


Figure 3.1 : Les contraintes, les efforts internes et leurs sens positifs en théorie de flexion des poutres.

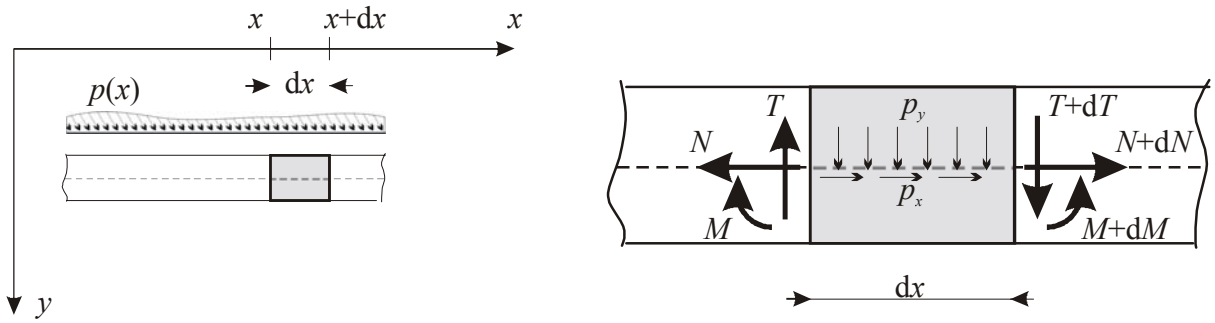


Figure 3.2 : Bilan des forces appliquées à un tronçon de poutre.

En écrivant les trois relations relatives à l'équilibre statique de ce tronçon, on obtient, après quelques simplifications :

$$\frac{\mathrm{d}T}{\mathrm{d}x}(x) = -p_y(x) \tag{3.6}$$

$$\frac{\mathrm{d}M}{\mathrm{d}x}(x) = T(x) \tag{3.7}$$

$$\frac{\mathrm{d}N}{\mathrm{d}x}(x) = -p_x(x) \tag{3.8}$$

Ces relations sont les trois équations différentielles reliant les composantes de la force répartie, l'effort tranchant, le moment fléchissant et l'effort normal.

3.2.2 Équations de compatibilité

La plus importante des hypothèses de la théorie des poutres est celle relative à la conservation de la planéité des sections droites (les sections normales à la ligne moyenne) après déformation (voir figure 3.3). Elle implique une variation linéaire du déplacement axial des points appartenant à ces plans perpendiculaires à l'axe x. Si on note $\theta(x)$ la rotation d'une section droite (ou de sa normale) par rapport à sa position initiale, et si on nomme $u(x)$ le déplacement axial de la ligne moyenne, positionnée à y_0, alors le déplacement suivant x d'un point (x,y) quelconque est donné par :

$$\bar{u}(x,y) = u(x) - (y - y_0)\cdot\theta(x) \tag{3.9}$$

On suppose aussi que le déplacement latéral de ces points, appelé flèche, ne dépend que de x.

$$\bar{v}(x,y) = v(x) \tag{3.10}$$

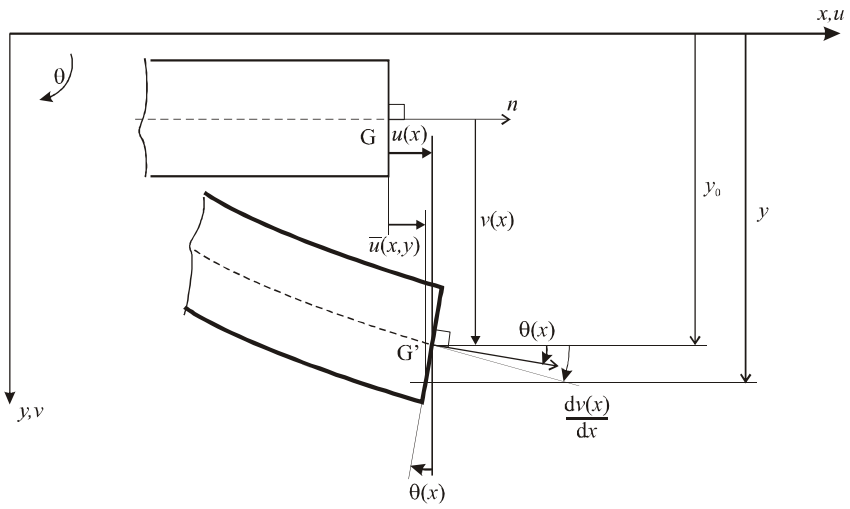


Figure 3.3 : Champs de déplacement supposé.

Il existe deux théories des poutres distinctes qui ne diffèrent que par l'hypothèse faite sur la valeur de $\theta(x)$. Il s'agit des théories de Bernoulli et de Timoshenko. Dans la première, on suppose que la rotation est donnée par la pente de la tangente à la flèche. Elle implique que les sections droites restent normales à la ligne moyenne après déformation.

$$\theta(x) = \frac{dv}{dx} \tag{3.11}$$

Dans la seconde, on suppose que la rotation $\theta(x)$ est égale à la pente de la ligne moyenne moins un angle β dû à la distorsion :

$$\theta(x) = \frac{dv}{dx} - \beta \tag{3.12}$$

La déformation longitudinale qui résulte de ce champ de déplacement est donnée par [9] :

$$\varepsilon_x = \frac{\partial \overline{u}}{\partial x} = \frac{\mathrm{d}u}{\mathrm{d}x} - \left(y - y_0\right) \cdot \frac{\mathrm{d}\theta}{\mathrm{d}x} \tag{3.13}$$

La distorsion est quant à elle donnée par [9] :

$$\gamma_{xy} = \frac{\partial \overline{u}}{\partial y} + \frac{\partial \overline{v}}{\partial x} \tag{3.14}$$

Si c'est l'hypothèse (3.11) qui est adoptée, alors on a :

$$\gamma_{xy} = -\frac{\mathrm{d}v}{\mathrm{d}x} + \frac{\mathrm{d}v}{\mathrm{d}x} = 0 \tag{3.15}$$

C'est-à-dire que dans la théorie de Bernoulli, la distorsion est négligée. Par contre, si on retient l'équation (3.12), alors la distorsion est donnée par :

$$\gamma_{xy} = -\frac{\mathrm{d}v}{\mathrm{d}x} + \beta + \frac{\mathrm{d}v}{\mathrm{d}x} = \beta \tag{3.16}$$

3.2.3 Équations constitutives

On peut rappeler [20] que dans le cas des matériaux élastiques linéaires isotropes, la relation contrainte – déformation dans le plan (*xy*) s'écrit :

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} = \frac{\mathrm{E}}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \dfrac{1-\nu}{2} \end{bmatrix} \times \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{bmatrix} \tag{3.17}$$

Où, E est le module de Young et ν est le coefficient de Poisson.

Si maintenant σ_y est supposée égale à zéro, on a :

$$\varepsilon_y = -\nu \varepsilon_x \tag{3.18}$$

A partir des équations (3.17) et (3.18), on peut écrire les relations suivantes :

$$\begin{cases} \sigma_x = \mathrm{E} \cdot \varepsilon_x \\ \tau_{xy} = \mathrm{G} \cdot \gamma_{xy} \end{cases} \tag{3.19}$$

Où, G est le module de cisaillement, et pour un matériau isotrope on a :

$$G = {E}/{2(1+\nu)} \tag{3.20}$$

Dans la théorie de flexion, les équations d'équilibre (3.6 à 3.8) sont exprimées en fonction des efforts internes (*M, N, T*). Les équations constitutives doivent alors être exprimées en fonction de ces efforts. Pour une loi de comportement élastique linéaire, il s'agit d'intégrer les relations (3.19) sur la section *S* de la poutre, conformément aux définitions de ces efforts données par les équations (3.3 à 3.5).

En se mettant dans les conditions de la flexion simple, c'est-à-dire en supposant l'effort normal et le déplacement axial de la ligne moyenne nuls (N=0 et $u(x)$ =0), la relation (3.13) devient :

$$\varepsilon_x = -\frac{d\theta}{dx}\cdot(y - y_0) \tag{3.21}$$

On introduit alors la notion de courbure :

$$\chi(x) = -\frac{d\theta}{dx} \tag{3.22}$$

Par suite, la déformation longitudinale s'écrit :

$$\varepsilon_x = \chi(x)\cdot(y - y_0) \tag{3.23}$$

On peut alors dire que cette déformation est une fonction linéaire de l'ordonnée y, dont la pente est la courbure χ. En utilisant ce résultat dans l'équation (3.19) on a :

$$\sigma_x = E\cdot\chi(x)\cdot(y - y_0) \tag{3.24}$$

C'est-à-dire qu'en élasticité linéaire la contrainte est une droite de l'ordonnée y de pente $E\cdot\chi$. Avec cette expression, le moment fléchissant donné par la relation (3.4) devient [15] :

$$M(x) = E\cdot\iint_S (y - y_0)^2 dS\cdot\chi(x) = E\cdot I\cdot\chi(x) \tag{3.25}$$

Où, I est le moment d'inertie de la section autour d'un axe normal au plan (xy) et passant par la ligne moyenne. Il convient de remarquer que pour $y = y_0$ la déformation et la contrainte sont nulles : c'est l'axe neutre.

Cette équation constitutive traduit alors la proportionnalité entre le moment de flexion et la courbure dans le cas d'une loi de comportement élastique linéaire.

La relation constitutive est donc directement liée à la géométrie de la section à étudier et à la courbe de comportement considérée. Dans ce qui suit, on présente l'élaboration de cette relation en flexion simple, dans le cas d'un comportement élasto - plastique parfait, pour deux types de sections : les sections triangulaires et les sections en T.

3.3 Hypothèses et définitions

3.3.1 Loi de comportement

Le matériau considéré dans cette étude est un matériau à un comportement élasto-plastique parfait, assimilable à de l'acier. Il est parfaitement défini par son module de Young (E) et sa limite élastique (σ_0).

Cependant, tout matériau solide peut être amené à la rupture. Cette remarque conduit à limiter la courbe élasto-plastique parfaite à l'intervalle de déformation $\left]-\varepsilon_p, \varepsilon_p\right[$; ε_p représentant la valeur critique de la déformation au-delà de laquelle il y'a rupture du matériau. Si ε_0 est la déformation correspondant à σ_0, on choisit de définir ε_p par :

$$\varepsilon_p = p\varepsilon_0 \tag{3.26}$$

La loi de comportement est alors celle représentée à la figure (3.4), son palier de ductilité ayant une longueur de $(p-1)\varepsilon_0$. La valeur de la contrainte à la rupture est évidemment égale à :

$$\sigma_p = \sigma_0 \tag{3.27}$$

Pour les applications numériques, il va falloir donner des valeurs précises à E, σ_0 et p. Dans le cas où le matériau est de l'acier, ce qui sied le mieux à la loi retenue, le module de Young vaut :

$$E = 210\,\text{GPa} \tag{3.28}$$

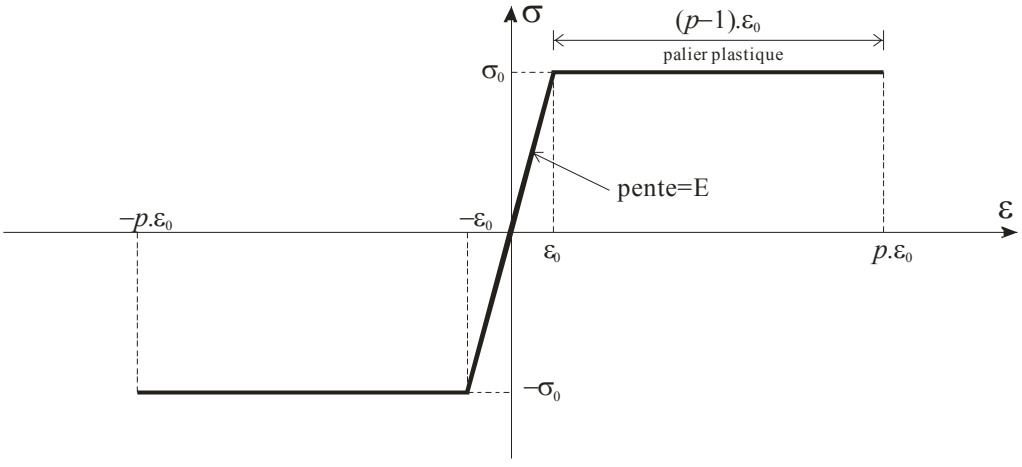


Figure 3.4 : Comportement élasto - plastique parfait

La limite élastique et la valeur de *p* dépendent, quant à elles, des nuances d'acier. Il existe cependant une exigence concernant la valeur de *p* [4]. En effet, le calcul en plasticité peut être utilisé dans l'analyse globale des structures ou de leurs éléments, à condition que l'acier

satisfasse entre autres à l'exigence suivante : l'allongement à la rupture doit être supérieur à vingt fois l'allongement élastique. Ceci conduit à considérer :

$$p = 20 \tag{3.29}$$

Concernant la limite élastique, on retient la valeur correspondant à un acier doux :

$$\sigma_0 = 210\,\text{MPa} \tag{3.30}$$

3.3.2 Plastification de la section d'une poutre

Les déformations axiales ε_x d'une poutre en flexion simple varient linéairement sur la hauteur de sa section du fait de l'hypothèse de la conservation de la planéité des sections droites :

$$\varepsilon_x = \chi(x)\cdot(y - y_0) \tag{3.31}$$

Quand le moment de flexion M est incrémenté, la contrainte plastique σ_0 est d'abord atteinte à la fibre extrême la plus éloignée de la ligne moyenne, lorsque ε_x atteint la valeur limite ε_0. Avec une augmentation de M, la plastification se propage formant une zone plastique « supérieure », notée S_{sup}. Quand la déformation ε_x atteint la valeur limite dans l'autre fibre extrême, naît alors une autre zone plastique dite « inférieure » et notée S_{inf}. A ce stade, on a une partie centrale encore élastique et deux zones extrêmes plastifiées. Pour le choix des axes présenté sur la figure (3.5), on a :

- En zone plastique supérieure : $\varepsilon_x \le -\varepsilon_0$ et $\sigma_x = -\sigma_0$
- En zone élastique : $-\varepsilon_0 \le \varepsilon_x \le \varepsilon_0$ et $\sigma_x = E\varepsilon_x$
- En zone plastique inférieure : $\varepsilon_x \ge \varepsilon_0$ et $\sigma_x = \sigma_0$

Pour délimiter ces zones, on note :

- y_1 la position correspondant à $\varepsilon_x = -\varepsilon_0$ et $\sigma_x = -\sigma_0$.
- y_2 la position correspondant à $\varepsilon_x = \varepsilon_0$ et $\sigma_x = \sigma_0$.

Par suite, l'expression de la contrainte en fonction de l'ordonnée s'écrit de la manière suivante :

$$\sigma_x = \begin{cases} -\sigma_0 & \text{pour } y \le y_1 \\ E\cdot\chi\cdot(y - y_0) & \text{pour } y_1 \le y \le y_2 \\ \sigma_0 & \text{pour } y_2 \le y \end{cases} \tag{3.32}$$

Il convient de remarquer que, du fait de la non linéarité du diagramme des contraintes, l'axe neutre (où, $\varepsilon_x = 0$ et $\sigma_x = 0$) doit changer de position y_0 pour conserver l'équilibre de la section. En effet, et conformément à l'hypothèse de flexion simple, la position y_0 de l'axe neutre est actualisée de façon que l'effort normal résultant de la distribution non linéaire des contraintes soit nul.

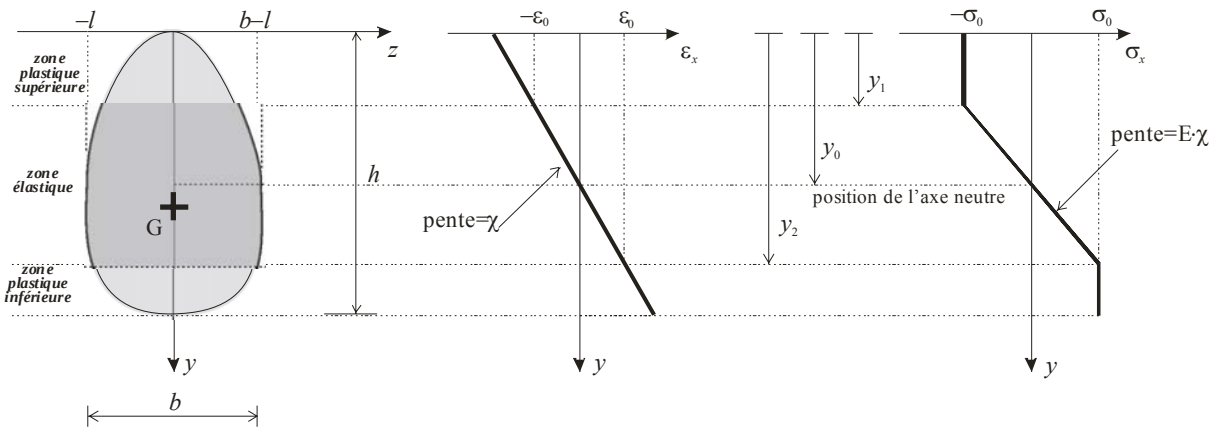


Figure 3.5 : Plastification d'une section de poutre en flexion simple

Avant de continuer, il est aussi important de noter que, dans le critère de plasticité utilisé ci-dessus, l'interaction entre σ_x et τ_{xy} a été négligée. Ceci est bien sûr inexact, mais l'expérience a montré que cet effet n'est pas très important, spécialement pour les poutres minces [16].

3.3.3 Moment plastique

Les deux zones plastiques, S_{sup} et S_{inf}, citées dans le paragraphe précédent peuvent théoriquement progresser jusqu'à se rencontrer lorsque la déformation devient infinie (!). La section est alors dite totalement plastifiée et le moment correspondant à cette situation ultime (et purement théorique du fait de la déformation infinie) est appelé moment plastique M_p.

Il existe alors un axe, dit axe plastique, séparant la zone de contrainte positive, S_{inf}, et celle où elle est négative, S_{sup}. La position ψ de cet axe, qui est la valeur limite de y_0, est déterminée par l'équilibre statique (effort normal nul) de la section totalement plastifiée :

$$\sigma_0 . S_{\text{inf}} - \sigma_0 . S_{\text{sup}} = 0 \tag{3.33}$$

La distribution des contraintes normales dans ce cas limite est :

$$\sigma_x = \begin{cases} -\sigma_0 & \text{pour } y \le \psi \\ +\sigma_0 & \text{pour } y \ge \psi \end{cases} \tag{3.34}$$

Le moment plastique est alors donné par :

$$M_p = \iint_S \sigma_x \cdot (y - y_0) \cdot dS = -\sigma_0 \iint_{S_{\text{sup}}} y \cdot dS + \sigma_0 \iint_{S_{\text{inf}}} y \cdot dS \tag{3.35}$$

On définit le module de flexion plastique W_{pl} par :

$$W_{pl} = \frac{M_p}{\sigma_p} = \frac{M_p}{\sigma_0} \tag{3.36}$$

3.3.4 Introduction des variables sans dimension associées.

Dans le cas général, la relation entre le moment fléchissant et la courbure dépend des caractéristiques géométriques de la section étudiée et des caractéristiques de la loi de comportement du matériau. Afin d'obtenir des équations dans lesquelles apparaissent le moins possible ces grandeurs, on introduit les variables sans dimensions suivantes, indépendamment de la section considérée.

- Variable associée à l'ordonnée y :

$$\eta = \frac{y}{h} \in [0,1] \tag{3.37}$$

- Variable associée à l'abscisse z :

$$\xi = \frac{z}{b} \in [-\lambda, 1-\lambda] \tag{3.38}$$

On peut noter que si l'axe y est un axe de symétrie de la section alors $\lambda = 1/2$. Ces variables adimensionnelles relatives à la géométrie sont représentées dans la figure (3.6.a).

- Variables associées à la déformation ε et à la contrainte σ :

$$e = \frac{\varepsilon}{\varepsilon_p} = \frac{\varepsilon}{p \cdot \varepsilon_0} \tag{3.39}$$

$$s = \frac{\sigma}{\sigma_p} = \frac{\sigma}{\sigma_0} \tag{3.40}$$

Où ε_p et σ_p représentent les valeurs de la déformation et de la contrainte à la rupture, respectivement. Il s'en suit que les valeurs sans dimensions définissant la limite d'élasticité sont :

$$e_0 = \frac{\varepsilon_0}{\varepsilon_p} = \frac{\varepsilon_0}{p \cdot \varepsilon_0} = \frac{1}{p} \tag{3.41}$$

$$s_0 = \frac{\sigma_0}{\sigma_p} = \frac{\sigma_0}{\sigma_0} = 1 \tag{3.42}$$

- Variable associée au module de Young E :

$$\alpha = \frac{\mathrm{E}\varepsilon_p}{\sigma_p} = \frac{\mathrm{E} \cdot p \cdot \varepsilon_0}{\sigma_0} = p \tag{3.43}$$

$$\Rightarrow s_0 = \alpha e_0 \tag{3.44}$$

- Variable associée à l'effort normal N :

$$n = \frac{N}{S\sigma_p} \tag{3.45}$$

S est l'aire de la section.

- Variable associée au moment fléchissant M :

$$m = \frac{M}{M_p} = \frac{M}{W_{pl}\sigma_p} \tag{3.46}$$

M_p et W_{pl} sont le moment plastique et le module de flexion plastique, respectivement.

- Variable associée à la courbure χ :
  En introduisant l'équation (3.37) et (3.39) dans la relation (3.31) on obtient :

$$e = \frac{\chi \cdot h}{\varepsilon_p}(\eta - \eta_0) \tag{3.47}$$

η_0 est la valeur de la variable sans dimension correspondant à y_0.

Cette dernière équation peut être notée comme suit :

$$e = k \cdot (\eta - \eta_0) \tag{3.48}$$

Ce qui représente la relation linéaire entre la déformation adimensionnelle e et l'ordonnée sans dimension η (Figure 3.6.b). La pente de cette droite est la courbure adimensionnelle k :

$$k = \frac{\chi \cdot h}{\varepsilon_p} \tag{3.49}$$

- Expression de la variable sans dimension associée à la contrainte :
  Les variables sans dimensions associées aux ordonnées y_1 et y_2 qui permettent de délimiter les zones plastiques sont :
  - η_1 correspondant à $\varepsilon = -\varepsilon_0$ et $\sigma = -\sigma_0$ $\left(\Leftrightarrow\ e = -e_0 \text{ et } s = -s_0\right)$.
  - η_2 correspondant à $\varepsilon = \varepsilon_0$ et $\sigma = \sigma_0$ $\left(\Leftrightarrow\ e = e_0 \text{ et } s = s_0\right)$.

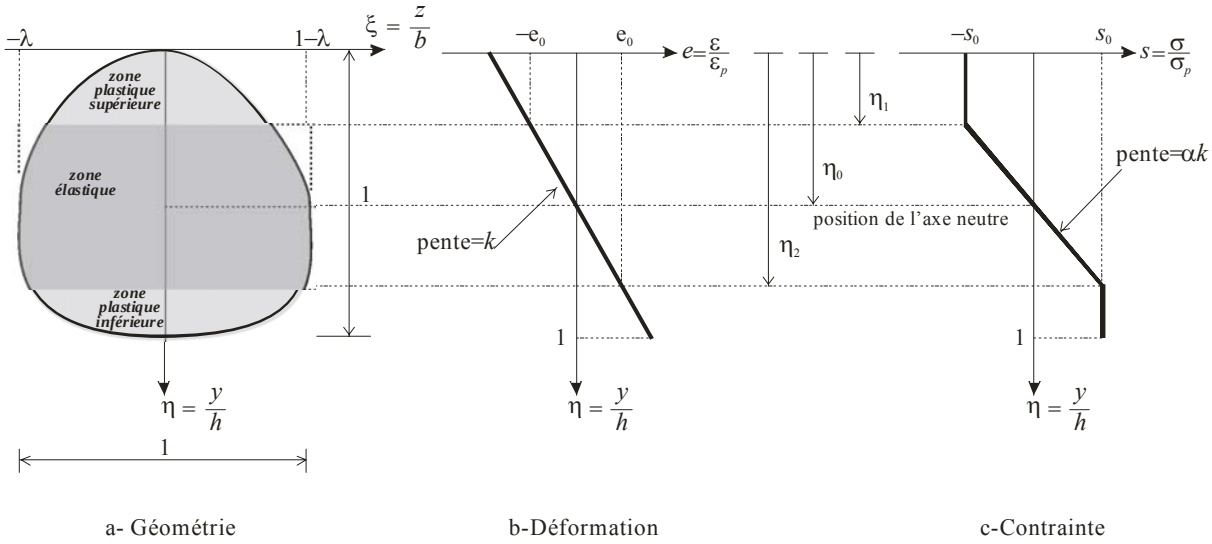
  η_0 est la position sans dimension associée à la position de l'axe neutre y_0.

Si on applique l'équation (3.48) aux deux points $\eta=\eta_1$ et $\eta=\eta_2$, on obtient :

$$\begin{cases} \eta_1 = \eta_0 - \dfrac{e_0}{k} \\ \eta_2 = \eta_0 + \dfrac{e_0}{k} \end{cases} \tag{3.50}$$

Par suite, l'expression de la variable sans dimension associée à la contrainte s'écrit en fonction de l'ordonnée adimensionnelle de la manière suivante (Figure 3.6.c) :

$$s = \begin{cases} -s_0 & \text{pour } \eta \le \eta_1 \\ \alpha k(\eta - \eta_0) & \text{pour } \eta_1 \le \eta \le \eta_2 \\ s_0 & \text{pour } \eta_2 \le \eta \end{cases} \tag{3.51}$$



a- Géométrie b-Déformation c-Contrainte

Figure 3.6 : Variables adimensionnelles.

3.4 Étude de la section triangulaire

3.4.1 Définition de la géométrie et calculs préliminaires

On considère une poutre de section triangulaire de hauteur *h* et de base *b* comme le montre la figure (3.7). Cette dernière donne également la forme sans dimension associée.

Préalablement à la détermination des équations reliant le moment fléchissant sans dimension et la courbure adimensionnelle, il est nécessaire de calculer l'aire, l'inertie par rapport au centre de gravité et le moment plastique de la section.

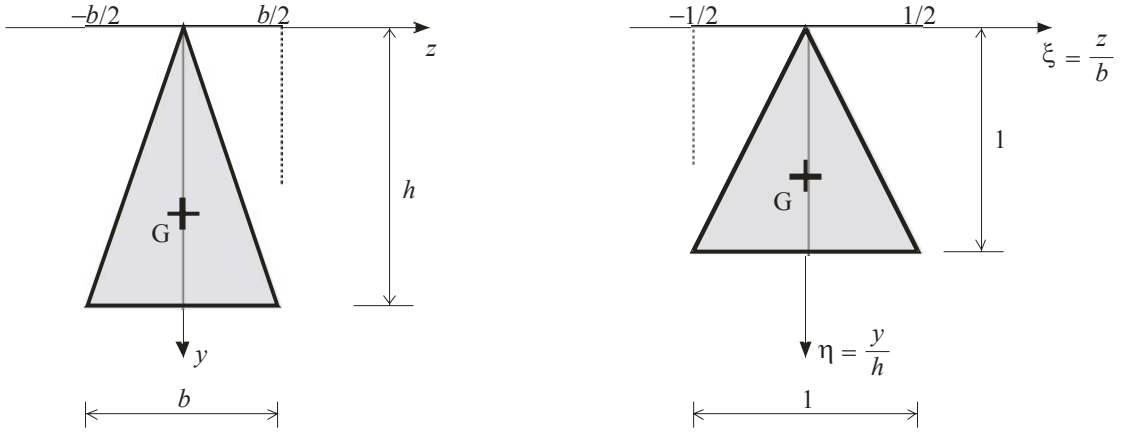


Figure 3.7 : Caractéristiques géométriques et choix des axes de la section triangulaire.

La surface du triangle est donnée par :

$$S = \frac{b \cdot h}{2} \tag{3.52}$$

L'inertie par rapport à l'axe O*z* est égale à :

$$I_Z = \iint_S y^2 .\mathrm{d}S = \int_{y=0}^{h} \int_{-\frac{yb}{2h}}^{\frac{yb}{2h}} y^2 \mathrm{d}z\mathrm{d}y = \frac{b}{h} \int_{y=0}^{h} y^3 \mathrm{d}y$$

$$I_Z = \frac{bh^3}{4}$$

Par application du théorème de Köenig ($I=I_Z-Sd^2$), l'inertie par rapport à l'axe passant par le centre de gravité est donnée par :

$$I = \frac{bh^3}{4} - Sd^2 = \frac{bh^3}{36} \tag{3.53}$$

Pour calculer le moment plastique, on suppose la section totalement plastifiée divisée par un axe plastique en deux surfaces S_{sup} et S_{inf}. L'ordonnée ψ de cet axe parallèle à O*z* est calculée de façon à conserver l'équilibre statique de la section (équation 3.33).

$$S_{\text{sup}} = S_{\text{inf}} \Rightarrow \psi = \frac{h}{\sqrt{2}} \tag{3.54}$$

D'après l'équation (3.35), le moment plastique s'écrit :

$$M_p = \iint_S \sigma y dS = -\sigma_p \iint_{S_{\text{sup}}} y dS + \sigma_p \iint_{S_{\text{inf}}} y dS = -\sigma_p \frac{b}{h} \int_0^{\psi} y^2 dy + \sigma_p \frac{b}{h} \int_{\psi}^{h} y^2 dy$$

$$M_p = \frac{\sigma_p b h^2}{3\beta} \Rightarrow W_{pl} = \frac{bh^2}{3\beta} \tag{3.55}$$

$$\text{Avec } \beta = 2 + \sqrt{2}$$

3.4.2 Expressions des efforts internes adimensionnels.

L'expression de l'effort normal est donnée par l'équation (3.3). En utilisant les variables sans dimension définies précédemment on obtient :

$$N = \iint_S \sigma dS = \int_0^h \int_{\frac{-yb}{2h}}^{\frac{yb}{2h}} \sigma dz dy = \frac{b}{h} \int_0^h \sigma y dy$$

$$N = \frac{b}{h} \sigma_p \int_0^1 s\eta h^2 d\eta$$

$$N = bh\sigma_p \int_0^1 s\eta d\eta$$

Connaissant l'expression de la surface et en utilisant la définition de l'effort normal sans dimension, donnée par l'équation (3.45), on obtient :

$$n = 2\int_0^1 s\eta d\eta \tag{3.56}$$

On introduit à présent les variables adimensionnelles dans l'expression du moment fléchissant :

$$M = \iint_S \sigma y dS = \int_0^h \int_{-\frac{yb}{2h}}^{\frac{yb}{2h}} \sigma . y . dz . dy$$

$$M = \frac{b}{h} \int_0^h \sigma y^2 dy = \frac{b}{h} \int_0^h \sigma_p s y^2 dy$$

$$M = \frac{b}{h} \sigma_p \int_0^1 s\eta^2 h^3 d\eta$$

$$M = bh^2 \sigma_p \int_0^1 s\eta^2 d\eta$$

La variable sans dimension associée au moment fléchissant est alors donnée par :

$$m = \frac{M}{M_p} = 3\beta \int_0^1 s\eta^2 d\eta \tag{3.57}$$

3.4.3 Formulation de la relation moment - courbure

Le travail à effectuer maintenant consiste à élaborer l'équation constitutive des sections triangulaires reliant le moment sans dimension à la courbure adimensionnelle, afin qu'elle soit la plus générale possible, en tenant compte de la loi de comportement élasto-plastique parfaite.

D'après ce qui a été dit précédemment concernant la plastification des sections des poutres en flexion, lorsque le moment de flexion est incrémenté on passe successivement par :

a) une phase élastique où toutes les déformations de la section restent inférieures à ε_0, la limite d'élasticité. Elle correspond à des « zones de plastification en dehors de la section » : $\eta_1 \leq 0$ et $\eta_2 \geq 1$
b) une phase de plastification en partie supérieure : $\eta_1 \geq 0$ et $\eta_2 > 1$
c) une phase de plastification en fibre inférieure : $\eta_1 > 0$ et $\eta_2 \leq 1$
d) l'état ultime indiquant que la déformation de la section a atteint ε_p, la valeur critique de la déformation au-delà de laquelle il y'a rupture du matériau.

3.4.3.1 Phase élastique.

Ni la fibre supérieure ni la fibre inférieure n'étant plastifiée, on se trouve dans des états de contrainte et de déformation élastiques. On a donc $\eta_1 \leq 0$ et $\eta_2 \geq 1$, ce qui permet d'écrire la relation (3.51) sous la forme suivante :

$$s = \alpha k(\eta - \eta_0) \qquad \text{pour } 0 \leq \eta \leq 1 \tag{3.58}$$

On peut alors expliciter les expressions de l'effort normal (3.56) et du moment fléchissant (3.57) :

$$n = 2\alpha k \int_0^1 (\eta - \eta_0)\eta d\eta = \alpha k\left(\frac{2}{3} - \eta_0\right)$$

$$m = 3\alpha\beta k \int_0^1 (\eta - \eta_0)\eta^2 d\eta = \alpha\beta k\left(\frac{3}{4} - \eta_0\right)$$

L'hypothèse de la flexion simple, compte tenu de l'expression de n, conduit à η_0=2/3, ce qui correspond à la position du centre de gravité de la section triangulaire, c'est-à-dire à la ligne moyenne. En introduisant cette valeur à l'expression de m on obtient :

$$m = \frac{\alpha\beta}{12}k$$

Grâce à l'hypothèse sur la valeur de la déformation à rupture introduite dans l'équation (3.43), on aboutit à la relation moment - courbure pendant cette phase élastique :

$$m = \frac{p\beta}{12}k \tag{3.59}$$

Il s'agit à présent de trouver les valeurs des différents paramètres adimensionnels correspondant à la fin de cette phase qui est atteinte lorsque la fibre supérieure se plastifie, ce qui correspond à $\eta = \eta_1 = 0$. C'est-à-dire lorsque la contrainte s donnée par l'équation (3.58) atteint la limite élastique en compression $-s_0$. On trouve alors les valeurs finales suivantes :

$$\begin{cases} \eta_{0_{01}} = \dfrac{2}{3} \qquad \eta_{1_{01}} = 0 \qquad \eta_{2_{01}} = \dfrac{4}{3} \\ k_{01} = \dfrac{3}{2p} \qquad m_{01} = \dfrac{2+\sqrt{2}}{8} \end{cases} \tag{3.60}$$

3.4.3.2 Plastification en partie supérieure.

Dans cette phase, la section comporte une zone plastifiée localisée dans sa partie supérieure et une zone élastique. On a donc $\eta_1 \geq 0$ et $\eta_2 > 1$, ce qui permet d'écrire la relation (3.51) sous la forme suivante :

$$s = \begin{cases} -s_0 & \text{pour } 0 \leq \eta_1 \\ \alpha k(\eta - \eta_0) & \text{pour } \eta_1 \leq \eta \leq 1 \end{cases} \tag{3.61}$$

On peut alors expliciter l'expression (3.56) de l'effort normal :

$$n = 2\left[\int_0^{\eta_1}(-s_0\eta d\eta + \int_{\eta_1}^{1}\alpha k(\eta-\eta_0)\eta d\eta\right]$$

$$n = -s_0\eta_1^2 + \alpha k\left[\left(\frac{2}{3}-\eta_0\right)-\left(\frac{2}{3}\eta_1-\eta_0\right)\eta_1^2\right] \tag{3.62}$$

L'expression (3.57) du moment fléchissant s'écrit à son tour :

$$m = 3\beta\left[\int_0^{\eta_1} -s_0\eta^2 d\eta + \int_{\eta_1}^{1} \alpha k(\eta-\eta_0)\eta^2 d\eta\right]$$

$$m = \beta(-s_0)\eta_1^3 + \beta\alpha k\left(\left(\frac{3}{4}-\eta_0\right)-\left(\frac{3}{4}\eta_1-\eta_0\right)\eta_1^3\right) \tag{3.63}$$

En annulant l'effort normal suite à l'hypothèse de flexion simple, on a :

$$s_0\eta_1^2 = \alpha k\left[\left(\frac{2}{3}-\eta_0\right)-\left(\frac{2}{3}\eta_1-\eta_0\right)\eta_1^2\right]$$

En tenant compte des relations (3.44) et (3.50), on arrive à trouver une expression de k en fonction de la seule variable η_1 :

$$k = \frac{3e_0}{2-3\eta_1+\eta_1^3} \tag{3.64}$$

L'utilisation de ce résultat dans le système (3.50) permet d'exprimer les variables η_0 et η_2 en fonction de la variable unique η_1 :

$$\begin{cases} \eta_0 = \dfrac{2+\eta_1^3}{3} \\ \eta_2 = \dfrac{4-3\eta_1+2\eta_1^3}{3} \end{cases} \tag{3.65}$$

De même, il est possible d'exprimer m en fonction de cette unique variable, en introduisant les relations (3.64) et (3.65) dans l'expression (3.63) :

$$m = \alpha\beta e_0 \frac{3\eta_1^2+2\eta_1+1}{4(\eta_1+2)} \tag{3.66}$$

Pour résumer, pendant cette phase de plastification en partie supérieure on obtient le système d'équations paramétriques suivant, après utilisation des équations (3.41) et (3.43) :

$$\begin{cases} \eta_0 = \dfrac{2+\eta_1^3}{3} \\ \eta_2 = \dfrac{4-3\eta_1+2\eta_1^3}{3} \\ k = \dfrac{3}{p\left(2-3\eta_1+\eta_1^3\right)} \\ m = \beta\dfrac{3\eta_1^2+2\eta_1+1}{4\left(\eta_1+2\right)} \end{cases} \tag{3.67}$$

Les valeurs finales de cette phase sont atteintes lorsque la fibre inférieure entre en plastification c'est-à-dire lorsque η_2 atteint la valeur de 1. Considérant cette valeur, la deuxième équation du système (3.67) conduit à déterminer la valeur finale de la variable η_1, notée $\eta_{1_{12}}$, solution de l'équation du troisième degré suivante :

$$2\eta_1^3-3\eta_1+1=\left(\eta_1-1\right)\left(\eta_1+\frac{1-\sqrt{3}}{2}\right)\left(\eta_1+\frac{1+\sqrt{3}}{2}\right)=0$$

Lors de la plastification de la fibre inférieure, la relation suivante est toujours vérifiée :

$$0<\eta_1<\eta_0<\eta_2=1$$

Donc l'unique valeur possible est :

$$\eta_{1_{12}}=\frac{\sqrt{3}-1}{2}$$

En introduisant cette valeur dans le système (3.67), on obtient l'expression suivante des valeurs finales :

$$\begin{cases} \eta_{0_{12}} = \dfrac{1+\sqrt{3}}{4} \\ \eta_{1_{12}} = \dfrac{\sqrt{3}-1}{2} \\ \eta_{2_{12}} = 1 \\ k_{12} = \dfrac{2\left(3+\sqrt{3}\right)}{3p} \\ m_{12} = \dfrac{\left(7-3\sqrt{3}\right)\beta}{8} \end{cases} \tag{3.68}$$

3.4.3.3 Plastification en partie inférieure.

Les fibres inférieures et supérieures sont plastifiées. On a alors $\eta_1>0$ et $\eta_2\leq 1$. L'expression de la contrainte adimensionnelle est donnée par la relation (3.51), ce qui permet d'expliciter l'expression (3.56) de l'effort normal :

$$n = 2\left[\int_0^{\eta_1}(-s_0)\eta d\eta + \int_{\eta_1}^{\eta_2}\alpha k(\eta-\eta_0)\eta d\eta + \int_{\eta_2}^{1}(s_0)\eta d\eta\right]$$

$$n = -s_0\eta_1^2 + \alpha k\left[\frac{2}{3}\left(\eta_2^3-\eta_1^3\right)-\eta_0\left(\eta_2^2-\eta_1^2\right)\right]+(s_0)\left(1-\eta_2^2\right)$$

$$n = \alpha k\left[\frac{2}{3}\left(\eta_2^3-\eta_1^3\right)+\eta_0\left(\eta_1^2-\eta_2^2\right)\right]+(s_0)\left(1-\eta_1^2-\eta_2^2\right) \qquad (3.69)$$

De même, on explicite l'expression (3.57) du moment fléchissant :

$$m = 3\beta\left[\int_{-1}^{\eta_1}(-s_0)\eta^2 d\eta + \int_{\eta_1}^{\eta_2}\alpha k(\eta-\eta_0)\eta^2 d\eta + \int_{\eta_2}^{1}(s_0)\eta^2 d\eta\right]$$

$$m = \beta\left[(-s_0)\eta_1^3 + \alpha k\left[\frac{3}{4}\left(\eta_2^4-\eta_1^4\right)-\eta_0\left(\eta_2^3-\eta_1^3\right)\right]+(s_0)\left(1-\eta_2^3\right)\right]$$

$$m = \beta\left[\alpha k\left(\frac{3}{4}\left(\eta_2^4-\eta_1^4\right)+\eta_0\left(\eta_1^3-\eta_2^3\right)\right)+(s_0)\left(1-\eta_1^3-\eta_2^3\right)\right] \qquad (3.70)$$

En annulant l'effort normal (3.69) suite à l'hypothèse de flexion simple, on a :

$$(-s_0)\left(1-\eta_1^2-\eta_2^2\right) = k\alpha\left(\frac{2}{3}\left(\eta_2^3-\eta_1^3\right)+\eta_0\left(\eta_1^2-\eta_2^2\right)\right)$$

En introduisant les équations (3.44) et (3.50) dans cette relation, on obtient :

$$\left(1-2\eta_1^2\right)k^2 - 4\eta_1 e_0 k - \frac{8}{3}e_0^2 = 0$$

La courbure adimensionnelle recherchée est une solution de cette équation du second degré. Les deux racines sont les suivantes :

$$k_1 = 2e_0 \frac{\eta_1 + \sqrt{\frac{2-\eta_1^2}{3}}}{1-2\eta_1^2} \qquad \text{et} \qquad k_2 = 2e_0 \frac{\eta_1 - \sqrt{\frac{2-\eta_1^2}{3}}}{1-2\eta_1^2}$$

De ces deux valeurs on retiendra celle qui assure la continuité avec la phase de plastification en fibre supérieure, c'est-à-dire celle qui fournit une valeur égale à k_{12} lorsqu'on remplace η_1 par $\eta_{1_{12}}$, k_{12} et $\eta_{1_{12}}$ sont données parmi les relations (3.68)

$$k_1\left(\eta_{1_{12}}\right) = \frac{2}{p} \frac{\frac{\sqrt{3}-1}{2} + \sqrt{\frac{2-\left(\frac{\sqrt{3}-1}{2}\right)^2}{3}}}{1-2\left(\frac{\sqrt{3}-1}{2}\right)^2} = \frac{3-\sqrt{3}-2\sqrt{1+\frac{\sqrt{3}}{2}}}{p\left(3-\sqrt{3}\right)}$$

En remarquant que $1+\frac{\sqrt{3}}{2} = \left(\frac{1+\sqrt{3}}{2}\right)^2$, on a :

$$k_1\left(\eta_{1_{12}}\right) = \frac{2\left(3+\sqrt{3}\right)}{3p}$$

Ce qui est égal à k_{12}. On peut alors affirmer que, pendant cette phase, $k=k_1$.

On introduit, dans un premier temps, cette expression de k dans le système (3.50) pour exprimer η_0 et η_2 en fonction de l'unique variable η_1. On explicite ensuite la relation (3.70). Après utilisation des équations (3.41) et (3.43), on obtient alors le système d'équations paramétriques suivant :

$$\begin{cases} k = \dfrac{2\left(\eta_1 + \sqrt{\dfrac{2-\eta_1^2}{3}}\right)}{p\left(1-2\eta_1^2\right)} \\ \eta_0 = \dfrac{\eta_1 + \sqrt{3\left(2-\eta_1^2\right)}}{4} \\ \eta_2 = \dfrac{\sqrt{3\left(2-\eta_1^2\right)} - \eta_1}{2} \\ \dfrac{m}{\beta} = \dfrac{4\sqrt{2-\eta_1^2}\left(-2\eta_1^5 + \eta_1^3 + 4\eta_1^2 - 3\eta_1 + 1\right) - 3\sqrt{3}\left(2\eta_1^2 - 4\eta_1 + 1\right)}{4\left(\sqrt{2-\eta_1^2}\left(4\eta_1^2+1\right) + 3\sqrt{3}\eta_1\right)} \end{cases} \tag{3.71}$$

Si le matériau considéré avait un comportement élasto-plastique parfait idéal, c'est-à-dire sans limite de rupture, les valeurs ultimes correspondraient à une section totalement plastifiée (équations 3.54 et 3.55) :

$$\begin{cases} M_{ult} = M_p \quad \text{donc} \quad m_{ult} = 1 \\ \eta_{0_{\upsilon\lambda\tau}} = \eta_{1_{ult}} = \eta_{2_{ult}} = \dfrac{1}{\sqrt{2}} \end{cases}$$

Dans ce mémoire, la loi de comportement retenue limite les déformations. Les valeurs ultimes des différentes grandeurs caractéristiques sont celles calculées lorsque la condition limite $|\varepsilon| = \varepsilon_p$ est atteinte. La déformation au niveau de la fibre supérieure étant la plus grande en valeur absolue, cette condition s'écrit :

$$\varepsilon(y=0) = -\varepsilon_p$$

Il est aussi possible de l'écrire, d'après (3.48), sous la forme suivante :

$$e(\eta=0) = -1 = -k\eta_0 \Rightarrow k\eta_0 = 1$$

En utilisant les deux premières équations du système (3.71) dans cette condition limite, on arrive à une condition sur η_1 seul :

$$4\eta_1^4\left(3p^2+1\right) - 4\eta_1^2\left[3p(p-1)+2\right] + 3(p-1)^2 = 0$$

Cette équation possède les quatre racines suivantes :

$$\begin{cases} \eta_1 = \pm\dfrac{1}{\sqrt{2}} \\ \eta_1 = \pm\sqrt{\dfrac{3}{2}}\dfrac{p-1}{\sqrt{3p^2+1}} \end{cases}$$

Sachant que η_1 ne peut prendre que des valeurs positives et que la valeur $1/\sqrt{2}$ correspond au cas où les déformations ne sont pas limitées, on peut affirmer que la valeur ultime ne peut être que la suivante :

$$\eta_{1_{ult}} = \sqrt{\frac{3}{2}}\frac{p-1}{\sqrt{3p^2+1}}$$

En introduisant cette valeur dans les relations (3.71), on obtient les expressions de l'ensemble des valeurs ultimes qui sont données dans ce qui suit.

$$\begin{cases} \eta_{0_{ult}} = p\sqrt{\dfrac{3}{2\left(3p^2+1\right)}} \\ \eta_{1_{ult}} = \left(p-1\right)\sqrt{\dfrac{3}{2\left(3p^2+1\right)}} \\ \eta_{2_{ult}} = \left(p+1\right)\sqrt{\dfrac{3}{2\left(3p^2+1\right)}} \\ k_{ult} = \dfrac{1}{p}\sqrt{\dfrac{2}{3}\left(3p^2+1\right)} \\ \dfrac{m_{ult}}{\beta} = 1-3\sqrt{\dfrac{3}{2}}\times\dfrac{p\left(p^2+1\right)}{\left(3p^2+1\right)^{\frac{3}{2}}} \end{cases} \tag{3.72}$$

A ce niveau il faut noter que les relations ci-dessus ne peuvent être retenues comme valeurs ultimes que si :

$$\eta_{1_{ult}} \geq \eta_{1_{12}}$$

Autrement dit que si la valeur de la déformation de ruine est effectivement atteinte lors de la seconde phase plastique et pas avant. On arrive à montrer que ceci est vérifié pour autant que :

$$p > 1{,}98$$

Comme on prend $p = 20$ (équation 3.29) dans les applications numériques de ce mémoire, on est certain d'avoir deux phases plastiques dans la relation constitutive de la flexion simple des poutres à section triangulaire. Cette relation est représentée dans la figure (3.8). On y distingue les trois phases : élastique, plastique 1 (en fibre supérieure) et plastique 2 (en fibre inférieure) définies, respectivement, par les relations (3.59) (3.67) (3.71).

Les coordonnées des trois points caractéristiques A, B et C délimitant les différentes phases sont données par les relations (3.60), (3.68) et (3.72), respectivement :

$$A\begin{cases} k_A = \dfrac{3}{2p} \\ m_A = \dfrac{2+\sqrt{2}}{8} \end{cases}, \quad B\begin{cases} k_B = \dfrac{2\left(3+\sqrt{3}\right)}{3p} \\ m_B = \dfrac{\left(2+\sqrt{2}\right)\left(7-3\sqrt{3}\right)}{8} \end{cases} \quad et \quad C\begin{cases} \mathrm{k_C} = \dfrac{1}{p}\sqrt{\dfrac{2\left(3p^2+1\right)}{3}} \\ m_C = \left(2+\sqrt{2}\right)\left[1-\dfrac{3p\left(p^2+1\right)}{3p^2+1}\sqrt{\dfrac{3}{2\left(3p^2+1\right)}}\right] \end{cases}$$

Remarque :

Dans le cas particulier où p=20, on a $M_{ult} = 0.997M_p$

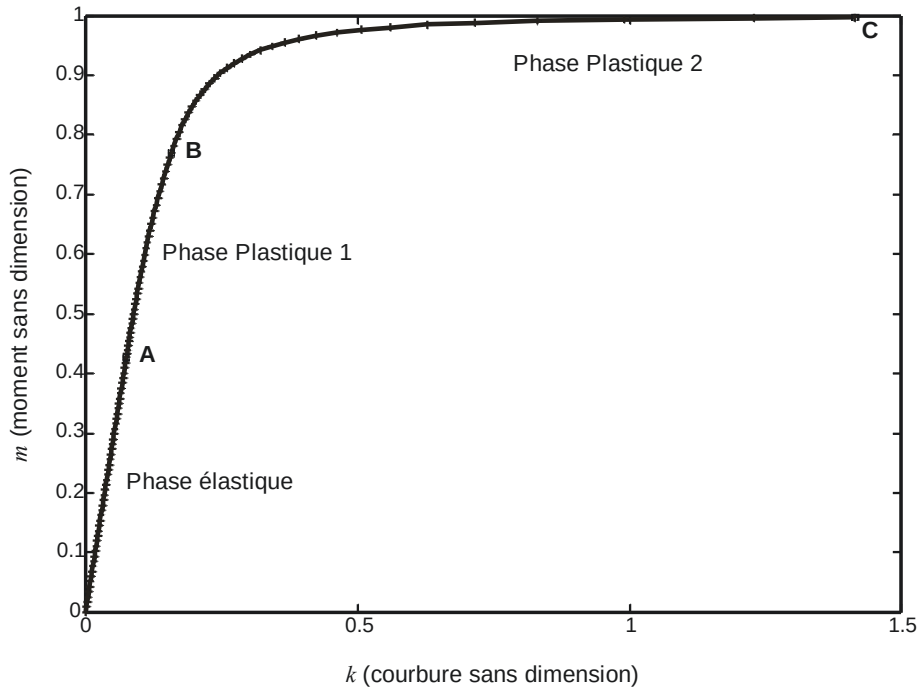


Figure 3.8 : Courbe moment sans dimension/courbure sans dimension. (*p*=20)

3.5 Étude de la section en T.

3.5.1 Introduction

Le travail qui vient d'être présenté pour la section triangulaire va à présent être effectué pour les sections en T. Celles-ci sont évidemment beaucoup plus courantes dans les structures de génie civil et leur étude présente un intérêt pratique certain. La principale différence par rapport au précédent cas réside dans la discontinuité de la géométrie en fonction de *y*. Cette particularité impliquera une discussion selon différents cas de figures.

3.5.2 Définition de la géométrie et calculs préliminaires

On considère une poutre à section en forme de T, de hauteur totale *h*, de base *b*, d'épaisseur de semelle t_f et d'épaisseur d'âme t_w montrée par la figure (3.9). Cette dernière donne également la forme de la section adimensionnelle et les variables sans dimension associées à :

- l'épaisseur de l'âme : $\tau = \dfrac{t_w}{b}$
- l'épaisseur de la semelle : $\zeta = \dfrac{t_f}{h}$
- la surface : $\Sigma = \dfrac{S}{bh} = \tau(1-\zeta)+\zeta$

A celles-ci on ajoute deux variables adimensionnelles utiles à la représentation ultérieure de la formulation sans dimension :

- $\Gamma = 1 - (1-\tau)(1-\zeta)^2$
- $\Pi = 1 - (1-\tau)(1-\zeta)^3$

Ainsi, on considère une poutre adimensionnelle de hauteur 1, de base 1, d'épaisseur de semelle τ et d'épaisseur d'âme ζ, comme le montre la figure 3.9

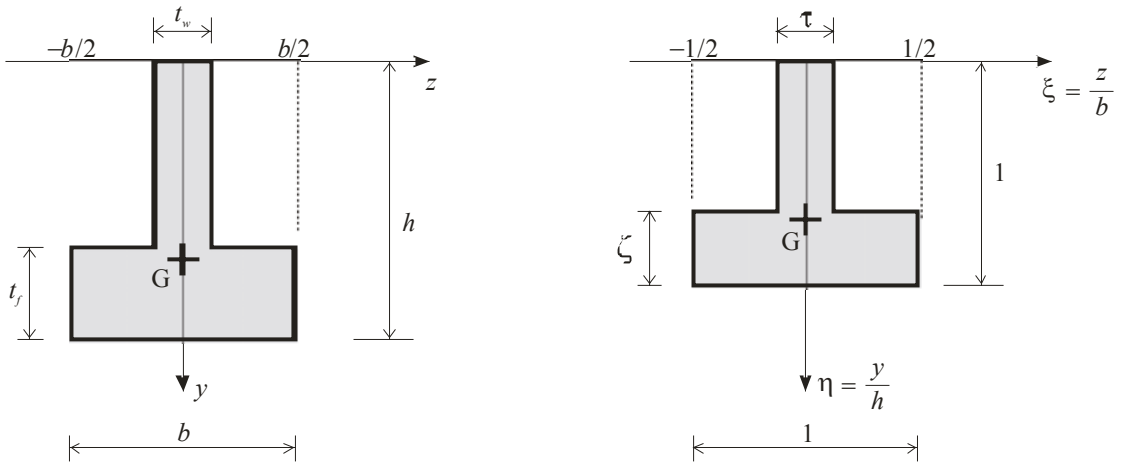


Figure 3.9 : Caractéristiques géométriques et choix des axes de la section en T.

Pour établir les relations reliant le moment à la courbure, et pour calculer les valeurs caractéristiques de ces deux variables, on doit tout d'abord donner l'expression du centre de gravité afin d'expliciter l'expression de l'inertie. Ensuite, on détermine la position de l'axe plastique dans le but d'exprimer le moment plastique.

On arrive aisément à l'expression de la position du centre de gravite ci-dessous :

$$y_G = \frac{1}{2}\frac{(h-t_f)^2 t_w + b t_f (2h - t_f)}{(h-t_f)\cdot t_w + b t_f} \tag{3.80}$$

L'inertie par rapport à l'axe O*z* est égale à :

$$I_Z = \frac{1}{3} t_w (h-t_f)^3 + \frac{1}{3} b\left(h^3 - (h-t_f)^3\right)$$

Par application du théorème de Köenig ($I=I_Z-Sd^2$), l'inertie par rapport à l'axe passant par le centre de gravité est donnée par :

$$I = \frac{bh^3}{12}\frac{4\Sigma\Pi - 3\Gamma^2}{\Sigma} \tag{3.81}$$

Pour calculer le moment plastique, on suppose la section totalement plastifiée divisée par un axe plastique en deux surfaces S_{sup} et S_{inf}. L'ordonnée ψ de cet axe parallèle à O*z* est la

valeur au delà (en deçà) de laquelle la contrainte est positive (négative), égale en valeur absolue à la contrainte plastique. Il faut donc considérer trois cas, conformément à la figure (3.10) :

- L'axe plastique est situé dans l'âme, la section est alors dite de classe A :

$$\psi < h - t_f \tag{3.82}$$

- L'axe plastique est situé à la jonction de l'âme et de la semelle, la section sera dite de classe B :

$$\psi = h - t_f \tag{3.83}$$

- L'axe plastique est situé dans la semelle, la section est dite de classe C :

$$\psi > h - t_f \tag{3.84}$$

Cette ordonnée ψ est calculée de façon à conserver l'équilibre statique de la section (équation 3.33) pour chaque classe.

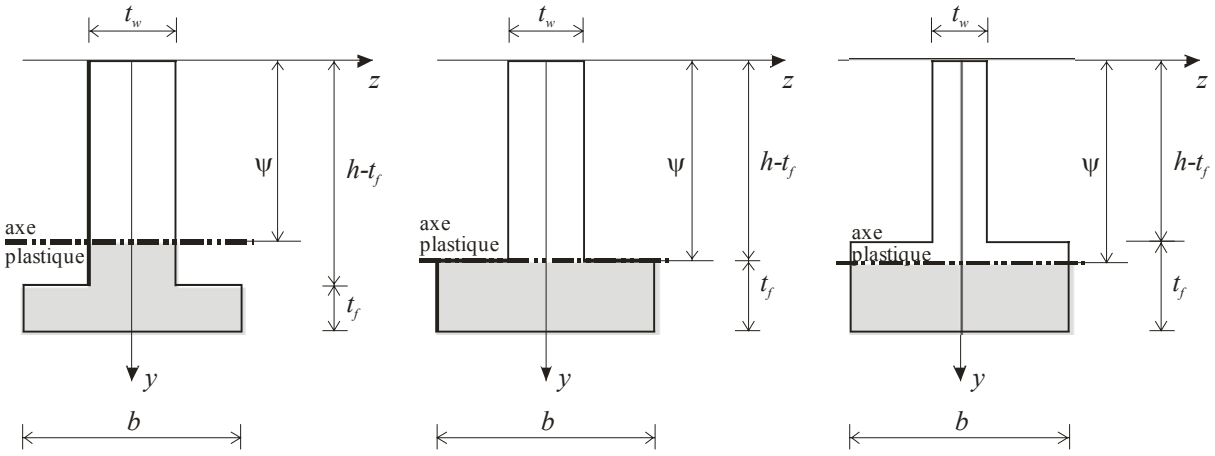


Figure 3.10 : Différentes positions de l'axe plastique

3.5.2.1 Section de classe A

Pour la section de classe A, l'axe plastique se situe dans l'âme, donc $\psi < h - t_f$

$$\begin{aligned} S_{\sup} &= \psi t_w \\ S_{\inf} &= (h - t_f - \psi) t_w + b t_f \end{aligned}$$

$$S_{\sup} = S_{\inf} \Rightarrow \psi = \frac{(h - t_f) t_w + b t_f}{2 t_w}$$

Pour qu'une section soit dite de classe A, il faut que $t_w(h - t_f) - b t_f > 0$ ou en variables adimensionnelles :

$$\tau(1 - \zeta) - \zeta > 0 \tag{3.85}$$

D'après l'équation (3.35), le moment plastique s'écrit :

$$M_p = \iint_S \sigma y dS = -\sigma_p \iint_{S_{\sup}} y dS + \sigma_p \iint_{S_{\inf}} y dS = -\sigma_p t_w \int_0^{\psi} y dy + \sigma_p t_w \int_{\psi}^{h-t_f} y dy + \sigma_p t_w \int_{h-t_f}^{h} y dy$$

$$\begin{cases} M_p = \dfrac{\sigma_p b h^2}{4\beta} \\ \beta = \dfrac{\tau}{\tau^2 (1 - \zeta)^2 + \zeta(2\tau - \zeta)} \end{cases} \tag{3.86}$$

3.5.2.2 Section de classe B.

Pour la section de classe B, l'axe plastique se situe à la jonction de l'âme et la semelle, donc $\psi = h - t_f$.

$$\begin{aligned} S_{\sup} &= (h - t_f) t_w \\ S_{\inf} &= b t_f \end{aligned}$$

Pour qu'une section soit dite de classe B, il faut que $t_w(h - t_f) - b t_f = 0$ ou en variables adimensionnelles :

$$\tau(1 - \zeta) - \zeta = 0 \tag{3.87}$$

D'après l'équation (3.35), le moment plastique s'écrit :

$$M_p = -\sigma_p t_w \int_0^{h-t_f} y dy + \sigma_p b \int_{h-t_f}^{h} y dy$$

$$\begin{cases} M_p = \dfrac{\sigma_p b h^2}{4\beta} \\ \beta = \dfrac{1}{2\left[\zeta(2-\zeta) - \tau(1-\zeta)^2\right]} \end{cases} \tag{3.88}$$

3.5.2.3 Section de classe C.

Pour la section de classe C, l'axe plastique se situe dans la semelle, donc $\psi > h - t_f$.

$$S_{\sup} = (h - t_f) t_w + \left[\psi - (h - t_f)\right] b$$
$$S_{\inf} = (h - \psi) b$$

Pour qu'une section soit dite de classe C, il faut que $t_w(h - t_f) - b t_f < 0$ ou en variables adimensionnelles :

$$\tau(1-\zeta) - \zeta < 0 \tag{3.89}$$

D'après l'équation (3.35), le moment plastique s'écrit :

$$M_p = -\sigma_p t_w \int_0^{h-t_f} y dy - \sigma_p b \int_{h-t_f}^{\psi} y dy + \sigma_p b \int_{\psi}^{h} y dy$$

D'où :

$$\begin{cases} M_p = \dfrac{\sigma_p b h^2}{4\beta} \\ \beta = \dfrac{1}{\zeta^2 + \tau(1-\zeta)\left[2 - \tau(1-\zeta)\right]} \end{cases} \tag{3.90}$$

3.5.3 Expressions des efforts internes adimensionnels.

L'expression de l'effort normal est donnée par l'équation (3.3). En utilisant les variables sans dimension définies précédemment on obtient :

$$N = \iint_S \sigma dS = \int_0^{h-t_f} \int_{\frac{-t_w}{2}}^{\frac{t_w}{2}} \sigma dz dy + \int_{h-t_f}^{h} \int_{\frac{-b}{2}}^{\frac{b}{2}} \sigma dz dy = t_w \int_0^{h-t_f} \sigma dy + \int_{h-t_f}^{h} \sigma dy$$

$$N = ht_w \int_0^{1-\zeta} \sigma_p s d\eta + bh \int_{1-\zeta}^{1} \sigma_p s d\eta$$

$$N = bh\sigma_p \tau \int_0^{1-\zeta} s d\eta + bh\sigma_p \int_{1-\zeta}^{1} s d\eta$$

Connaissant l'expression de la surface et en utilisant la définition de l'effort normal sans dimension, donnée par l'équation (3.45), on obtient :

$$n\Sigma = \tau \int_0^{1-\zeta} s d\eta + \int_{1-\zeta}^{1} s d\eta \tag{3.91}$$

On introduit à présent les variables adimensionnelles dans l'expression du moment fléchissant :

$$M = \iint_S \sigma y dS = \int_0^{h-t_f} \int_{\frac{-t_w}{2}}^{\frac{t_w}{2}} \sigma.y.dz.dy + \int_{h-t_f}^{h} \int_{\frac{-b}{2}}^{\frac{b}{2}} \sigma.y.dz.dy = t_w \int_0^{h-t_f} \sigma.y.dy + b \int_{h-t_f}^{h} \sigma.y.dy$$

$$M = h^2 t_w \int_0^{1-\zeta} \sigma_p s\eta d\eta + bh^2 \int_{1-\zeta}^{1} \sigma_p s\eta d\eta$$

$$M = bh^2\sigma_p \tau \int_0^{1-\zeta} s\eta d\eta + bh^2\sigma_p \int_{1-\zeta}^{1} s\eta d\eta$$

La variable sans dimension associée au moment fléchissant est alors donnée par :

$$m = \frac{M}{M_p} = 4\beta \left[\tau \int_0^{1-\zeta} s\eta d\eta + \int_{1-\zeta}^{1} s\eta d\eta \right] \tag{3.92}$$

3.5.4 Formulation de la relation moment - courbure

Comme pour les sections triangulaires, le travail à effectuer maintenant consiste à élaborer l'équation constitutive des sections en forme de T reliant le moment sans dimension à la courbure adimensionnelle, afin qu'elle soit la plus générale possible, en tenant compte de la loi de comportement élasto-plastique parfaite.

On rappelle que concernant la plastification des sections des poutres en flexion, lorsque le moment de flexion est incrémenté on passe successivement par :

a) une phase élastique où toutes les déformations de la section restent inférieures à ε_0, la limite d'élasticité. Elle correspond à des « zones de plastification en dehors de la section » : $\eta_1 \leq 0$ et $\eta_2 \geq 1$
b) une phase de plastification en partie supérieure : $\eta_1 \geq 0$ et $\eta_2 > 1$
c) une phase de plastification en fibre inférieure : $\eta_1 > 0$ et $\eta_2 \leq 1$
d) l'état ultime indiquant que la déformation de la section a atteint ε_p, la valeur critique de la déformation au-delà de laquelle il y'a rupture du matériau.

3.5.4.1 Phase élastique.

Ni la fibre supérieure ni la fibre inférieure n'étant plastifiée, on se trouve dans des états de contrainte et de déformation élastiques. On a donc $\eta_1 \leq 0$ et $\eta_2 \geq 1$, ce qui permet d'écrire la relation (3.51) sous la forme suivante :

$$s = \alpha k(\eta - \eta_0) \qquad \text{pour } 0 \leq \eta \leq 1 \tag{3.93}$$

On peut alors expliciter les expressions de l'effort normal (3.91) et du moment fléchissant (3.92) :

$$n\Sigma = \tau \int_0^{1-\zeta} \alpha k(\eta - \eta_0) d\eta + \int_{1-\zeta}^{1} \alpha k(\eta - \eta_0) d\eta$$

$$\frac{m}{4\beta} = \tau \int_0^{1-\zeta} \alpha k(\eta - \eta_0)\eta d\eta + \int_{1-\zeta}^{1} \alpha k(\eta - \eta_0)\eta d\eta$$

La position de l'axe neutre obtenue en supposant une flexion simple ($n\Sigma=0$), est :

$$\eta_0 = \frac{\Gamma}{2\Sigma}$$

En introduisant cette valeur à l'expression de m, et en tenant compte de l'hypothèse sur la valeur de la déformation à rupture introduite, dans l'équation (3.43), on aboutit à la relation moment - courbure pendant cette phase élastique :

$$m = \frac{\beta p}{3\Sigma}\left[4\Sigma\Pi - 3\Gamma^2\right] \times k \tag{3.95}$$

Il s'agit à présent de trouver les valeurs des différents paramètres adimensionnels correspondant à la fin de cette phase qui est atteinte lorsque la fibre supérieure se plastifie, ce qui correspond à $\eta = \eta_1 = 0$. C'est-à-dire lorsque la contrainte s donnée par l'équation (3.93) atteint la limite élastique en compression $-s_0$. On trouve alors les valeurs finales données dans ce qui suit.

$$\begin{cases} \eta_{0_{01}} = \dfrac{\Gamma}{2\Sigma} \qquad \eta_{1_{01}} = 0 \qquad \eta_{2_{01}} = \dfrac{\Gamma}{\Sigma} \\ k_{01} = \dfrac{2\Sigma}{p\Gamma} \qquad m_{01} = \beta \dfrac{2\left(4\Sigma\Pi - 3\Gamma^2\right)}{3\Gamma} \end{cases} \tag{3.96}$$

3.5.4.2 Plastification de la fibre supérieure.

Dans cette phase, la section comporte une zone plastifiée localisée dans sa partie supérieure et une zone élastique. On a donc $\eta_1 \geq 0$ et $\eta_2 > 1$, ce qui permet d'écrire la relation (3.51) sous la forme suivante :

$$s = \begin{cases} -s_0 & \text{pour } 0 \leq \eta_1 \\ \alpha k\left(\eta - \eta_0\right) & \text{pour } \eta_1 \leq \eta \leq 1 \end{cases} \tag{3.97}$$

On peut alors expliciter l'expression (3.91) de l'effort normal :

$$n\Sigma = \tau \int_0^{\eta_1} -s_0 d\eta + \tau \int_{\eta_1}^{1-\zeta} \alpha k\left(\eta - \eta_0\right) d\eta + \int_{1-\zeta}^{1} \alpha k\left(\eta - \eta_0\right) d\eta$$

L'expression (3.92) du moment fléchissant s'écrit à son tour :

$$\frac{m}{4\beta} = \tau \int_0^{\eta_1} -s_0 \eta d\eta + \tau \int_{\eta_1}^{1-\zeta} \alpha k\left(\eta - \eta_0\right) \eta d\eta + \int_{1-\zeta}^{1} \alpha k\left(\eta - \eta_0\right) \eta d\eta$$

Après calculs et formulation des différentes hypothèses, on obtient le système d'équations paramétriques suivant, qui résume cette phase de plastification en partie supérieure.

$$\begin{cases} \eta_0 = \dfrac{\tau\eta_1^2 + \Gamma}{2\Sigma} \\ \eta_2 = \dfrac{\tau\eta_1^2 - \Sigma\eta_1 + \Gamma}{\Sigma} \\ k = \dfrac{2\Sigma}{p\left(\tau\eta_1^2 - 2\Sigma\eta_1 + \Gamma\right)} \\ \dfrac{m}{\beta} = \dfrac{2\left[2\Sigma\tau\eta_1^3 - 3\tau\Gamma\eta_1^2 + \left(4\Pi\Sigma - 3\Gamma^2\right)\right]}{3\left(\tau\eta_1^2 - 2\Sigma\eta_1 + \Gamma\right)} \end{cases} \tag{3.98}$$

Les valeurs finales de cette phase sont atteintes lorsque la fibre inférieure entre en plastification alors que l'âme n'est pas totalement plastique. C'est à dire lorsque η_2 atteint la valeur de 1. Ceci se traduit par les conditions suivantes :

$$\begin{cases} \eta_{1_{12}} < 1-\zeta \\ \eta_{2_{12}} = 1 \end{cases}$$

Les valeurs caractéristiques correspondant à la fin de cette première phase plastique sont :

$$\begin{cases} \eta_{0_{12}} = \dfrac{2\tau + \Sigma - \sqrt{\Delta}}{4\tau} \\ \eta_{1_{12}} = \dfrac{\Sigma - \sqrt{\Delta}}{2\tau} \\ \eta_{2_{12}} = 1 \\ k_{12} = \dfrac{\Sigma - 2\tau + \sqrt{\Delta}}{p\zeta^2(1-\tau)} \\ m_{12} = \beta \dfrac{2\left[2\Sigma^2 + \zeta(\tau - 3\Sigma) + \sqrt{\Delta}(2\zeta - \Sigma)\right]}{3\Gamma} \end{cases} \tag{3.99}$$

Où,

$$\Delta = \Sigma^2 - 4\tau(\Gamma - \Sigma)$$

3.5.4.3 Plastification de la fibre inférieure.

Les fibres inférieures et supérieures sont plastifiées. On a alors $0<\eta_1<1-\zeta$ et $1-\zeta\leq\eta_2\leq 1$. L'expression de la contrainte adimensionnelle est donnée par la relation (3.51), ce qui permet d'expliciter l'expression (3.91) de l'effort normal :

$$n\Sigma = \tau\int_0^{\eta_1} -s_0 d\eta + \tau\int_{\eta_1}^{1-\zeta} \alpha k(\eta - \eta_0) d\eta + \int_{1-\zeta}^{\eta} \alpha k(\eta - \eta_0) d\eta + \int_{\eta_2}^{1} s_0 d\eta$$

L'expression (3.92) du moment fléchissant s'écrit à son tour :

$$\frac{m}{4\beta} = \tau\int_0^{\eta_1} -s_0 \eta d\eta + \tau\int_{\eta_1}^{1-\zeta} \alpha k(\eta - \eta_0)\eta d\eta + \int_{1-\zeta}^{\eta_2} \alpha k(\eta - \eta_0)\eta d\eta + \int_{\eta_2}^{1} s_0 \eta d\eta$$

Après calculs et formulation des différentes hypothèses, on obtient le système d'équations paramétriques suivant, qui résume cette phase de plastification en partie inférieure.

$$\begin{cases} \eta_0 = \dfrac{2-\Sigma+2\eta_1+\sqrt{r^2-4s}}{4} \\ \eta_2 = \dfrac{2-\Sigma+\sqrt{r^2-4s}}{2} \\ k = \dfrac{4}{p\left(r+\sqrt{r^2-4s}\right)} \\ \dfrac{m}{\beta} = \dfrac{2}{3}\left[-2\tau\eta_1^2+4(\Sigma-\zeta)\eta_1+\Sigma(1+3\zeta-\Sigma)-\zeta+\sqrt{r^2-4s}(\Sigma-2\zeta)\right] \end{cases} \tag{3.100}$$

Où ;

$$\begin{cases} r = 1+(1-\tau)(1-\zeta)-2\eta_1 \\ s = (1-\tau)\left[(1-\zeta)-\eta_1\right]^2 \end{cases}$$

Nous avons établi une formulation mathématique de l'appartenance à l'une ou à l'autre des classes définies au paragraphe 5.2, elle peut s'écrire comme suit :

- La section étant de classe A, elle sera dite :
  - ∗ De sous classe A1 si :

$$\tau \le \frac{(p+1)\zeta}{(p-1)(1-\zeta)} \tag{3.101}$$

  - ∗ De sous classe A2 si non.

- La section étant de classe C, elle sera dite :
  - ∗ De sous classe C1 si :

$$\tau \ge \frac{(p+1)\zeta-2}{(p-1)(1-\zeta)} \tag{3.102}$$

  - ∗ De sous classe C2 si non.

a Sous classes A1, B et C1 : valeurs ultimes.

Les valeurs ultimes du moment fléchissant et de la courbure, pour les sections considérées, sont effectivement atteintes lors de cette phase, c'est à dire lorsque les fibres supérieure et inférieure sont plastifiées. Elles s'écrivent sous la forme :

$$\begin{cases} \eta_{0_{\upsilon\lambda\tau}} = p\dfrac{p(1-\Sigma)+1+\sqrt{\Lambda}}{(p+1)^2-\tau(p-1)^2} \\ \eta_{1_{\upsilon\lambda\tau}} = (p-1)\dfrac{p(1-\Sigma)+1+\sqrt{\Lambda}}{(p+1)^2-\tau(p-1)^2} \\ \eta_{2_{\upsilon\lambda\tau}} = (p+1)\dfrac{p(1-\Sigma)+1+\sqrt{\Lambda}}{(p+1)^2-\tau(p-1)^2} \\ k_{ult} = \dfrac{(p+1)^2-\tau(p-1)^2}{p\left[p(1-\Sigma)+1+\sqrt{\Lambda}\right]} \\ \dfrac{m_{ult}}{\beta} = \dfrac{2}{3}\left[-2\tau\eta_{1\,ult}^2+4(\Sigma-\zeta)\eta_{1_{ult}}+\Sigma(1+3\zeta-\Sigma)-\zeta+\sqrt{r_{ult}^2-4s_{ult}}(\Sigma-2\zeta)\right] \end{cases} \tag{3.103}$$

Où ;

$$\begin{cases} r_{ult} = 1+(1-\tau)(1-\zeta)-2\eta_{1_{ult}} \\ s_{ult} = (1-\tau)\left[(1-\zeta)-\eta_{1_{ult}}\right]^2 \\ \Lambda = \left[p(1-\Sigma)+1\right]^2-(1-\zeta)(1-\Sigma)\left[(p+1)^2-\tau(p-1)^2\right] \end{cases}$$

b Sous classe A2 : valeurs finales.

Les valeurs ultimes sont, pour ce type de sections, atteintes lorsque la semelle est totalement plastifiée, les valeurs finales, correspondant à la plastification de la fibre supérieure de la semelle, s'écrivent :

$$\begin{cases} \eta_{0_{23}} = \dfrac{\Sigma}{2\tau} \qquad \eta_{1_{23}} = \dfrac{\zeta}{\tau} \qquad \eta_{2_{23}} = 1-\zeta \\ k_{23} = \dfrac{2\tau}{p(\Sigma-2\zeta)} \qquad \text{et} \quad m_{23} = \dfrac{2}{3}\beta\left[-2\dfrac{\zeta^2}{\tau}+\Sigma(1-\zeta)+3\zeta\right] \end{cases} \tag{3.104}$$

c Sous classe C2 : valeurs finales.

Les valeurs ultimes sont, pour ce type de sections, atteintes lorsque l'âme est totalement plastifiée, les valeurs finales, correspondant à la plastification de la fibre inférieure de l'âme, s'écrivent :

$$\begin{cases} \eta_{0_{23}} = 1 - \dfrac{\Sigma}{2} \qquad \eta_{1_{23}} = 1 - \zeta \qquad \eta_{2_{23}} = 1 - \tau(1-\zeta) \\ k_{23} = \dfrac{2}{p(2\zeta - \Sigma)} \qquad \text{et} \qquad m_{23} = \dfrac{2}{3}\beta\left[\Sigma(3 + 5\zeta - 2\Sigma) - \zeta(3 + 2\zeta)\right] \end{cases} \tag{3.105}$$

3.5.4.4 Phase plastique n : 3.

a Sous classe A2 : semelle totalement plastifiée. .

La semelle étant totalement plastifiée. On a alors $0<\eta_1\leq 1-\zeta$ et $\eta_2\leq 1-\zeta$. L'expression de la contrainte adimensionnelle est donnée par la relation (3.51), ce qui permet d'expliciter l'expression (3.91) de l'effort normal :

$$n\Sigma = \tau\int_0^{\eta_1} -s_0 d\eta + \tau\int_{\eta_1}^{\eta_2} \alpha k(\eta - \eta_0) d\eta + \tau\int_{\eta_2}^{1-\zeta} s_0 d\eta + \int_{1-\zeta}^{1} s_0 d\eta$$

L'expression (3.92) du moment fléchissant s'écrit à son tour :

$$\frac{m}{4\beta} = \tau\int_0^{\eta_1} -s_0 \eta d\eta + \tau\int_{\eta_1}^{\eta_2} \alpha k(\eta - \eta_0)\eta d\eta + \tau\int_{\eta_2}^{1-\zeta} s_0 \eta d\eta + \int_{1-\zeta}^{1} s_0 \eta d\eta$$

Après calculs et formulation des différentes hypothèses, on obtient le système d'équations paramétriques suivant, qui résume cette phase de plastification totale de la semelle.

$$\begin{cases} \eta_0 = \dfrac{\Sigma}{2\tau} \\ \eta_2 = \dfrac{\Sigma - \tau\eta_1}{\tau} \\ k = \dfrac{2\tau}{p(\Sigma - 2\tau\eta_1)} \\ \dfrac{m}{\beta} = \dfrac{2}{3}\left[-2\tau^2\eta_1^2 + 2\tau\Sigma\eta_1 + \Sigma(\Sigma - 3\zeta) + 3\tau\zeta\right] \end{cases} \tag{3.106}$$

Les valeurs ultimes seront atteintes lorsque la déformation maximale sera égale à ε_p. Ces valeurs s'écrivent sous la forme suivante :

$$\begin{cases} \eta_{0_{ult}} = \dfrac{\Sigma}{2\tau} \\ \eta_{1_{\upsilon\lambda\tau}} = \dfrac{(p-1)\Sigma}{2\tau p} \\ \eta_{2_{\upsilon\lambda\tau}} = \dfrac{(p+1)\Sigma}{2\tau p} \\ k_{ult} = \dfrac{2\tau}{\Sigma} \\ m_{ult} = 1 - \dfrac{\beta\Sigma^2}{3\tau p^2} \end{cases} \tag{3.107}$$

b Sous classe C2 : âme totalement plastifiée.

L'âme étant totalement plastifiée, on a alors $\eta_1 \geq 1-\zeta$ et $\eta_2 \geq 1-\zeta$. L'expression de la contrainte adimensionnelle est donnée par la relation (3.51), ce qui permet d'expliciter l'expression (3.91) de l'effort normal :

$$n\Sigma = \tau \int_0^{1-\zeta} -s_0 d\eta + \int_{1-\zeta_1}^{\eta_1} -s_0 d\eta + \int_{\eta_1}^{\eta_2} \alpha k(\eta - \eta_0) d\eta + \int_{\eta_2}^{1} s_0 d\eta$$

L'expression (3.93) du moment fléchissant s'écrit à son tour :

$$\frac{m}{4\beta} = \tau \int_0^{1-\zeta} -s_0 \eta d\eta + \int_{1-\zeta}^{\eta_1} -s_0 \eta d\eta + \int_{\eta_1}^{\eta_2} \alpha k(\eta - \eta_0)\eta d\eta + \int_{\eta_2}^{1} s_0 \eta d\eta$$

Après calculs et formulation des différentes hypothèses, on obtient le système d'équations paramétriques suivant, qui résume cette phase de plastification totale de l'âme.

$$\begin{cases} \eta_0 = 1 - \dfrac{\Sigma}{2} \\ \eta_2 = 2 - \Sigma - \eta_1 \\ k = \dfrac{2}{p(2 - \Sigma - 2\eta_1)} \\ \dfrac{m}{\beta} = \dfrac{2}{3}\left[-2\eta_1^2 + 2(2-\Sigma)\eta_1 + \Sigma(5 + 3\zeta - 2\Sigma) - (2 + 3\zeta)\right] \end{cases} \tag{3.108}$$

Les valeurs ultimes seront atteintes lorsque la déformation maximale sera égale à sa valeur à rupture, ε_p. Ces valeurs s'écrivent sous la forme suivante :

$$\begin{cases} \eta_{0ult} = 1 - \dfrac{\Sigma}{2} \\ \eta_{1ult} = \dfrac{(p-1)(2-\Sigma)}{2p} \\ \eta_{2ult} = \dfrac{(p+1)(2-\Sigma)}{2p} \\ k_{ult} = \dfrac{2}{2-\Sigma} \\ m_{ult} = 1 - \dfrac{\beta(2-\Sigma)^2}{3p^2} \end{cases} \tag{3.109}$$

Contrairement aux sections triangulaires et pour un paramètre de ductilité p fixé, la relation moment - courbure adimensionnelle des sections en T n'est pas unique. Elle dépend des épaisseurs adimensionnelles τ et ζ de l'âme et de la table respectivement. La figure (3.11) représente la relation moment - courbure pour des valeurs de τ, ζ et p données. Elle couvre tous les cas de figures, c'est à dire toutes les classes possibles de sections en T (A1, A2, B, C1 et C2). On observe alors que l'ensemble de ces courbes dessine un fuseau relativement épais.

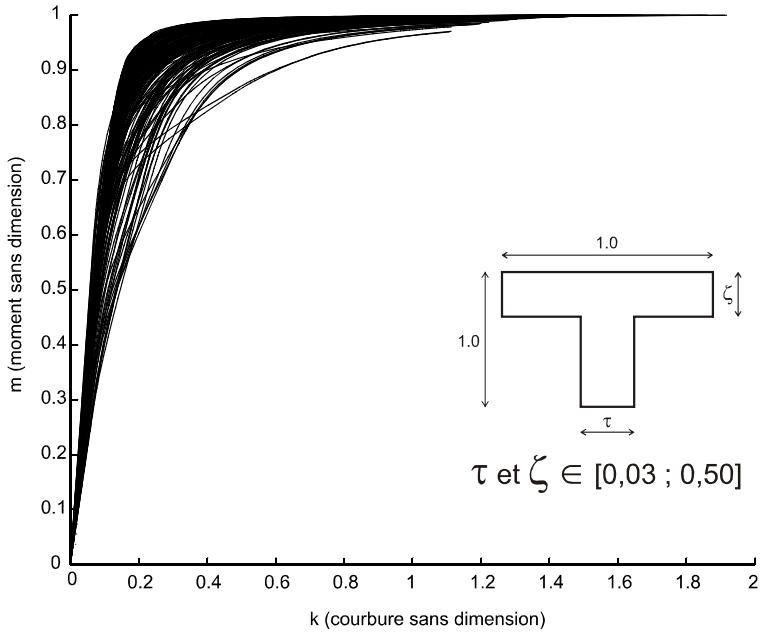


Figure 3.11 : Courbes moment – courbure de quelques sections représentatives.

3.6 Conclusion.

Dans ce chapitre nous avons formalisé le problème à résoudre. Nous avons commencé par présenter les hypothèses simplificatrices de la théorie de flexion simple des poutres et nous avons posé les équations qui régissent leur comportement. Il s'agit des relations traduisant l'équilibre, des équations de compatibilité et des équations constitutives. Ces dernières prennent la forme de relations moment - courbure. Elles dépendent de la géométrie de la section et de la loi de comportement du matériau considéré. Nous avons supposé un comportement du matériau élasto-plastique parfait et nous avons exposé les lois constitutives des poutres à sections triangulaires et en T.

Dans le chapitre suivant nous allons exposer la méthode numérique retenue pour résoudre les équations différentielles établies plus haut. Parmi les méthodes numériques on peut utiliser soit la méthode des différences finies, soit la méthode des éléments finis. Cette dernière, qui reste la méthode la plus puissante et la plus facile à généraliser, est alors choisie. Ce choix est fait parce que, d'un côté cette méthode reste plus efficace que la méthode des différences finies surtout lorsque la géométrie de la structure à étudier devient complexe. D'un autre côté, sa grande capacité à s'adapter aux différents problèmes n'est plus à démontrer.

Chapitre 4

Présentation du code de calcul éléments finis utilisant les RNA

4.1 Introduction

L'objectif de ce travail est de prouver que les réseaux de neurones artificiels peuvent être utilisés efficacement dans un code de calcul en éléments finis en tant qu'outil d'interpolation des lois de comportement. Cette recherche est inspirée d'un travail effectué à l'université de Clermont-Ferrand II en 1996 [4] où son auteur a essayé d'introduire, sans grand succès, les réseaux de neurones dans un code de calcul en différences finies pour le même objectif.

Le choix de la méthode des éléments finis est d'abord fait dans un souci d'élaborer le travail autrement, mais surtout parce que la MEF reste plus efficace que la méthode des différences finies lorsque la géométrie de la structure à étudier devient complexe. D'un autre côté, sa grande capacité à s'adapter aux différents problèmes n'est plus à démontrer [2, 7, 22].

Le type d'élément fini retenu pour cette étude est un élément poutre en flexion simple. Ce choix vient du fait que, pour les éléments finis de flexion, la loi de comportement à introduire dans le code de calcul préconisé prend la forme d'une relation moment - courbure (M-χ), c'est-à-dire qu'il s'agit d'une relation résultant de l'intégration de la courbe contrainte - déformation (σ-ε). Les expressions (M-χ) théoriques peuvent alors être relativement compliquées même dans le cas de relations (σ-ε) très simples (voir chapitre 3). Leur insertion dans un programme numérique peut s'avérer laborieuse et leur utilisation fastidieuse d'où, l'idée de les utiliser comme exemple à notre objectif en les faisant apprendre à des réseaux de neurones artificiels, réputés pour leur grande capacité d'apprentissage.

Parmi les théories de flexion des poutres, nous avons utilisé celle de Timoshenko. Elle présente l'avantage de prendre en considération la distorsion et permet, par conséquent, d'étudier aussi bien les poutres élancées que les poutres épaisses. Nous ferons néanmoins une approximation en utilisant les lois constitutives qui, développées au chapitre précédent, négligent toute interaction entre les différents efforts internes, ce qui est bien entendu inexacte mais l'expérience a montré que cet effet n'est pas important, spécialement dans le cas des poutres minces.

Pour montrer le travail réalisé, nous avons décidé de commencer ce chapitre par une présentation de la méthode des éléments finis appliquée aux poutres de Timoshenko en flexion simple [16]. La non linéarité matérielle y est prise en compte en utilisant soit, une relation moment - courbure simplifiée soit, en intégrant numériquement la relation contrainte - déformation sur la section de la poutre. Dans ce dernier cas, la poutre doit être divisée en couches (layers) ce qui demande un effort de modélisation supplémentaire, surtout si la forme de la section de la poutre n'est pas simple (triangulaire, en T, en I, …). On peut en arriver à perdre l'intérêt de la modélisation très simple des poutres qui n'exige, dans le plan, que deux points pour les définir entièrement. Le premier programme réalisé dans le cadre de ce magister a été écrit, en langage PASCAL, pour étudier la flexion simple des poutres de Timoshenko en élasto-plasticité par la MEF, en utilisant la relation (M-χ) simplifiée.

Nous présenterons ensuite comment implanter (insérer) les relations (M-χ) théoriques dans un code de calcul numérique. On peut soit, en faire l'évaluation et la résolution exacte soit, utiliser les réseaux de neurones. Concernant ce dernier cas, nous commencerons par présenter la stratégie adoptée dans la référence [4] et qui n'a pas donné satisfaction en fournissant des résultats très dispersés. Nous présenterons ensuite la solution que nous proposons pour utiliser les réseaux de neurones artificiels comme outils d'interpolation de la loi de comportement.

La dernière partie de ce chapitre explique la mise en œuvre pratique de la solution proposée, de la recherche des réseaux de neurones adéquats, à la programmation de la stratégie adoptée et son insertion dans le code de calcul en éléments finis adopté.

4.2 La MEF pour étudier la flexion simple de la poutre de Timoshenko en élasto-plasticité.

4.2.1 Introduction

Dans le chapitre précédent nous avons établi les relations qui régissent la théorie de la flexion des poutres. Elle se présente sous forme d'équations différentielles qu'on résout soit en utilisant la méthode des différences finies, soit la méthode des éléments finis. Le choix de la méthode des éléments finis est fait parce qu'elle reste plus efficace que la méthode des différences finies lorsque la géométrie de la structure à étudier devient complexe.

Nous rappelons que c'est la théorie de Timoshenko qui a été retenue. Nous allons présenter la MEF appliquée aux poutres de Timoshenko en flexion simple en élasticité. Nous exposons ensuite la modélisation en éléments finis et son extension pour une analyse élasto-plastique. Nous terminons ce paragraphe par une présentation de l'algorithme du premier programme réalisé dans le cadre de ce magister. Il permet d'étudier la flexion simple des poutres de Timoshenko en élasto-plasticité par la MEF, en utilisant une relation moment - courbure simplifiée.

4.2.2 La MEF pour étudier la poutre de Timoshenko en flexion simple en élasticité.

Pour représenter le champ de déplacement et de déformation, la méthode des éléments finis utilise des fonctions d'interpolation cinématiquement admissibles dont les amplitudes

inconnues sont appelées degrés de liberté. Le principe des travaux virtuels est ensuite utilisé pour écrire l'équilibre et dériver les équations matricielles régissant le comportement [10].

4.2.2.1 Principe des travaux virtuels.

Ce principe peut être énoncé comme suit : Un corps déformable soumis aux forces de volume $\{f\}$ et développant un état de contraintes $\{\sigma\}$est en équilibre si et seulement si quelque soit le champs de déplacement virtuel $\langle \delta u \rangle$ cinématiquement admissible on a :

$$\iiint_V \langle \delta\varepsilon \rangle \{\sigma\} \cdot dV - \iiint_V \langle \delta u \rangle \cdot \{f\} dV = 0 \tag{4.1}$$

Où $\langle \delta\varepsilon \rangle$ est le tenseur des déformations virtuelles résultant de champs de déplacement virtuel.

L'équation ci-dessus indique que la somme des travaux virtuels doit être nulle pour avoir l'équilibre. En se mettant dans les conditions de la théorie de la flexion simple des poutres planes, c'est-à-dire en utilisant les équations (3.1) (3.2) (3.9) (3.16) (3.21) et si l'axe des *x* est confondu avec la ligne moyenne ($y_0 = 0$), ce principe s'écrit :

$$\iiint_V \left\langle -y\frac{d(\delta\theta)}{dx} \quad \varepsilon_y \quad \delta\beta \right\rangle \cdot \begin{Bmatrix} \sigma_x \\ 0 \\ \tau_{xy} \end{Bmatrix} \cdot dV - \iiint_V \langle \delta u \quad \delta v \rangle \cdot \begin{Bmatrix} 0 \\ f_y \end{Bmatrix} dV = 0 \tag{4.2}$$

En notant la charge par unité de longueur $p_y = \iint_S f_y \cdot dS$ et en utilisant les définitions du moment de flexion et de l'effort tranchant données par les relations (3.4) et (3.5), le principe des travaux virtuels devient :

$$\int_0^l \left(-\frac{d(\delta\theta)}{dx} \cdot M + \delta\beta \cdot T \right) \cdot dx - \int_0^l \delta v \cdot p_y \cdot dx = 0 \tag{4.3}$$

En utilisant les équations constitutives élastiques (3.19) et (3.25), on obtient :

$$\int_0^l \left(\frac{d(\delta\theta)}{dx} \cdot EI \cdot \frac{d(\theta)}{dx} + \delta\beta \cdot G\hat{S} \cdot \beta \right) \cdot dx - \int_0^l \delta v \cdot p_y \cdot dx = 0 \tag{4.4}$$

Où EI est la rigidité flexionnelle et $G\hat{S}$ la raideur de cisaillement. $\hat{S}$ est la surface réduite et a remplacé S pour tenir compte de la distribution non uniforme des contraintes de cisaillement.

4.2.2.2 Représentation des déplacements et des déformations.

L'élément fini de Hughes [16] utilisé dans ce mémoire a deux nœuds, 1 et 2, et deux degrés de liberté par nœud, $v_i^{(e)}$ et $\theta_i^{(e)}$ (voir figure 4.1). La flèche $v^{(e)}$ en un point quelconque de l'élément (e) est donnée par l'interpolation des déplacements nodaux :

$$v^{(e)} = N_1^{(e)} v_1^{(e)} + N_2^{(e)} v_2^{(e)} \tag{4.5}$$

De la même manière, la rotation $\theta^{(e)}$ est donnée par l'interpolation des rotations nodales :

$$\theta^{(e)} = N_1^{(e)} \theta_1^{(e)} + N_2^{(e)} \theta_2^{(e)} \tag{4.6}$$

Les fonctions d'interpolation N_1 et N_2 sont données par :

$$N_1 = \frac{1}{l^{(e)}} \left(x_2^{(e)} - x^{(e)} \right)$$

$$N_2 = \frac{1}{l^{(e)}} \left(x^{(e)} - x_1^{(e)} \right)$$

Où x_1 et x_2 sont les coordonnées des nœuds 1 et 2, $l^{(e)}$ est la longueur de l'élément et x est la coordonnée d'un point quelconque de l'élément *(e)*.

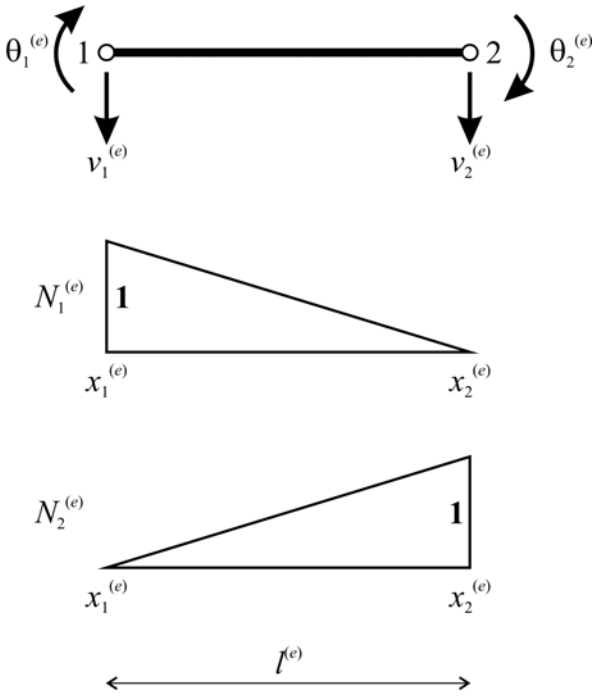


Figure 4.1 : L'élément fini poutre et ses fonctions d'interpolation.

D'après l'équation (3.22), la courbure ou déformation flexionnelle $\varepsilon_f^{(e)}$ s'écrit en fonction des degrés de liberté sous la forme :

$$-\left(\frac{d\theta}{dx}\right)^{(e)} = -\left(\frac{dN_1}{dx}\right)^{(e)} \theta_1^{(e)} - \left(\frac{dN_2}{dx}\right)^{(e)} \theta_2^{(e)}$$

$$\varepsilon_f^{(e)} = \begin{bmatrix} 0 & \frac{1}{l^{(e)}} & 0 & -\frac{1}{l^{(e)}} \end{bmatrix} \times \begin{bmatrix} v_1^{(e)} \\ \theta_1^{(e)} \\ v_2^{(e)} \\ \theta_2^{(e)} \end{bmatrix} = B_f^{(e)} \varphi^{(e)} \tag{4.7}$$

$\varphi^{(e)}$ est le vecteur des degrés de liberté de l'élément.

La relation (3.12) permet de trouver la distorsion β qu'on note $\varepsilon_s^{(e)}$ pour harmoniser la notation et indiquer qu'il s'agit de la déformation de cisaillement.

$$\left(\frac{dv}{dx} - \theta\right)^{(e)} = \left(\frac{dN_1}{dx}\right)^{(e)} v_1^{(e)} - N_1^{(e)} \theta_1^{(e)} + \left(\frac{dN_2}{dx}\right)^{(e)} v_2^{(e)} - N_2^{(e)} \theta_2^{(e)}$$

$$\varepsilon_s^{(e)} = \begin{bmatrix} -\frac{1}{l^{(e)}}, & -\frac{\left(x_2^{(e)} - x^{(e)}\right)}{l^{(e)}}, & \frac{1}{l^{(e)}}, & -\frac{\left(x^{(e)} - x_1^{(e)}\right)}{l^{(e)}} \end{bmatrix} \times \begin{bmatrix} w_1^{(e)} \\ \theta_1^{(e)} \\ w_2^{(e)} \\ \theta_2^{(e)} \end{bmatrix} = B_s^{(e)} \varphi^{(e)} \tag{4.8}$$

4.2.2.3 La matrice de rigidité et le vecteur force élémentaires.

En utilisant la représentation des déplacements et des déformations présentée ci-dessus ainsi que le principe des travaux virtuels donné par l'équation (4.3), on arrive à montrer que le système d'équations d'équilibre de l'élément poutre s'écrit :

$$\mathrm{p}^{(e)} - \mathrm{f}^{(e)} = 0 \tag{4.9}$$

Ce qui traduit l'équilibre entre le vecteur force $\mathrm{p}^{(e)}$ apporté par les efforts internes et le vecteur $\mathrm{f}^{(e)}$, contribution des forces extérieures :

$$\mathrm{p}^{(e)} = \int_{x_1^{(e)}}^{x_2^{(e)}} \left[B_f^{(e)}\right]^T M^{(e)} dx + \int_{x_1^{(e)}}^{x_2^{(e)}} \left[B_s^{(e)}\right]^T T^{(e)} dx \tag{4.10}$$

$$\mathrm{f}^{(e)} = \int_{x_1^{(e)}}^{x_2^{(e)}} \left[N_1^{(e)}, \quad 0, \quad N_2^{(e)}, \quad 0 \right]^T p_y \, dx \tag{4.11}$$

En élasticité linéaire et grâce aux équations constitutives (3.19) et (3.25), l'équation (4.9) devient :

$$\left[K_f^{(e)} + K_s^{(e)} \right] \times \varphi^{(e)} - \mathrm{f}^{(e)} = 0 \tag{4.12}$$

Où la matrice de rigidité élémentaire comporte deux contributions : une flexionnelle et l'autre due au cisaillement.

$$K_f^{(e)} = \int_{x_1^{(e)}}^{x_2^{(e)}} \left[B_f^{(e)} \right]^T \left(\mathrm{EI} \right)^{(e)} B_f^{(e)} dx \tag{4.13}$$

$$K_s^{(e)} = \int_{x_1^{(e)}}^{x_2^{(e)}} \left[B_s^{(e)} \right]^T \left(\mathrm{G\hat{S}} \right)^{(e)} B_s^{(e)} dx \tag{4.14}$$

La matrice de rigidité flexionnelle (4.13) peut être évaluée en utilisant une intégration à 1 point de Gauss, et prend la forme suivante :

$$K_f^{(e)} = \left(\frac{\mathrm{EI}}{l} \right)^{(e)} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \tag{4.14}$$

Si $K_s^{(e)}$ est évaluée exactement en utilisant 2 points de Gauss, on obtient :

$$K_s^{(e)} = \left(\frac{\mathrm{G\hat{S}}}{l} \right)^{(e)} \begin{bmatrix} 1 & \frac{l}{2} & -1 & \frac{l}{2} \\ \frac{l}{2} & \frac{l^2}{3} & -\frac{l}{2} & \frac{l^2}{6} \\ -1 & -\frac{l}{2} & 1 & -\frac{l}{2} \\ \frac{l}{2} & \frac{l^2}{6} & -\frac{l}{2} & \frac{l^2}{3} \end{bmatrix}^{(e)}$$

Seulement, il a été montré qu'avec cette formulation de $K_s^{(e)}$, on obtient une rigidité du cisaillement surestimée. Ce phénomène est connu sous le nom de verrouillage en cisaillement (Shear Locking). Pour y remédier on intègre $K_s^{(e)}$ avec 1 point de Gauss. On obtient alors la matrice de rigidité formulée ci-après et qui donne d'excellents résultats.

$$K_s^{(e)} = \left(\frac{\mathrm{G\hat{S}}}{l}\right)^{(e)} \begin{bmatrix} 1 & \frac{l}{2} & -1 & \frac{l}{2} \\ \frac{l}{2} & \frac{l^2}{4} & -\frac{l}{2} & \frac{l^2}{4} \\ -1 & -\frac{l}{2} & 1 & -\frac{l}{2} \\ \frac{l}{2} & \frac{l^2}{4} & -\frac{l}{2} & \frac{l^2}{4} \end{bmatrix}^{(e)} \tag{4.15}$$

Le vecteur des forces nodales obtenu par évaluation de l'intégrale figurant dans l'équation (4.11) est donné par :

$$\mathrm{f}^{(e)} = \left[\frac{(p_y l)^{(e)}}{2}, \quad 0, \quad \frac{(p_y l)^{(e)}}{2}, \quad 0\right] \tag{4.16}$$

4.2.2.4 Les efforts internes.

Le moment fléchissant de l'élément s'obtient à partir de la relation moment - courbure (3.25) et de l'équation (4.7). En élasticité on a :

$$M^{(e)} = (\mathrm{EI})^{(e)} B_f{}^{(e)} \varphi^{(e)} = (\mathrm{EI})^{(e)} \left[0, \quad \frac{1}{l^{(e)}}, \quad 0, \quad -\frac{1}{l^{(e)}}\right] \begin{bmatrix} v_1{}^{(e)} \\ \theta_1{}^{(e)} \\ v_2{}^{(e)} \\ \theta_2{}^{(e)} \end{bmatrix}$$

$$M^{(e)} = \left(\frac{\mathrm{EI}}{l}\right)^{(e)} \left(\theta_1{}^{(e)} - \theta_2{}^{(e)}\right) \tag{4.17}$$

On constate alors que le moment fléchissant garde une valeur constante le long de l'élément de Hughes. D'après la loi de Hook (3.19) et la relation (4.8), l'effort tranchant varie quant à lui linéairement le long de chaque élément. Pour rester cohérent avec la remarque précédente, on évalue une valeur de l'effort tranchant, celle calculée au milieu de l'élément :

$$T^{(e)} = (\mathrm{G\hat{S}})^{(e)} B_s{}^{(e)} \varphi^{(e)} = (\mathrm{G\hat{S}})^{(e)} \left[-\frac{1}{l^{(e)}}, \quad -\frac{1}{2}, \quad \frac{1}{l^{(e)}}, \quad -\frac{1}{2}\right] \begin{bmatrix} v_1{}^{(e)} \\ \theta_1{}^{(e)} \\ v_2{}^{(e)} \\ \theta_2{}^{(e)} \end{bmatrix}$$

$$T^{(e)} = (\mathrm{G\hat{S}})^{(e)} \left\{\left(\frac{v_2{}^{(e)} - v_1{}^{(e)}}{l^{(e)}}\right) - \left(\frac{\theta_1{}^{(e)} + \theta_2{}^{(e)}}{2}\right)\right\} \tag{4.18}$$

4.2.3 La poutre de Timoshenko en élasto-plasticité.

4.2.3.1 La relation moment - courbure non linéaire simplifiée.

La façon la plus simple de considérer la flexion élasto-plastique est de supposer qu'initialement, la réponse de la poutre est élastique et la section devient instantanément et entièrement plastique dès que le moment fléchissant atteint, en valeur absolue, la valeur du moment plastique définie par l'équation (3.35). Cette supposition implique une approximation grossière qui est expliquée ci-dessous.

Comme le montre la figure (4.2) dans le cas d'une section symétrique, la section est supposée totalement plastifiée quand le moment fléchissant de la situation (c) devient égal au moment plastique défini par la distribution des contraintes de la situation (d). Auquel cas, les contraintes aux fibres extrêmes dépassent, en valeur absolue, la contrainte plastique σ_0 du matériau alors qu'en principe, la plastification du matériau commence à partir de la situation (b).

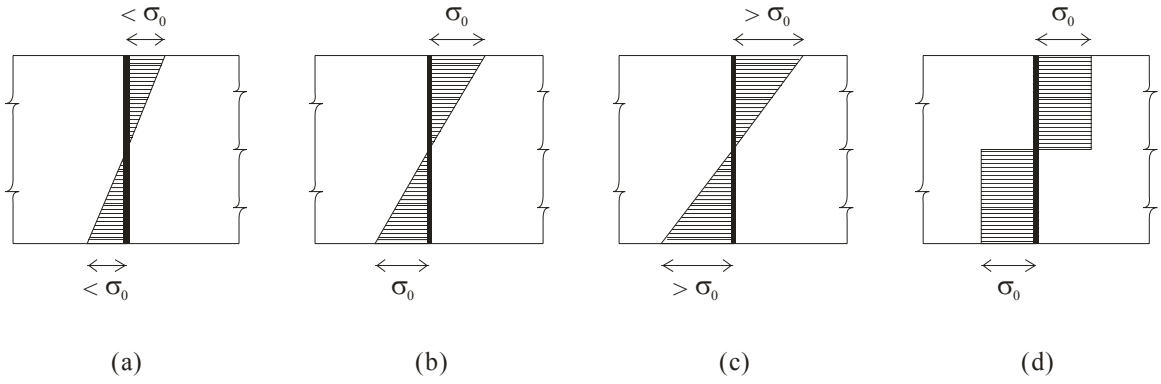


Figure 4.2 : Plastification d'une section de poutre [16].

La relation moment - courbure d'une poutre en élasto-plasticité est alors supposée avoir la forme bilinéaire montrée par la figure (4.3). Initialement la poutre se déforme élastiquement avec une raideur flexionnelle EI jusqu'à ce que le moment plastique M_p soit atteint et la section se plastifie totalement. Sous des charges plus grandes, le matériau est supposé présenter un écrouissage linéaire caractérisé par une raideur flexionnelle tangentielle $(EI)_T$.

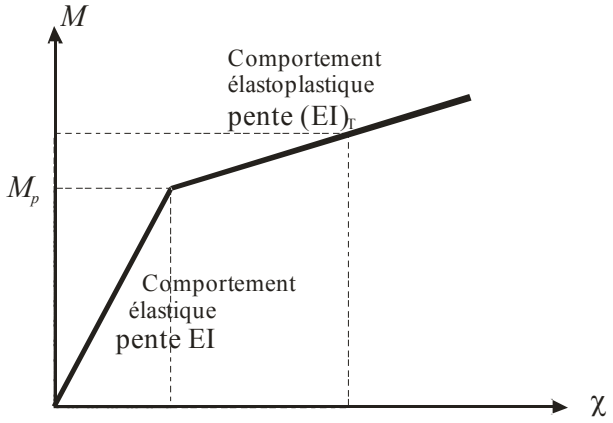


Figure 4.3 : Relation moment - courbure élasto-plastique simplifiée.

4.2.3.2 Solution des équations non linéaires.

Le principe des travaux virtuels appliqué à une structure formée d'un ensemble d'éléments permet d'aboutir au système d'équations d'équilibre de la structure qui est sous la forme :

$$\mathrm{p} - \mathrm{f} = 0 \tag{4.19}$$

Les vecteurs p et f s'obtiennent en assemblant, de la manière usuelle, les contributions, $\mathrm{p}^{(e)}$ et $\mathrm{f}^{(e)}$, de tous les éléments formant la structure. Les équations (4.7), (4.8) et (4.10) permettent d'exprimer la contribution $\mathrm{p}^{(e)}$ dans le cas de l'élément de Hughes comme suit :

$$\mathrm{p}^{(e)} = \left[-T^{(e)}, \quad M^{(e)} - \frac{(Tl)^{(e)}}{2}, \quad T^{(e)}, \quad -M^{(e)} - \frac{(Tl)^{(e)}}{2} \right]^T \tag{4.20}$$

Dans les problèmes élasto-plastiques, le moment $M^{(e)}$ est une fonction non linéaire et en général, on ne peut déterminer $M^{(e)}$, et par suite $\mathrm{p}^{(e)}$, qu'approximativement. L'équation (4.19) est donc non linéaire et puisqu'on ne connaît qu'une approximation du vecteur p alors, la différence $\mathrm{p} - \mathrm{f}$ est égale à un vecteur de forces résiduelles ψ qu'on essayera de faire tendre vers zéro par un algorithme itératif. Le chargement est alors appliqué à la structure progressivement. Pour chaque incrément de charge $\Delta\mathrm{f}$, on opère comme suit [16] :

1. On actualise le vecteur f dû aux forces extérieures, on initialise le compteur r des itérations et on décale le vecteur des forces résiduelles de l'incrément de charge :

$$\mathrm{f} = \mathrm{f} + \Delta\mathrm{f}\ ,\ r = 0\ ,\ \psi^{(r)} = \Delta\mathrm{f} + \psi$$

   Où, ψ est le résidu qui a persisté après que l'algorithme itératif du précédent incrément ait convergé. On inclut ainsi la correction de l'équilibre des incréments précédents.
2. On évalue, si nécessaire, la nouvelle matrice de rigidité tangente de la structure, K_T.
3. On résout $K_T \cdot \Delta\varphi^{(r)} = \psi^{(r)}$, pour avoir l'incrément des degrés de liberté.
4. On actualise le vecteur des degrés de liberté : $\varphi = \varphi + \Delta\varphi^{(r)}$.
5. De l'incrément des degrés de liberté, on déduit, pour chaque élément *e*, l'incrément de déformation. A partir de l'état de contrainte de l'élément à l'itération précédente et de l'incrément de déformation, on teste si l'élément s'est plastifié ou non. On déduit ensuite une valeur actualisée de la contrainte. Pour les poutres en flexion simple, objet de la présente étude, ceci peut être schématisé par :

$$\left(M^{(r-1)}, \Delta\chi\right) \rightarrow M^{(r)} \tag{4.21}$$

$$\left(T^{(r-1)}, \Delta\beta\right) \rightarrow T^{(r)}$$

   Avec ces valeurs actualisées de l'état de contrainte, on évalue $\mathrm{p}^{(e)}$ à partir de l'équation (4.20) et on en déduit le vecteur des forces résiduelles de l'élément,

$\left[\psi^{(e)}\right]^{(r+1)} = p^{(e)} - f^{(e)}$, qu'on assemble pour obtenir le vecteur global des forces résiduelles $\psi^{(r+1)}$.

6. Tester la convergence des degrés de liberté.
7. Si la solution a convergé effectuer l'affectation $\psi = \psi^{(r+1)}$ et revenir à l'étape 1. Dans le cas contraire, incrémenter les compteur des itérations, $r = r + 1$, et aller à l'étape 2.

Sur la base de cet algorithme et de la formulation matricielle présentée ci-dessus, nous avons écrit un programme en langage PASCAL qui permet de traiter la flexion simple des poutres de Timoshenko dans le cas de la relation constitutive approchée présentée au paragraphe 4.2.3.1.

Les inconvénients d'une telle approche ont été présentés au paragraphe (4.2.3.1). Pour les éviter, nous avons trouvé dans la bibliographie [16] une autre modélisation en éléments finis de la poutre de Timoshenko. Elle est présentée dans le paragraphe suivant.

4.2.3.3 La poutre de Timoshenko divisée en couches (Layered beam).

Dans cette approche, la poutre est divisée en un nombre fini de couches (layers). On suppose qu'au fur et à mesure que le chargement augmente, les couches se plastifient entièrement dès que les contraintes normales en leurs centres atteignent la limite d'élasticité σ_0. Les autres couches restent élastiques. La plastification se propage donc le long de la section, couche par couche, jusqu'à ce qu'elle se généralise. Cette approche équivaut à une intégration numérique de la relation contrainte – déformation sur la section de la poutre.

La division en couches demande néanmoins un effort de modélisation supplémentaire important, surtout si la forme de la section de la poutre n'est pas simple (triangulaire, en T, en I, …). On peut en arriver à perdre l'intérêt de la modélisation très simple des poutres qui n'exige, dans le plan, que deux points pour les définir entièrement.

Pour éviter ce problème, nous avons pensé à insérer les relations constitutives du chapitre 3, qui ont été intégrées analytiquement, dans le programme présenté ci-dessus afin de gagner en précision sans perdre les avantages d'une modélisation simple.

4.3 L'insertion des relations moment - courbure théoriques dans un code de calcul numérique

L'objectif de ce travail de magister est d'améliorer le programme utilisant la méthode des éléments finis présenté dans le paragraphe précédent. Il s'agit alors d'y introduire des relations moment - courbure (M-χ) intégrées analytiquement au lieu de la relation simplifiée bilinéaire montrée dans la figure (4.2) et dont les inconvénients ont été cités au paragraphe 4.2.3.1.

Comme l'intégration de la loi de comportement se fait sur la section de la poutre, il est alors évident que chaque forme a sa propre relation moment - courbure. Pour montrer l'intérêt de la démarche proposée, nous avons retenu deux types de sections assez complexes et qui sont non symétriques par rapport à l'axe de flexion. Il s'agit des sections de forme triangulaire (figure 3.7) assez peu conventionnelle, et des sections en T (figure 3.9) très utilisées en génie

civil. Le travail d'intégration des relations moment - courbure pour ces deux types de sections a été développé dans un mémoire de DEA de l'université Blaise Pascal de Clermont-Ferrand, France, [4] et a été présenté au chapitre 3 de ce mémoire.

Dans ce paragraphe, nous nous sommes intéressé à la façon d'introduire ces courbes dans un code de calcul numérique. Le travail cité ci-dessus a eu pour objectif d'insérer ces mêmes courbes dans un programme informatique écrit sur la base de la méthode des différences finies pour étudier les poutres classiques de Bernoulli soumises à la flexion simple.

Pour la mise en œuvre de ce travail, il était nécessaire de déterminer la valeur de la courbure à l'itération r, $\chi^{(r)}$, à partir de la valeur de la courbure à l'itération précédente $\chi^{(r-1)}$ et de la valeur de l'incrément de moment ΔM :

$$\left(\chi^{(r-1)}, \Delta M\right) \rightarrow \chi^{(r)} \tag{4.22}$$

L'auteur a alors utilisé deux stratégies. La première se sert directement des expressions (M-χ) théoriques et utilise des méthodes numériques pour résoudre les équations rencontrées. On l'appellera « résolution par les méthodes numériques ». La deuxième utilise les réseaux de neurones pour réaliser l'interpolation schématisée par l'équation (4.22). On l'appellera « résolution par les réseaux de neurones selon la référence [4] ». Cette stratégie a donné des résultats très dispersés et n'a donc pas été concluante.

Dans ce qui suit, nous commençons par présenter les deux solutions citées ci-dessus. Nous proposerons ensuite une stratégie de substitution qui devrait améliorer la qualité des résultats issus de l'utilisation des réseaux de neurones comme outil d'interpolation de la loi de comportement.

4.3.1 Résolution par les méthodes numériques.

Dans cette approche, on utilise directement les relations moment - courbure analytiques établies au paragraphe (3.4) pour les sections triangulaires et au paragraphe (3.5) pour les sections en T.

Pour que ces courbes soient les plus générales possibles, elles ont été développées en utilisant des variables sans dimension. Elles sont données sous la forme d'équations paramétriques. Les expressions des variables adimensionnelles associées au moment et à la courbure (notées m et k, respectivement) sont établies en fonctions du paramètre η_1 défini au paragraphe (3.3.4) (ou par la figure 3.6). La relation moment - courbure est sous la forme :

$$m(\eta_1)\text{-}k(\eta_1).$$

Ainsi, pour évaluer, à partir de ces formules, la courbure à l'itération r, χ^r, en fonction de la courbure à l'itération précédente $\chi^{(r-1)}$ et de l'incrément de moment ΔM, il faut opérer comme suit :

1. Calculer les valeurs sans dimensions associées à $\chi^{(r-1)}$ et ΔM à l'aide des équations (3.49) et (3.46), respectivement. On les note $k^{(r-1)}$ et Δm, respectivement.
2. Déterminer la valeur de η_1 à l'itération (r-1), $\eta_1^{(r-1)}$, telle que $k(\eta_1^{(r-1)})=k^{(r-1)}$. On utilise alors un algorithme de dichotomie pour résoudre cette équation.
3. Calculer la valeur du moment adimensionnel à l'itération r : $m^{(r)}=m(\eta_1^{(r-1)})+\Delta m$.

4. Evaluer la valeur de η_1 à l'itération r, $\eta_1^{(r)}$, telle que $m(\eta_1^{(r)})=m^{(r)}$. Cette équation est alors résolue par un algorithme de dichotomie.
5. Calculer la courbure sans dimension : $k^{(r)}=k(\eta_1^{(r)})$.
6. Déduire la courbure recherchée $\chi^{(r)}$ en utilisant l'équation (3.49).

Il faut noter que les étapes 1 et 6 ne servent qu'à passer aux variables adimensionnelles et en revenir. Elles seront toujours présentes et nécessaires dans tout processus utilisant les relations moment - courbure adimensionnelles (m – k).

Quant aux étapes 2 et 4, le problème posé est de rechercher la valeur η sur l'intervalle [a,b], telle que $f(\eta)=f_0$, ce qui revient à chercher la racine de la fonction $h(\eta)=f(\eta)-f_0$ sur ce même intervalle. On utilise alors la méthode de la bissectrice dont l'algorithme est le suivant :

- η_0=a, η_2=b.
- Répéter :
  - η =(η_0+ η_2)/2
  - si $h(\eta)\times h(\eta_0)<0$ alors η_2= η
  - sinon η_0= η,

  Jusqu'à ce que (η_2- η_0)<précision

Afin d'éviter ce calcul itératif qui est potentiellement long, nous avons pensé utiliser les réseaux de neurones et leur grande capacité d'apprentissage pour évaluer directement $k^{(r)}$ à partir de $k^{(r-1)}$ et Δm. Le premier travail effectué dans ce sens a été réalisé dans la référence [4]. La stratégie qui y a été utilisée est présentée ci-après. Nous donnerons ensuite la solution que nous proposons pour utiliser les réseaux de neurones artificiels.

4.3.2 Résolution par les réseaux de neurones selon la référence [4].

La stratégie retenue dans la référence [4] consiste à trouver des réseaux de neurones dont les entrées (inputs) sont $k^{(r-1)}$ et Δm, et dont la sortie (output) est $k^{(r)}$. Autrement-dit, ces réseaux doivent assurer l'interpolation de la fonction f, telle que :

$$k^{(r)} = f\left(k^{(r-1)}, \Delta m\right) \tag{4.23}$$

Ainsi, ces réseaux ont pour mission de remplacer les étapes 2, 3, 4 et 5 de l'algorithme présenté au paragraphe précédent.

L'apprentissage de ces réseaux s'est fait en utilisant des triplets de valeurs ($k^{(r-1)}$, Δm, $k^{(r)}$) calculées à partir des expressions théoriques établies au chapitre 3. Ces triplets sont censés couvrir toutes les situations possibles afin que les réseaux éduqués aient le domaine d'utilisation le plus général possible.

A ce niveau, il faut noter que, pour les sections triangulaires, les fonctions $k(\eta_1)$ et $m(\eta_1)$ sont indépendantes des dimensions de la poutre étudiée, c'est à dire que la relation k-m (sans dimensions) est unique pour un paramètre de ductilité p fixé (p=20 ; voir le § 3.3.1, équation 3.71). Ce n'est pas le cas des sections en T dont les courbes k et m dépendent également de la forme de la section et plus précisément des variables sans dimension, τ et ζ, associées aux

épaisseurs de l'âme et de la table, respectivement (voir les équations 3.98, 3.100 et 3.102). Autrement dit, il y a une infinité de relations moment - courbure adimensionnelles dans le cas des sections en T.

L'objectif de l'étude était de chercher à utiliser efficacement les réseaux de neurones pour étudier la flexion simple des poutres. Il était alors impensable d'utiliser un nombre trop élevé de réseaux pour remplacer l'algorithme du paragraphe précédent. Après avoir tracé plusieurs courbes moment - courbure de sections en T, l'auteur conclut qu'elles étaient « relativement proches » les unes des autres. Il décida alors d'éduquer des réseaux de neurones en considérant une **seule** (!!!) courbe m-k qu'il a qualifiée de **« MOYENNE »** et qui correspond à la section en T à ailes égales et à coins arrondis de classe C1 (voir § 3.5.4) définie par :

$$b=h=70mm$$

$$t_f=t_w=8mm$$

Ce qui donne pour les épaisseurs sans dimension

$$\tau = \zeta = 8/70$$

Il paraît évident qu'on est loin de couvrir toutes les « situations possibles » comme cela devrait être le cas. De plus, et malgré cette simplification (et d'autres qu'on évitera d'étaler dans ce mémoire), l'auteur a été obligé de recourir à plusieurs réseaux de neurones pour interpoler la relation (4.23). Il a utilisé 5 réseaux pour les sections triangulaires et 7 pour les (ou plutôt pour la) sections en T, chacun couvrant un domaine de validité particulier.

Tous ces réseaux de neurones sont de type perceptron multicouches et comportent 4 couches de cellules : La couche d'entrée est composée de 2 cellules, la couche de sortie d'une seule cellule et les 2 couches cachées ont un nombre m de cellules. L'architecture d'un tel réseau peut être abrégée sous la notation 2-m-m-1 indiquant ainsi le nombre d'unités d'entrée, le nombre de cellules cachées pour chaque niveau existant puis le nombre d'unités de sortie. La fonction utilisée pour activer les unités d'entrée et de sortie est la fonction identité alors que la fonction sigmoïde logistique est affectée à l'ensemble des cellules cachées. L'algorithme d'apprentissage des réseaux utilisé est la méthode de la rétropropagation du gradient.

Pour montrer la complexité du travail réalisé dans la référence [4], nous donnons le domaine de validité et l'architecture des différents réseaux développés.

1. Pour les sections triangulaires
   - $0.07500 \le k < 0.15774$: réseau n°1 de type 2-6-6-1
   - $0.15774 \le k < 0.20572$: réseau n°2 de type 2-20-20-1
   - $0.20572 \le k < 0.32968$: réseau n°3 de type 2-20-20-1
   - $0.32968 \le k < 0.75669$: réseau n°4 de type 2-20-20-1
   - $0.75669 \le k < 1.41480$: réseau n°5 de type 2-20-20-1
2. Pour les sections en T
   - $0.07062 \le k < 0.10933$: réseau n°1 de type 2-6-6-1
   - $0.10933 \le k < 0.16859$: réseau n°2 de type 2-6-6-1
   - $0.16859 \le k < 0.26289$: réseau n°3 de type 2-6-6-1

$0.26289 \leq k < 0.40757$: réseau n°4 de type 2-20-20-1

$0.40757 \leq k < 0.54184$: réseau n°5 de type 2-20-20-1

$0.54184 \leq k < 0.69538$: réseau n°6 de type 2-20-20-1

$0.69538 \leq k < 1.13685$: réseau n°7 de type 2-20-20-1.

Pour un couple d'entrées $\left(k^{(r-1)}, \Delta m\right)$ l'utilisation du réseau approprié fournit la valeur de sortie $k^{(r)}$. Avec ce nombre relativement élevé de réseaux, l'erreur maximum commise sur des échantillons composés de nombreux éléments, utilisés comme pattern de test, a été de l'ordre de 2% pour les sections en T et de l'ordre de 3.5% pour les sections triangulaires. Sur l'ensemble des valeurs utilisées pour le test, 90% (pour les sections triangulaires) ou 95% (pour les sections en T) des sorties désirées ont été approchées avec un écart inférieur à 1%.

Une fois ce travail d'interpolation de la relation $k^{(r)} = f\left(k^{(r-1)}, \Delta m\right)$ réalisé au moyen des réseaux de neurones, il fallait l'introduire dans un code de calcul à base de différences finies pour étudier la flexion des poutres. Plusieurs poutres à une seule travée avec des conditions aux limites et des sollicitations représentatives des différents cas envisageables ont alors été considérées.

L'erreur entre les résultats de la flèche maximum obtenus à partir du code utilisant les formules théoriques et celui utilisant les réseaux de neurones variait de 1% à 16% pour les sections triangulaires, et de 0.1% à 26.4% pour les sections en T.

La disparité des résultats n'a pas permis à l'auteur de valider le code de calcul utilisant les réseaux de neurones. Des tentatives d'explication ont alors été fournies.

Dans ce qui suit, nous présentons une autre façon d'utiliser les réseaux de neurones pour remplacer les étapes 2, 3, 4 et 5 de l'algorithme dit des « méthodes numériques » présenté au paragraphe (4.3.1).

4.3.3 Méthodologie proposée pour une résolution par les réseaux de neurones

Dans ce travail de magister nous nous sommes fixé comme objectif de prouver que les réseaux de neurones peuvent être utilisés efficacement dans l'étude de la flexion simple des poutres dans le cas de la non linéarité matérielle. Nous utilisons alors une approche différente et beaucoup plus simple que celle présentée ci avant et qui, malheureusement, n'a pas été concluante.

Nous changerons la méthode de résolution des équations aux dérivées partielles régissant la flexion des poutres. Nous utiliserons alors la méthode des éléments finis plutôt que la méthode des différences finies. Ce choix est d'abords dicté par la nécessité de faire différent que le mémoire de DEA. Plus important encore est, que nous pensons que la méthode des éléments finis reste un moyen plus efficace que la méthode des différences finies surtout, lorsque la géométrie de la structure à étudier devient compliquée.

Une autre modification apportée par rapport au travail cité en 4.3.2 est le type de poutre étudiée. Dans ce travail, nous nous intéressons à la poutre de Timoshenko qui permet de modéliser aussi bien les poutres élancées que les poutres épaisses du moment qu'elle prend en considération la déformation due à l'effort tranchant.

Nous voulons néanmoins insister sur le fait que ce changement de méthode de résolution des équations différentielles et la modification du type de poutre sont indépendants du

changement de la stratégie à utiliser pour interpoler les relations moment - courbure théoriques à partir des réseaux de neurones.

Cette correction de stratégie pouvait très bien être apportée au programme en différences finies élaboré dans la référence [4]. Nous n'avons pas fait ainsi, car nous voulions proposer une meilleure méthode dans le sens où la méthode des éléments finis est très ouverte aux extensions. Nous pourrons assez facilement étendre le programme élaboré à l'étude des portiques 2D ou 3D.

L'approche développée dans ce mémoire pour introduire la relation moment - courbure dans un code de calcul en éléments finis au moyen des réseaux de neurones est déduite de la stratégie appelée « résolution par les méthodes numériques » présentée au paragraphe (4.3.1). Nous pouvons observer que, dans ce cas, la démarche consiste à évaluer $m^{(r-1)}$ à partir de $k^{(r-1)}$ en passant par un calcul numérique du paramètre $\eta_1^{(r-1)}$, de rajouter l'incrément de moment Δm à $m^{(r-1)}$ pour obtenir $m^{(r)}$, et enfin, d'en déduire la courbure $k^{(r)}$ en passant de nouveau par une évaluation numérique du paramètre $\eta_1^{(r)}$. Ceci est représenté schématiquement dans la figure (4.4).

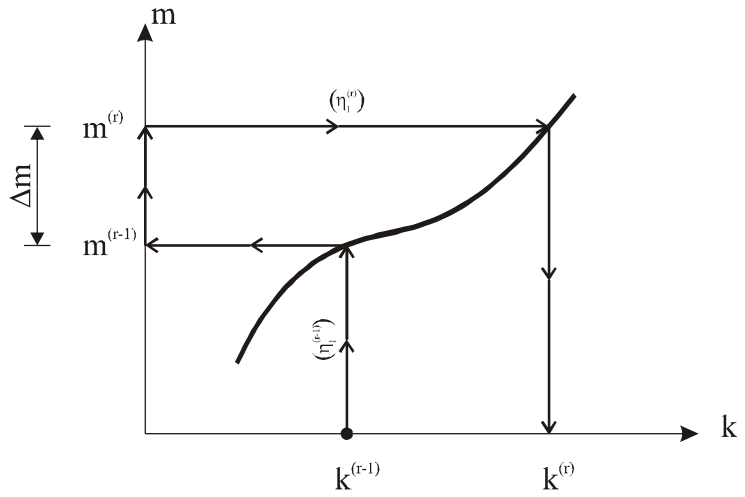


Figure 4.4 : Résolution de $k^{(r)}=f(k^{(r-1)},\Delta m)$

Il s'agit alors de faire un aller-retour sur la même courbe mais en empruntant deux chemins distincts.

Pour réaliser ce travail, nous avons eu l'idée de faire apprendre le chemin « Aller » à un réseau de neurones et le chemin « retour » à un autre. Il est donc nécessaire de trouver deux réseaux pour faire le cycle complet. Le premier assure l'interpolation f_1, telle que :

$$k = f_1(m) \tag{4.24}$$

Et le second fournit l'interpolation f_2 telle que :

$$m = f_2(k) \tag{4.25}$$

Pour évaluer, par exemple, $k^{(r)}$ à partir de $k^{(r-1)}$ et Δm (comme c'est nécessaire dans la résolution par la méthode des différences finies), il suffit d'utiliser le 2ème réseau de neurones

assurant l'interpolation $m^{(r-1)} = f_2\left(k^{(r-1)}\right)$, de rajouter Δm à $m^{(r-1)}$ pour avoir $m^{(r)}$ et d'utiliser ensuite le premier réseau de neurones pour avoir $k^{(r)} = f_1\left(m^{(r)}\right)$.

Dans la résolution par la méthode des éléments finis dans sa version méthode des déplacements, qui est la méthode utilisée dans ce mémoire, à l'itération (r), nous disposons plutôt de $m^{(r-1)}$ et de Δk, et il est alors nécessaire d'évaluer $m^{(r)}$ (voir § 4.2.3).

Dans ce cas, nous commençons par utiliser le 1er réseau d'interpolation pour avoir $k^{(r-1)} = f_1\left(m^{(r-1)}\right)$. Nous rajoutons à cette courbure l'incrément de courbure Δk pour avoir $k^{(r)}$ pour enfin, utiliser le 2ème réseau de neurones afin d'avoir la valeur recherchée $m^{(r)} = f_2\left(k^{(r)}\right)$.

Pour mettre en œuvre cette méthodologie, nous avons commencé par le cas des sections triangulaires car, pour un paramètre de ductilité p fixé (p=20 ; voir § 3.3.1), la relation moment - courbure adimensionnelle de ces sections est unique. C'est à dire que cette relation est indépendante des dimensions de la poutre étudiée (voir figure.3.7, § 3.4.3) ce qui en fait une courbe « facile à apprendre » par les réseaux de neurones. Dans une seconde étape, nous avons cherché à interpoler la loi de comportement des sections en T. Elle est plus difficile que la précédente car son expression dépend de la forme de la section étudiée.

Les réseaux de neurones résultant de cette recherche sont utilisés pour programmer (en langage PASCAL) la stratégie proposée qui est alors insérée dans le code de calcul (éléments finis) présenté au paragraphe (4.2.3).

4.4 Interpolation de la loi de comportement des sections triangulaires.

La relation moment - courbure des sections triangulaires est donnée dans le paragraphe (3.4.3). Pour un paramètre de ductilité p fixé, les expressions de la courbure et du moment adimensionnels ne dépendent que du paramètre η_1 défini dans la figure (3.6) : $k(\eta_1)$ et $m(\eta_1)$. Cette loi de comportement est indépendante des dimensions de la poutre étudiée et caractérise donc l'ensemble des poutres de section triangulaire (pour un p donné). Dans ce mémoire, nous avons fixé le paramètre p=20, ce qui correspond à la ductilité minimale requise pour l'acier (voir § 3.3.1).

Pour réaliser l'apprentissage de cette relation par les réseaux de neurones, il est nécessaire de disposer d'un ensemble de couples (m,k) constituant le pattern d'apprentissage. Nous avons alors écrit, sous MATLAB, un petit programme (script) nommé « Plast.m » (voir annexe A.1) pour générer ces couples de données et pour dessiner la courbe m-k (Figure 4.5).

Conformément à la stratégie présentée dans le paragraphe (4.3.3), il faut deux réseaux de neurones pour réaliser le cycle complet aller-retour sur la loi de comportement. Pour le premier réseau l'entrée est m et la sortie k, alors que pour le deuxième, nous utilisons comme entrée k et comme sortie m. Nous les notons, respectivement, KvsM2 et MvsK2.

Nous avons cherché, par tâtonnement, deux réseaux ayant une unité d'entrée et une unité de sortie, le nombre, la disposition des cellules cachées et les fonctions d'activation étant variables. L'algorithme utilisé pour l'apprentissage des réseaux est la méthode de la rétropropagation du gradient.

Nous avons commencé par « tâtonner » en utilisant un programme appelé PROPAGATOR [18]. Voyant que le travail n'aboutissait pas car le niveau des erreurs d'interpolation restait trop élevé par rapport aux objectifs souhaités, le doute s'est installé et nous avons employé le

logiciel MATLAB et son « Neural Network Toolbox » [5]. Après plusieurs tentatives qui ont pris beaucoup de temps en cherchant des réseaux compliqués, l'architecture retenue est finalement assez simple. A ce niveau, il faut noter que ce temps et cette énergie perdus nous ont appris qu'avec les réseaux de neurones, il faut avoir une démarche très simple. Il s'agit d'avoir une attitude similaire à celle qu'on adopterait face à des enfants. Il faut donc être très pédagogue, méthodique et leur présenter les choses le plus simplement possible. D'un autre côté, nous pensons que, paradoxalement, il est plus « facile de retenir » un réseau de neurone pour interpoler un problème réputé compliqué (mouvements sismiques, glissements de terrains, …) car on se suffirait d'un niveau d'erreur relativement élevé. Dans notre cas où il fallait reproduire une solution exacte « compliquée » (tout est relatif !), nous avions besoin de faire baisser l'erreur d'interpolation à un niveau très bas et on s'est égaré car nous nous sentions obligés de faire compliqué.

L'architecture retenue pour les deux réseaux comporte en plus de la couche d'entrée à une cellule et de la couche de sortie à cellule unique, une seule couche cachée comportant huit cellules. La fonction d'activation de la couche cachée est la fonction sigmoïde logistique (nommée Logsig dans MATLAB), alors que la fonction d'activation de la couche de sortie est la fonction identité (Purelin). Cette architecture des deux réseaux KvsM2 et MvsK2 est schématisée par la figure (4.6).

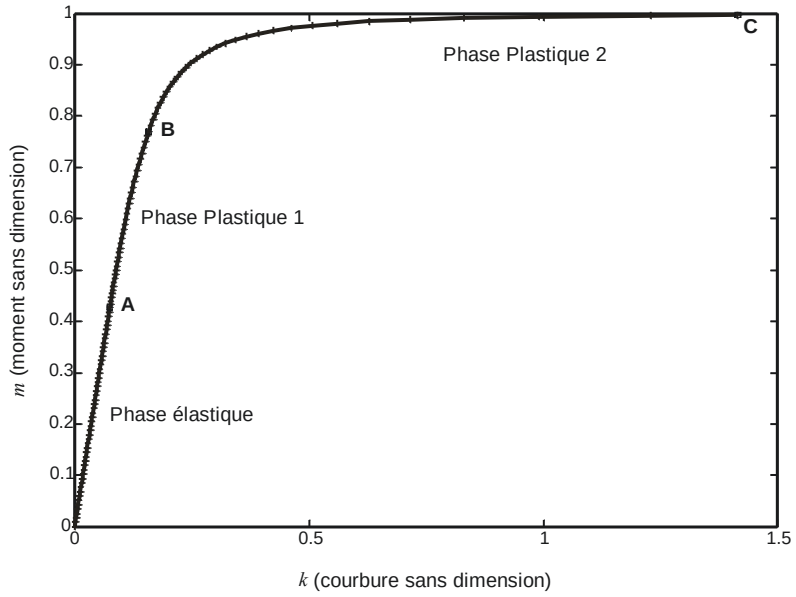


Figure 4.5 : Relation m-k des sections triangulaires. (p=20)

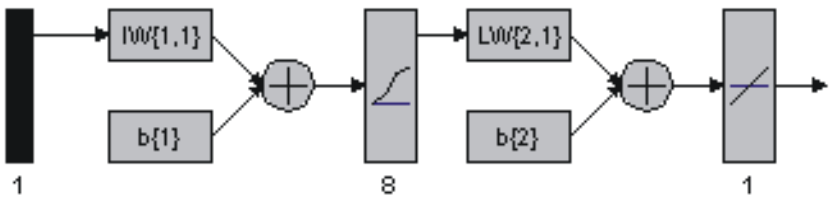
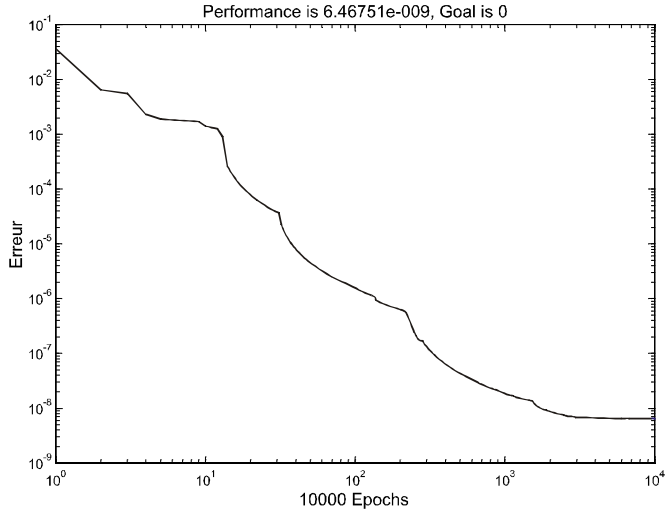


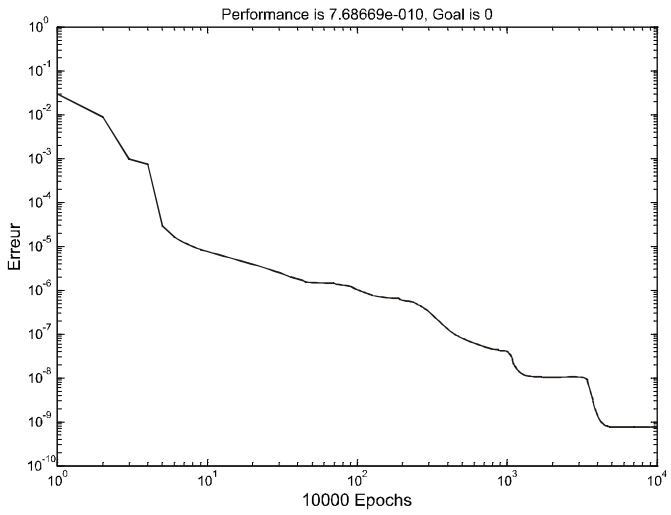
Figure 4.6 : Architecture des réseaux permettant l'interpolation de la loi de comportement des sections triangulaires.

L'évolution, lors de l'apprentissage, de la moyenne quadratique des erreurs, appelée performance, entre les valeurs souhaitées (targets) et celles évaluées par le réseau (outputs) en fonction du nombre de cycles d'apprentissage (epochs) est donnée dans la figure (4.7) pour les deux réseaux. Après 10 000 cycles, les performances sont descendues à des valeurs très faibles : $6{,}5\ 10^{-9}$ et $7{,}7\ 10^{-10}$.

La valeur absolue de l'erreur maximum commise sur des échantillons « test » composés de nombreux éléments, est de $3{,}3\times10^{-4}$ pour le réseau KvsM2 et de 9×10^{-5} pour le réseau MvsK2.



a. Réseau KvsM2



b. Réseau MvsK2

Figure 4.7 : Erreur quadratique en fonction du nombre de cycles d'apprentissage (sections triangulaires)

Avec des erreurs aussi faibles, nous estimons que l'apprentissage de la relation moment - courbure adimensionnelle des sections triangulaires est réalisé de manière quasi parfaite et avec ces deux réseaux nous pouvons effectuer le cycle aller-retour sur la loi de comportement nécessaire à la stratégie de résolution retenue.

En comparaison avec la stratégie adoptée dans la référence [4], cela ne nous fait que deux réseaux 1-8-1 (conformément à la notation introduite au § 4.3.2) en remplacement des 5 réseaux (à deux couches cachées et ayant jusqu'à 20 cellules par couche cachée) utilisés pour interpoler la loi de comportement adimensionnelle des sections triangulaires.

Une fois l'apprentissage terminé, nous récupérons les valeurs des poids des connexions synaptiques et les valeurs des biais des cellules qui ont été optimisés pour l'interpolation à l'aide du logiciel MATLAB afin de les utiliser dans le code de calcul en éléments finis que nous avons baptisé **BEAMEPRN.PAS.**

D'après la figure (4.6) montrant l'architecture des deux réseaux, il faut récupérer, pour chacun d'eux, la matrice IW{1,1} des poids des connections entre la seule cellule d'entrée et les 8 cellules de la seule couche cachée ; sa dimension est 8×1. Il faut aussi récupérer le vecteur b{1} constitué des 8 biais des cellules cachées, la matrice LW{2,1} de dimension (1×8) des poids des connections entre les 8 cellules de la couche cachée et la seule cellule de sortie et enfin la valeur des biais b{2} de la couche de sortie.

A partir de ces vecteurs et matrices, si nous fournissons une valeur X comme entrée à un réseau, nous obtenons comme sortie la valeur Y telle que :

$$Y = \underbrace{\text{Purelin}\Big(\underbrace{LW\{2,1\}}_{1\times 8} \times \underbrace{\text{Logsig}\Big(\underbrace{IW\{1,1\}}_{8\times 1} \times \underbrace{X}_{1\times 1} + \underbrace{b\{1\}}_{8\times 1}\Big)}_{8\times 1} + \underbrace{b\{2\}}_{1\times 1}\Big)}_{\mathbf{1\times 1}}$$

Comme l'architecture des réseaux KvsM2 et MvsK2 est la même, nous avons écrit une seule fonction (SIM) pour les simuler dans le programme élaboré. Selon la valeur donnée à une variable nommée Flag, elle permet soit de calculer l'interpolation

$$k = f_1(m) \qquad \text{pour Flag} = 0$$

Soit d'évaluer l'autre interpolation

$$m = f_2(k) \qquad \text{pour Flag} \neq 0$$

Quand à la stratégie présentée au paragraphe (4.3.3) pour évaluer le moment $m^{(r)}$ à l'itération r à partir de $m^{(r-1)}$ et Δk, elle est mise en œuvre dans la fonction Mip1 du programme BEAMEPRN.PAS. Le reste de ce code est quasiment identique à celui utilisant la relation moment - courbure bilinéaire.

Dans ce qui suit, le travail de recherche des réseaux d'interpolation et de programmation présenté ci-dessus va être effectué pour le cas des sections en T qui sont beaucoup plus fréquemment utilisées en Génie Civil que les sections triangulaires.

4.5 Interpolation de la loi de comportement des sections en T.

Contrairement aux sections triangulaires et pour un paramètre de ductilité p fixé, la relation moment - courbure adimensionnelle des sections en T n'est pas unique. Elle dépend des épaisseurs adimensionnelles τ et ζ de l'âme et de la table respectivement (voir § 3.5).

Pour générer deux vecteurs (m et k) représentant la relation moment - courbure pour des valeurs de τ, ζ et p données, nous avons écrit une fonction sous MATLAB nommée Tplas.m (voir annexe A.2). Elle utilise les différentes expressions présentées au paragraphe (3.5.4) et couvre tous les cas de figures, c'est à dire toutes les classes possibles de sections en T (A1, A2, B, C1 et C2).

Nous avons ensuite fixé le paramètre p=20 et nous avons calculé, à l'aide de la fonction « Tplas.m », 200 relations moment - courbure correspondant à 200 couples (τ,ζ) générés aléatoirement à l'intérieur de l'intervalle [0.03 , 0.5] (Voir annexe A.3). Nous les avons toutes tracées dans la figure (4.8). Nous y avons également représenté, en blanc, la courbe correspondant à $\tau = \zeta = 8/70$ qui a été retenue comme courbe « moyenne » pour résumer le comportement des sections en T dans la référence [4]. Nous observons alors que l'ensemble de ces courbes dessine un fuseau relativement épais.

Il parait alors évident qu'il est impensable de réduire cet ensemble de courbes à une seule courbe « moyenne ». Il serait impossible d'obtenir des résultats précis dans tous les cas. Nous rappelons que, dans la référence [4], l'erreur commise suite à l'utilisation des réseaux de neurones a atteint 26.4% ce qui est énorme et nous pensons que cette erreur pouvait être plus grande pour d'autres classes de poutres que celles étudiées.

Il faut donc que les deux réseaux que nous cherchons à développer, pour mettre en application la stratégie proposée, puissent effectuer l'interpolation en tenant compte également de τ et ζ. Ainsi, le premier réseau doit pouvoir assurer l'interpolation

$$k = g_1(\tau, \zeta, m) \tag{4.26}$$

et le second doit permettre d'évaluer

$$m = g_2(\tau, \zeta, k) \tag{4.27}$$

Avec ces 2 interpolations, on peut effectuer l'aller - retour dans la loi de comportement.

Les deux réseaux que nous cherchons doivent avoir 3 cellules d'entrée et 1 cellule de sortie. Le nombre et la disposition des cellules cachées ainsi que les fonctions d'activation sont des variables à déterminer par tâtonnement. L'algorithme utilisé pour l'apprentissage des réseaux est la méthode de la rétropropagation du gradient.

Pour couvrir un maximum de possibilités, nous utilisons comme pattern d'apprentissage 200 courbes moment - courbure correspondant à 200 couples (τ,ζ) générés aléatoirement à l'intérieur de l'intervalle [0.03,0.5]. Chacune de ces courbes est définie par « 91 » couples (m, k) (voir annexe A.3).

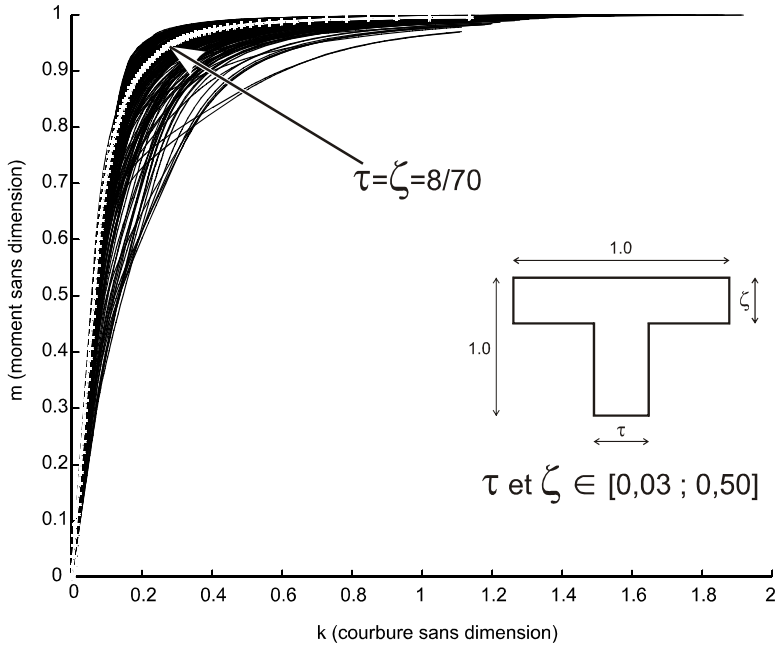


Figure 4.8 : Relation m-k de 200 sections en T. (p=20)

Pour éduquer les réseaux, nous disposons donc de 200×91 soit 18 200 quadruplés (m, k, τ, ζ) :

- Le réseau n 1 aura 18 200 données (τ, ζ, m) pour 18 200 k désirées.
- Le réseau n 2 aura 18 200 entrées (τ, ζ, k) pour 18 200 sorties m.

Après plusieurs tentatives et à l'aide du logiciel MATLAB, l'architecture retenue pour les deux réseaux se compose, en plus de la couche d'entrée à 3 cellules et de la couche de sortie à une cellule, de deux couches cachées comportant chacune 10 cellules. La fonction d'activation des cellules des 2 couches cachées est la fonction sigmoïde logistique (Logsig) alors que la fonction d'activation de la couche de sortie est la fonction identité (Purelin).

Le réseau assurant l'interpolation g_1 est nommé « Km310101 » et celui fournissant la fonction g_2 est appelé « Mk310101 ». Leur architecture est représentée dans la figure (4.9).

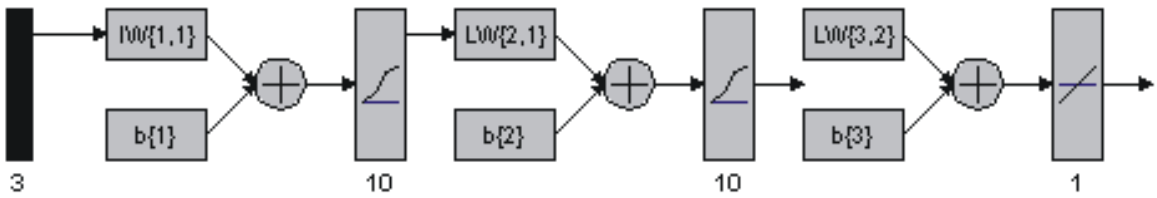


Figure 4.9 : Architecture des réseaux permettant l'interpolation de la loi de comportement des sections en T.

Cette architecture a été retenue car après 10 000 cycles d'apprentissage (epochs), la moyenne quadratique des différences entre les 18 200 valeurs calculées par le réseau et celles souhaitées, appelée performance, est descendue à un niveau de l'ordre de 10^{-6}. L'évolution de cette erreur en fonction du nombre de cycles d'apprentissage (pour les deux réseaux) est représentée dans la figure (4.10).

L'erreur maximum commise sur les 18 200 échantillons de valeurs proposées au réseau pour l'apprentissage est de 0,0127 pour le réseau Km310101 et de 0,0140 pour le réseau Mk310101. Nous avons alors jugé que ces 2 réseaux peuvent être retenus pour interpoler les deux relations (4.26 et 4.27) qui sont utilisées pour mettre en œuvre la stratégie proposée au paragraphe (4.3.3).

Il faut noter, à titre de comparaison, que ces deux réseaux 3-10-10-1 (conformément à la notation introduite au § 4.3.2) sont développés en remplacement de 7 réseaux (3 réseaux 2-6-6-1 et 4 réseaux 2-20-20-1) utilisés dans la référence [4] pour interpoler la loi de comportement des sections ou plutôt de la section en T.

Pour utiliser les résultats de cet apprentissage réalisé sur MATLAB dans le code éléments finis développé en langage PASCAL, il est nécessaire de récupérer les valeurs des poids des connections synaptiques et des biais des cellules. Conformément à l'architecture montrée dans la figure (4.9), il s'agit des matrices IW{1,1}, LW{2,1} et LW{3,2} de dimensions respectives 10×3, 10×10 et 1×10 ainsi que des vecteurs de biais b{1}, b{2} et b{3} de dimensions 10×1, 10×1 et 1×1, respectivement.

Si nous fournissons un triplet de valeurs, noté X et de dimension 3×1, comme donnée à l'un de ces réseaux, la valeur de sortie Y, de dimension 1×1, est obtenue par :

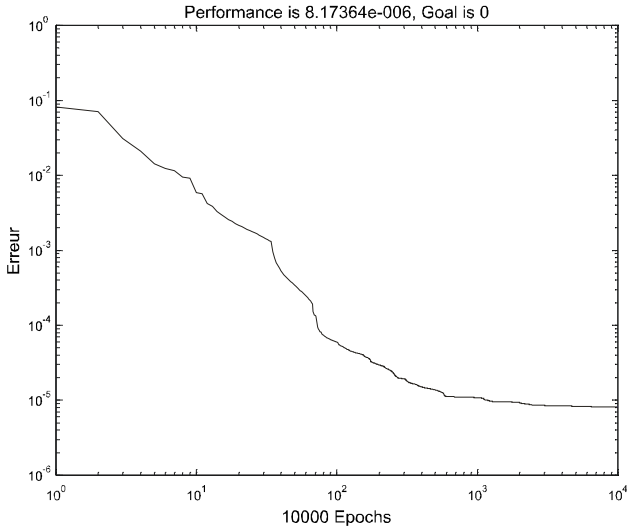
$$\underbrace{Y = Purelin\Big(\underbrace{LW\{3,2\}}_{\mathbf{1\times 10}}\times \underbrace{Logsig\big(\underbrace{LW\{2,1\}}_{10\times 10}\times \underbrace{Logsig\big(\underbrace{IW\{1,1\}}_{10\times 3}\times \underbrace{X}_{3\times 1} + \underbrace{b\{1\}}_{10\times 1}\big)}_{10\times 1} + \underbrace{b\{2\}}_{10\times 1}\big)}_{\mathbf{10\times 1}} + \underbrace{b\{3\}}_{\mathbf{1\times 1}}\Big)}_{\mathbf{1\times 1}} \qquad (4.28)$$

Pour que l'exportation de l'ensemble de ces matrices et vecteurs à partir du logiciel MATLAB reste pratique, nous avons pensé les regrouper dans un seul fichier par réseau. Ainsi, à partir de MATLAB, nous avons créé deux fichiers, un par réseau, contenant l'ensemble des valeurs des poids et des biais ($=10\times 3+10\times 10+1\times 10+10+10+1=161$ valeurs). Il s'agit de : Km310101.WGT et Mk310101.WGT. Nous avons ensuite écrit des fonctions en PASCAL qui permettent de lire correctement les valeurs stockées dans ces fichiers afin de les utiliser dans la fonction (SIM) qui effectue, dans le programme élaboré, le calcul de l'équation (4.28).

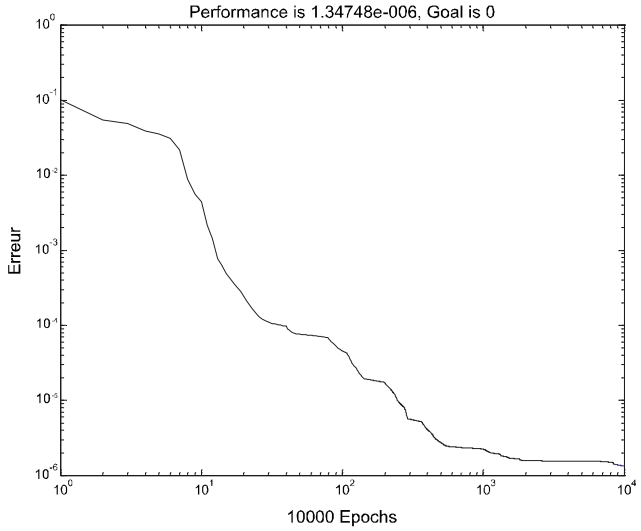
La stratégie présentée au paragraphe (4.3.3) pour évaluer le moment $m^{(r)}$ à l'itération r à partir de $m^{(r-1)}$ et Δk (et en fonction de τ et ζ !) est mise en œuvre dans la fonction (MiP1).

Toutes les fonctions relatives à l'utilisation de ces réseaux de neurones dans le code éléments finis sont regroupées dans l'unité PASCAL « NNTOOL.PAS » (annexe B.2). Le

programme principal contenant le code éléments finis est dans le fichier « T2.PAS ». Son code source est donné dans l'annexe B.1.



a. Réseau Km310101



b. Réseau Mk310101

Figure 4.10 : Erreur quadratique en fonction du nombre de cycles d'apprentissage (sections en T)

4.6 Conclusion

Dans ce chapitre nous avons commencé par écrire un programme en langage Pascal permettant de calculer la flexion simple des poutres de Timoshenko en élasto-plasticité par la méthode des éléments finis. La relation moment - courbure est alors fournie par la courbe simplifiée bilinéaire.

Pour améliorer ce programme, nous avons voulu y implanter des relations moment - courbure résultant d'un développement analytique, donc, plus performantes que la précédente mais aussi, plus compliquées. Les courbes retenues correspondent à deux types de sections transversales de poutres non symétriques (pour faire compliqué !) par rapport à l'axe de flexion : il s'agit des sections triangulaires et des sections en T.

Pour la mise en œuvre pratique de cette idée, nous avons pensé utiliser les réseaux de neurones artificiels réputés pour leur grande capacité d'apprentissage. Nous en avons retenu un type des plus simples : il s'agit des réseaux de type « PERCEPTRONS multicouches » utilisant la rétropropagation du gradient comme règle d'apprentissage.

Nous avons alors proposé une stratégie d'implantation et nous avons ensuite cherché les réseaux de neurones permettant d'effectuer différentes tâches d'interpolation nécessaires à cette stratégie. La recherche des réseaux adéquats et leur optimisation se sont faites à l'aide du logiciel MATLAB.

Finalement, les relations moment - courbure des poutres à sections triangulaires et en T ont été introduites dans le code éléments finis sous la forme de procédures programmées selon la stratégie proposée et utilisant les réseaux de neurones artificiels.

Dans le chapitre qui suit, nous allons tester le programme élaboré et l'efficacité de la stratégie proposée. Nous prendrons différents exemples de poutres fléchies qui se veulent représentatifs des différentes situations envisageables. Nous comparerons alors les résultats obtenus à ceux trouvés dans la littérature et à ceux fournis par le logiciel CAST3M.

Chapitre 5

Résultats et interprétations

5.1 Introduction

Dans le chapitre précédent nous avons élaboré un programme à base d'éléments finis pour étudier la flexion simple des poutres en élasto-plasticité. Les lois de comportement des sections triangulaires et en T, soit les relations moment - courbure résultant d'un développement analytique, y ont été introduites selon une stratégie utilisant les réseaux de neurones artificiels.

Dans ce chapitre, nous allons tester ce code de calcul et la stratégie adoptée par des exemples simples. Il s'agit de poutres à une seule travée de longueur L unitaire, avec des conditions aux limites et des sollicitations représentatives des différents cas envisageables. Nous étudierons alors des poutres isostatiques et hyperstatiques, chargées soit par un moment, soit par une force ponctuelle ou répartie. Les différentes configurations considérées sont illustrées en figure (5.1).

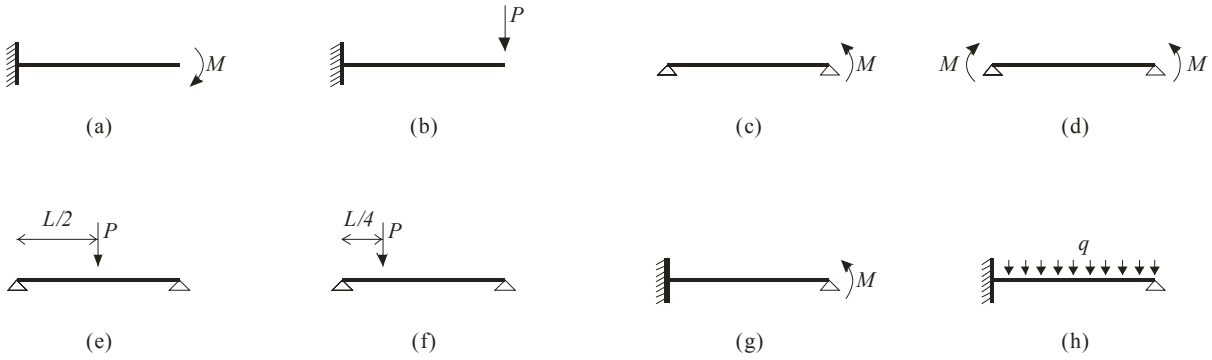


Figure 5.1 : Les schémas statiques traités

Dans une première étape, nous présentons des résultats relatifs au comportement d'une poutre à section triangulaire. La géométrie retenue pour l'étude est représentée en figure (5.2).

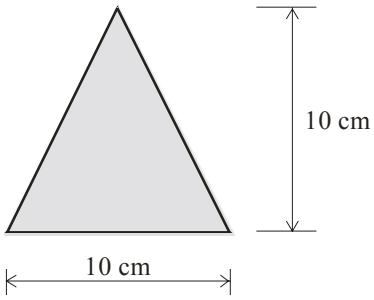


Figure 5.2 : La section triangulaire étudiée

Ensuite, nous présentons les résultats se rapportant à des poutres ayant des sections en T. On considère alors cinq géométries différentes, soit un exemple par type de section en T (C1, A1, B, C2 et A2, conformément aux définitions introduites au chapitre 3). Ces formes retenues pour l'étude sont illustrées en figure (5.3).

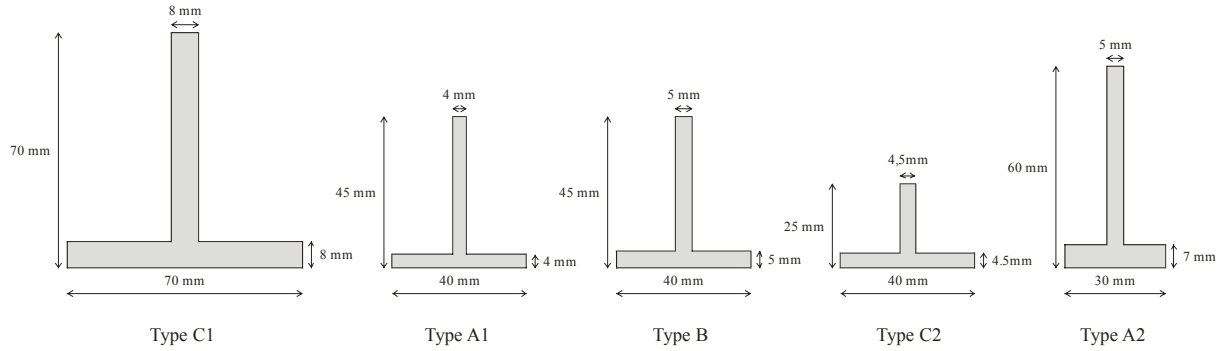


Figure 5.3 : Les sections en T étudiées

5.2 Exemple de fichier de données

L'utilisation du programme développé nécessite l'élaboration de fichiers de données. Ces derniers doivent définir la géométrie et les propriétés mécaniques du modèle, ainsi que le chargement qui lui est appliqué. Préalablement à leur création, il faudra donc choisir le maillage : le nombre de nœuds et leurs coordonnées, le nombre d'éléments et leurs connectivités, le nombre d'appuis et les valeurs des déplacements imposés, … Il faut également mener un calcul afin d'estimer les propriétés géométriques (surfaces, inerties, …) et mécaniques (moment plastique, moment limite d'élasticité, …).

Dans un calcul non linéaire le chargement est appliqué progressivement de 0 à sa valeur finale en plusieurs étapes ou incréments. Le premier incrément dans le fichier de données correspondra à une sollicitation légèrement inférieure à celle provoquant l'apparition du premier point plastique dans la poutre étudiée. On s'arrange ainsi à « passer » la phase

élastique en une seule étape. D'un autre côté, la somme de tous les incréments est prise égale à une valeur légèrement inférieure à la sollicitation de ruine de la section.

A titre d'exemple, nous donnons dans le tableau (5.1) le fichier utilisé pour étudier la poutre console de section triangulaire (figure 5.2), de longueur 1 m et soumise à un moment de flexion en son extrémité libre (schéma statique de la figure 5.1.a). La première ligne du fichier contient un titre qui sera reporté dans le fichier des résultats. La ligne n°2 comporte les informations générales sur le modèle tel que le nombre de nœuds (NPoin=21), le nombre d'éléments (NElem=20), le nombre de conditions d'appuis (NBoun=1), le nombre de matériaux (NMats=1), … La ligne 3 de ce fichier définit le seul « matériau » du modèle. On y trouve les dimensions *b* et *h* de la section, le module de Young *E*, le module de cisaillement *G* et la limite d'élasticité σ_0. Les lignes 4 à 23 contiennent les connectivités des éléments ainsi que le numéro du matériau correspondant. Dans les lignes 24 à 44, on trouve les coordonnées des nœuds. La ligne 45 définit l'encastrement au nœud 1 en imposant des déplacements nuls à ses deux degrés de liberté. Le chargement de référence est fourni dans la ligne 47 : un moment unitaire est appliqué à l'extrémité de l'élément 20. Les lignes 48 à 58 donnent les informations sur la mise en charge incrémentale et sur le calcul itératif à chaque incrément. On y trouve dans l'ordre :

- NIter qui est le nombre maximum d'itérations,
- NOutP qui est un indicateur pour imprimer ou non les résultats de cet incrément de charge,
- Facto qui, multiplié par le chargement de référence, définit l'incrément de charge et
- Toler qui fixe la précision recherchée.

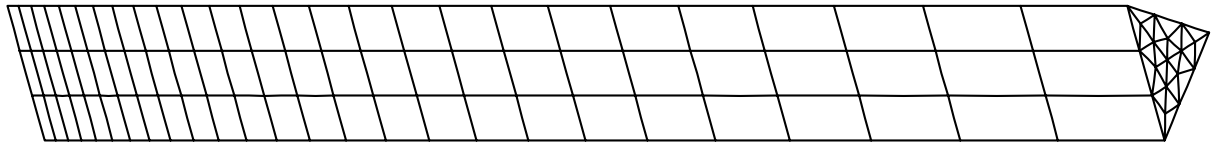
Il faut remarquer que pour cet exemple, le moment total appliqué, définit par la somme des Facto fois le chargement de référence, est égal à 18,74 KN.m.

Une fois le fichier de données créé, nous pouvons exécuter le programme élaboré. Il fournit alors, dans un fichier de résultats, l'état de contrainte et la déformée pour chaque incrément de charge. A l'aide d'un logiciel de représentation graphique, il est alors possible de dessiner, par exemple, la déformée de la poutre pour un incrément de charge donnée ou l'évolution de la flèche d'un point particulier en fonction du chargement. Nous pouvons aussi tracer le diagramme des efforts internes *M(x)* et *T(x)*.

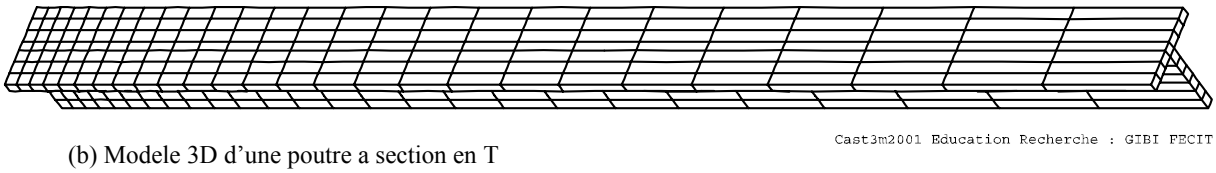
Pour valider le programme élaboré et la stratégie basée sur les réseaux de neurones adoptée, nous avons comparé les résultats obtenus à ceux fournis par le logiciel d'éléments finis CAST3M. Nous avons utilisé une analyse non linéaire avec une loi de comportement élasto-plastique parfaite sur des modèles tridimensionnels. Les poutres à sections triangulaires sont modélisées par des éléments prismatiques à quinze nœuds (PR15) alors que pour discrétiser les poutres en T on a utilisé les éléments cubiques à vingt nœuds (CU20). Le modèle illustré dans la figure (5.4.a) est celui utilisé pour étudier la poutre donnée comme exemple dans ce paragraphe. Il emploie 702 éléments et 2222 nœuds. Quant à la figure (5.4.b), elle montre un modèle comportant 338 éléments et 2564 nœuds, utilisé pour discrétiser l'une des sections en T étudiées.

A ce niveau, il faut noter que la méthode développée dans ce mémoire est nommée « MEF-RNA » dans les légendes des figures présentant les résultats comparatifs. Les résultats issus de CAST3M sont labellisés « CAST3M (MEF 3D) ». Dans toutes les figures on trouve également des courbes « Elastique » calculées à partir de la théorie élastique linéaire de flexion des poutres. En fonction de leur disponibilité, on peut aussi trouver dans ces figures les résultats obtenus dans le mémoire de DEA qui a inspiré ce travail. Nous rappelons qu'on y a fait usage de la méthode des différences finies (MDF) pour résoudre les équations

différentielles de la théorie de flexion des poutres de Bernoulli. La relation constitutive non linéaire est quant à elle introduite de deux façons : en résolvant numériquement les équations paramétriques présentées au chapitre 3, ou bien en utilisant les réseaux de neurones artificiels. Les labels utilisés pour les résultats correspondants sont « MDF-Equ.Param » et « MDF-RNA », respectivement.



(a) Modele 3D d'une poutre a section triangulaire



(b) Modele 3D d'une poutre a section en T

Figure 5.4 : Modèles tridimensionnels sur CAST3M

5.3 Section triangulaire

5.3.1 Calculs préliminaires

Pour les sections triangulaires, l'expression du moment fléchissant en fonction de sa variable sans dimension associée est donnée, d'après les équations (3.46) et (3.55), par :

$$M = \frac{\sigma_p b h^2}{3\left(2+\sqrt{2}\right)} m$$

La valeur du moment indiquant la fin de la phase plastique, noté M_e, s'écrit comme suit, compte tenu de la relation (3.60) :

$$M_e = \frac{\sigma_p b h^2}{3\left(2+\sqrt{2}\right)} m_{01} = \frac{\sigma_p b h^2}{24}$$

En utilisant la relation (3.72), le moment à la rupture, noté M_u, est donné par :

$$M_u = \frac{\sigma_p b h^2}{3\left(2+\sqrt{2}\right)} m_{ult} = \frac{\sigma_p b h^2}{3}\left(1 - 3\sqrt{\frac{3}{2}} \times \frac{p\left(p^2+1\right)}{\left(3p^2+1\right)^{\frac{3}{2}}}\right)$$

FICHIER de DONNÉES	N°Ligne	COMMENTAIRES
Exemple 1	1	*Titre de l'exemple*
21 20 1 1 5 2 11 2 2	2	*NPoin,NElem,NBoun,NMats,NProp,NNodE,NIncs,NAlgo,NDofN*
1 0.1 0.1 2.1E8 1.3125E8 210E3	3	*Jmats,Props[Jmats,1..5] : b,h,E,G,Sig0*
1 1 2 1	4	*Jelem,Lnods[Jelem,1],Lnods[Jelem,2],MatNo[Jelem]*
2 2 3 1	5	
3 3 4 1	6	
4 4 5 1	7	
5 5 6 1	8	
6 6 7 1	9	
7 7 8 1	10	
8 8 9 1	11	
9 9 10 1	12	
10 10 11 1	13	
11 11 12 1	14	
12 12 13 1	15	
13 13 14 1	16	
14 14 15 1	17	
15 15 16 1	18	
16 16 17 1	19	
17 17 18 1	20	
18 18 19 1	21	
19 19 20 1	22	
20 20 21 1	23	
1 0.00	24	*Jpoin,coord[Jpoin]*
2 0.05	25	
3 0.10	26	
4 0.15	27	
5 0.20	28	
6 0.25	29	
7 0.30	30	
8 0.35	31	
9 0.40	32	
10 0.45	33	
11 0.50	34	
12 0.55	35	
13 0.60	36	
14 0.65	37	
15 0.70	38	
16 0.75	39	
17 0.80	40	
18 0.85	41	
19 0.90	42	
20 0.95	43	
21 1.00	44	
1 1 0.0 1 0.0	45	*NodFx,icode[DDL1],Value[DDL1],icode[DDL2],value[DDL2]...*
2 0.0 0.0 0.0 0.0	46	*Jelem,RLoad[Jelem,1],...,RLoad[Jelem,NEvab]*
20 0.0 0.0 0.0 1.0	47	*Moment unitaire a l'extremite de l'element N°20*
15 1 8.74 0.05	48	*NIter,NOutP,Facto,Toler*
15 1 1.00 0.05	49	
15 1 1.00 0.05	50	
15 1 1.00 0.05	51	
15 1 1.00 0.05	52	
15 1 1.00 0.05	53	
15 1 1.00 0.05	54	
15 1 1.00 0.05	55	
15 1 1.00 0.05	56	
15 1 1.00 0.05	57	
15 1 1.00 0.05	58	*Rem: La somme des Facto = 18,74*

Tableau 5.1 : Exemple de fichier de données

Pour le matériau choisi (équations 3.27 à 3.30) et la géométrie de la poutre retenue (figure 5.2), on évalue :

$$I = \frac{bh^3}{36} = 2{,}778 \cdot 10^{-6}\ \text{m}^4\ ;\ M_e = 8{,}75\ \text{KN.m}\ ;\ M_u = 20{,}44\ \text{KN.m}$$

5.3.2 Résultats et commentaires

Pour valider le programme élaboré dans le cas des poutres à section triangulaires, nous avons retenu cinq schémas statiques à présenter dans ce mémoire. A celui illustré dans la figure (5.1.a), nous avons appliqué un moment concentré maximum de 18,74 KN.m, ce qui représente 92% du moment ultime. La figure (5.5) donne l'évolution de la flèche à l'extrémité libre de la console en fonction de la charge. La déformée induite par le moment maximum est, quant à elle, dessinée dans la figure (5.6.a). La valeur de la flèche maximum obtenue par le programme élaboré est de 2,6273 cm (voir tableau 5.2). Celle calculée par le logiciel CAST3M est égale à 2,6679 cm. Ce qui représente une différence de 1,52%. La méthode des différences finies utilisant les expressions exactes de la relation constitutive a donné un résultat de 2,6254 cm alors que celle utilisant les réseaux de neurones n'a prédit qu'une flèche de 2,1926 cm, soit des différences respectives de 1,59% et 17,82% par rapport au résultat de CAST3M.

Les quatre autres cas étudiés sont ceux illustrés par les figures (5.1.b), (5.1.c), (5.1.d) et (5.1.g). Selon les schémas statiques, la charge (force ou moment) appliquée est celle qui provoque un moment fléchissant maximum de 18,74 KN.m. Les déformées des poutres à sections triangulaires étudiées sont illustrées par la figure (5.6). Le tableau 5.2, qui peut être considéré comme un récapitulatif des résultats obtenus, donne les flèches maxima obtenues par les diverses méthodes pour les différents cas étudiés. En dessous des valeurs, nous donnons, en italique, l'erreur de chaque résultat par rapport à celui calculé par CAST3M.

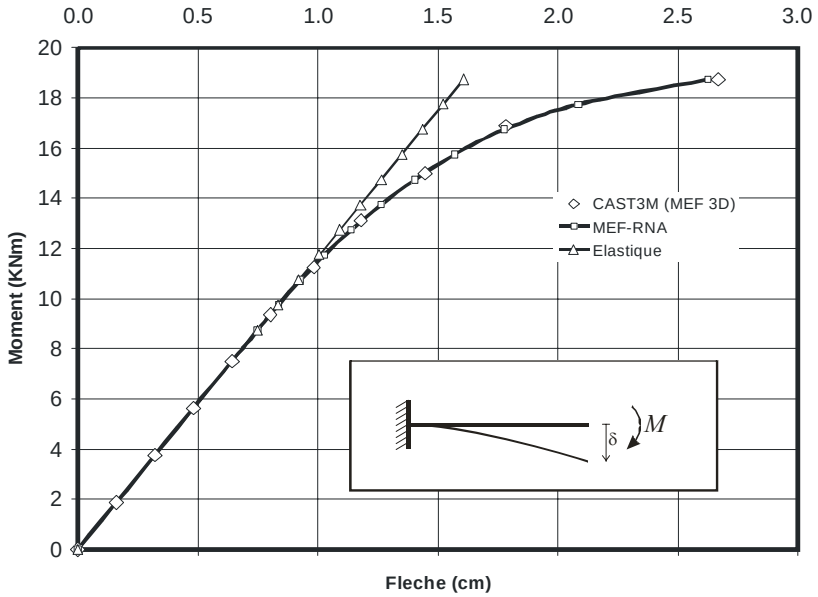


Figure 5.5 : Flèche à l'extrémité libre de la console en fonction du moment appliqué.

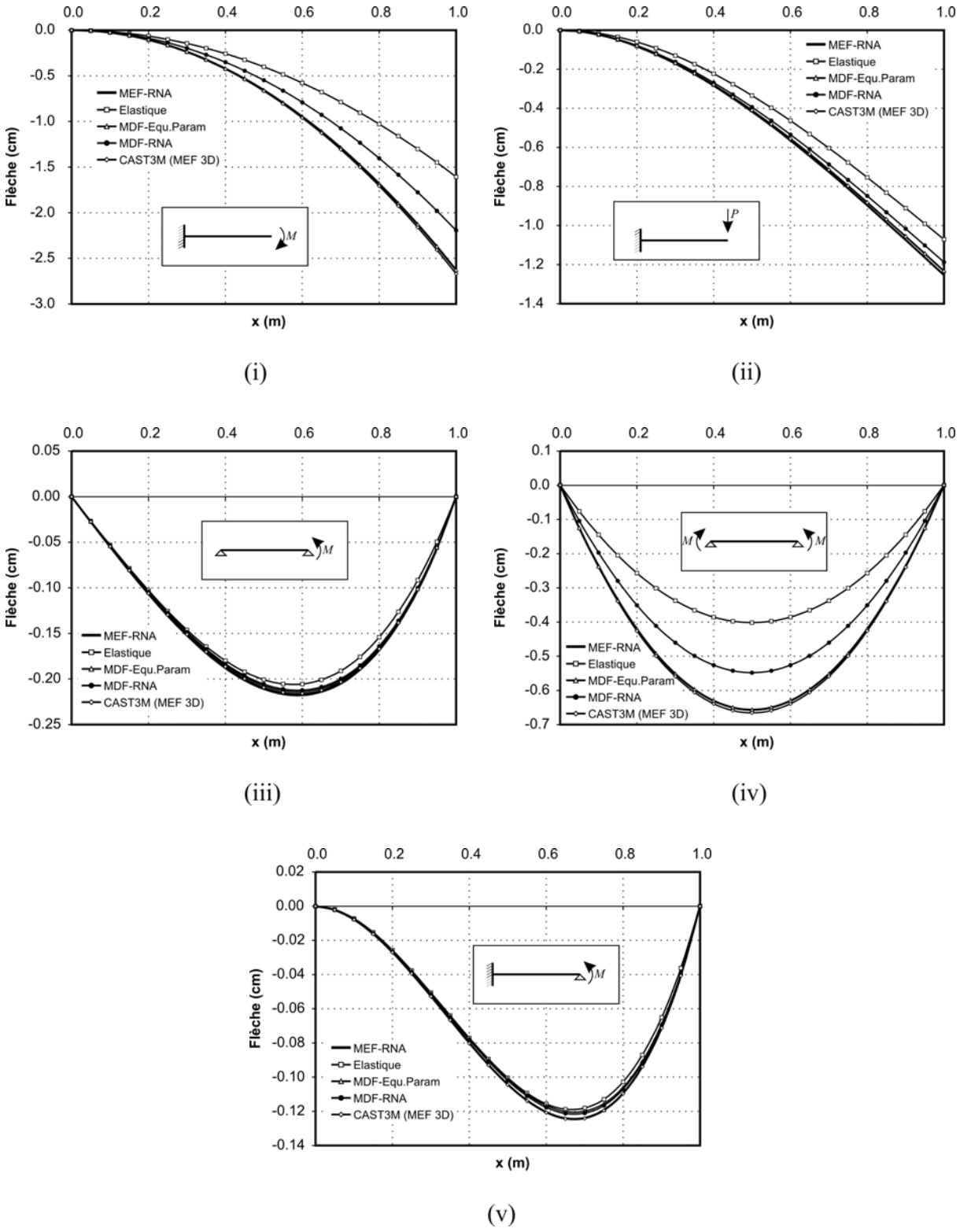


Figure 5.6 : Tracés des déformées des poutres à section triangulaire étudiées.

Schéma Statique Figure N°	Charge Appliquée	**Flèche maximum** (cm)				
		MEF-RNA	Elastique	MDF. Equ.Param	MDF. RNA	**CAST3M. MEF.3D**
(5.1.a)	*M*=18.74 KN.m	2.6273 *1.52%*	1.6063 *39.79%*	2.6254 *1.59%*	2.1926 *17.82%*	**2.6679**
(5.1.b)	*P*=18.74 KN	1.2541 *1.57%*	1.0709 *13.27%*	1.2317 *0.24%*	1.1872 *3.85%*	**1.2347**
(5.1.c)	*M*=18.74 KN.m	0.2177 *0.79%*	0.2056 *4.81%*	0.2140 *0.93%*	0.2123 *1.71%*	**0.2160**
(5.1.d)	*M*=18.74 KN.m	0.6568 *1.35%*	0.4016 *39.68%*	0.6562 *1.44%*	0.5482 *17.66%*	**0.6658**
(5.1.g)	*M*=18.74 KN.m	0.1241 *0.24%*	0.1188 *4.50%*	0.1201 *3.46%*	0.1213 *2.49%*	**0.1244**

Tableau 5.2 : Récapitulatif des flèches maxima pour les sections triangulaires

Les résultats obtenus à partir du programme élaboré sont, pour les cinq cas, très proches de ceux donnés par le logiciel CAST3M. La différence est restée inférieure à 1,57%. La stratégie retenue pour interpoler la relation constitutive non linéaire à l'aide des réseaux de neurones artificiels semble efficace. Nous sommes loin des résultats très dispersés trouvés dans la référence [4].

Dans ce qui suit, nous allons présenter les résultats calculés pour des poutres ayant diverses formes de sections en T.

5.4 Sections en T

5.4.1 Calculs préliminaires

Pour les sections en T, l'expression du moment d'inertie *I* est donnée par l'équation (3.81), rappelée ci-dessous :

$$I = \frac{bh^3}{12}\frac{4\Sigma\Pi - 3\Gamma^2}{\Sigma}$$

D'après les équations (3.46), (3.86), (3.88) et (3.90), l'expression du moment fléchissant *M* en fonction de sa variable sans dimension associée *m* est donnée par :

$$M = \frac{\sigma_p bh^2}{4\beta} m$$

Il s'en suit que la valeur du moment indiquant la fin de la phase plastique, noté M_e, s'écrit comme suit :

$$M_e = \frac{\sigma_p b h^2}{4\beta} m_{01}$$

Le moment à la ruine de la section, noté M_u, est donné par :

$$M_u = \frac{\sigma_p b h^2}{4\beta} m_{ult}$$

Où les expressions de β, m_{01} et m_{ult} dépendent du type de section en T (voir le paragraphe 5 du chapitre 3).

Pour le matériau choisi (équations 3.27 à 3.30) et les géométries des sections en T retenues (figure 5.3), nous évaluons l'inertie I, la limite élastique M_e et le moment ultime M_u. Le résultat de ce calcul est résumé dans ce qui suit :

- La section de Type C1 :

$$I = 4{,}8408\ 10^{-7}\ \text{m}^4\ ;\ M_e = 2{,}0512\ \text{KN.m}\ ;\ M_u = 3{,}6806\ \text{KN.m}$$

- La section de Type A1 :

$$I = 6{,}4187\ 10^{-8}\ \text{m}^4\ ;\ M_e = 0{,}4264\ \text{KN.m}\ ;\ M_u = 0{,}7706\ \text{KN.m.}$$

- La section de Type B :

$$I = 7{,}7709\ 10^{-8}\ \text{m}^4\ ;\ M_e = 0{,}5222\ \text{KN.m}\ ;\ M_u = 0{,}9421\ \text{KN.m}$$

- La section de Type C2 :

$$I = 1{,}3064\ 10^{-8}\ \text{m}^4\ ;\ M_e = 0{,}1482\ \text{KN.m}\ ;\ M_u = 0{,}2703\ \text{KN.m}$$

- La section de Type A2 :

$$I = 1{,}6833\ 10^{-7}\ \text{m}^4\ ;\ M_e = 0{,}8890\ \text{KN.m}\ ;\ M_u = 1{,}5953\ \text{KN.m}$$

5.4.2 Le problème rencontré et la correction de la stratégie

Dans le cas des sections en T, le programme réalisé selon l'idée originale n'a pas fonctionné. L'algorithme incrémental divergeait ou bien les résultats obtenus après atteinte du nombre maximum d'itérations étaient erronés car trop éloignés des prédictions des autres méthodes. Paradoxalement, nous avons observé que les résultats récupérés suite à l'arrêt du programme pour divergence de l'algorithme étaient plus proches de ceux attendus et étaient

donc « meilleurs » que ceux obtenus lorsque le programme semblait converger en continuant à itérer.

Nous expliquons ceci par le fait que, dans la stratégie proposée au paragraphe 4.3.3, nous utilisons deux réseaux de neurones pour effectuer l'aller-retour sur la courbe *M*-χ. Ces deux réseaux ne fournissent qu'une approximation de la relation exacte et comportent donc une erreur. C'est justement cette différence « fixe » entre les deux « courbes » approchées qui est à l'origine du problème. Elle est en effet toujours présente et ne peut jamais être réduite par itération. Dans le programme élaboré et pour arrêter l'algorithme itératif nous nous servons d'un ratio calculé à partir des différences entre les résultats de deux itérations successives. Tant que ce ratio diminue, même légèrement, nous continuons à itérer jusqu'à ce qu'il soit inférieur à la tolérance requise ou que le nombre maximum d'itérations soit atteint. Nous arrêtons par contre immédiatement le processus si ce ratio tend à augmenter ; nous disons alors que l'algorithme diverge.

Sachant que l'algorithme itératif cumule les vecteurs des résidus, l'erreur « fixe », relativement petite mais toujours présente, entre les deux courbes s'additionne donc. Ainsi, plus nous itérons plus nous nous éloignons de la solution. Ceci explique pourquoi les résultats obtenus suite à un arrêt prématuré du programme pour divergence de l'algorithme étaient, contre toute attente, « meilleurs ».

La stratégie en question a fonctionné pour le cas des poutres à sections triangulaires car, la relation constitutive étant simple, l'erreur commise par les deux réseaux de neurones d'interpolation était beaucoup plus petite que dans le cas des sections en T. Il suffit pour cela d'observer les courbes données par les figures (4.5) et (4.8). L'erreur d'interpolation est alors suffisamment petite pour ne pas entraver la convergence du processus itératif.

La solution a été de « corriger » la stratégie proposée et de ne faire appel qu'à un seul réseau pour interpoler la relation moment - courbure. Nous avons alors décidé d'utiliser exclusivement le réseau assurant l'interpolation :

$$m = f_2(k)$$

Pour évaluer $m^{(r)}$ en fonction de $m^{(r-1)}$ et Δk (voir paragraphe 4.3.3), nous commençons par calculer $k^{(r-1)}$ à partir de $m^{(r-1)}$. Nous utilisons alors le réseau de neurones retenu et un algorithme de dichotomie pour trouver la racine de l'équation :

$$f_2\left(k^{(r-1)}\right) - m^{(r-1)} = 0$$

Nous rajoutons ensuite à cette courbure, $k^{(r-1)}$, l'incrément de courbure Δk pour obtenir $k^{(r)}$. Enfin, nous utilisons le réseau de neurones afin de calculer la valeur recherchée $m^{(r)} = f_2\left(k^{(r)}\right)$.

Une fois cette nouvelle stratégie programmée, nous avons pu étudier la flexion simple non linéaire des poutres ayant des sections en T en utilisant les réseaux de neurones artificiels pour interpoler la relation constitutive. Les résultats obtenus sont donnés et commentés dans les paragraphes suivants. Comme il existe cinq types de sections en T (C1, A1, B, C2 et A2), nous présentons l'étude d'une section par type. Pour chacune d'elles nous considérons divers schémas statiques.

5.4.3 Section en T de Type C1

La section représentant ce type de formes en T est montrée par la figure (5.3). Les six schémas statiques étudiés sont ceux illustrés par les figures (5.1.a), (5.1.b), (5.1.d), (5.1.e), (5.1.f) et (5.1.h). Selon les cas, la charge (force ou moment) appliquée est celle qui provoque un moment fléchissant maximum de 3,3 KN.m, ce qui représente 89,7% du moment ultime de la section.

Les déformées des poutres à section en T de type C1 étudiées sont illustrées par la figure (5.7). Le tableau 5.3, qui peut être considéré comme un récapitulatif des résultats obtenus, donne les flèches maxima obtenues par les diverses méthodes pour les différents cas étudiés. En dessous des valeurs, nous donnons, en italique, le pourcentage de différence de chaque résultat par rapport à celui calculé par CAST3M.

Les résultats du programme sont proches de ceux donnés par le logiciel CAST3M. La différence maximum obtenue est de seulement 2,34%. L'autre étude qui utilise également les réseaux de neurones pour interpoler la relation moment - courbure a eu jusqu'à 23,7% de différence, soit environ une erreur dix fois plus grande. Nous pouvons alors avancer que le programme semble bien fonctionner.

D'un autre côté et afin de mettre en évidence le travail de modélisation effectué sur CAST3M, nous présentons le dessin d'une poutre déformée. Nous avons retenu comme exemple le modèle correspondant au schéma statique (5.1.h). La figure (5.8) donne, en perspective, la déformée de la poutre en question. La figure (5.9) donne la distribution de la flèche sous la forme d'un réseau de courbes isovaleurs en remplissant par des couleurs différentes les zones contenant des valeurs différentes.

Schéma Statique Figure N°	Charge Appliquée	**Flèche maximum** (cm)				
		MEF-RNA	Elastique	MDF. Equ.Param	MDF. RNA	**CAST3M. MEF.3D**
(5.1.a)	*M*=3.3 KN.m	2.7915 *2.34%*	1.6230 *40.50%*	2.7437 *0.59%*	2.0843 *23.59%*	**2.7277**
(5.1.b)	*P*=3.3 KN	1.2758 *0.67%*	1.0820 *14.62%*	1.2433 *1.89%*	1.1976 *5.50%*	**1.2673**
(5.1.d)	*M*=3.3 KN.m	0.6979 *2.21%*	0.4058 *40.57%*	0.6859 *0.46%*	0.5211 *23.68%*	**0.6828**
(5.1.e)	*P*=13.2 KN	0.3208 *0.98%*	0.2705 *16.51%*	0.3018 *6.86%*	0.2958 *8.72%*	**0.3240**
(5.1.f)	*P*=17.65 KN	0.2896 *1.17%*	0.2526 *13.79%*	0.2860 *2.39%*	0.2830 *3.42%*	**0.2930**
(5.1.h)	*q*=26.4 KN/m	0.1496 *2.25%*	0.1399 *8.57%*	0.1480 *3.30%*	0.1434 *6.31%*	**0.1530**

Tableau 5.3 : Récapitulatif des flèches maxima pour les sections en T type C1.

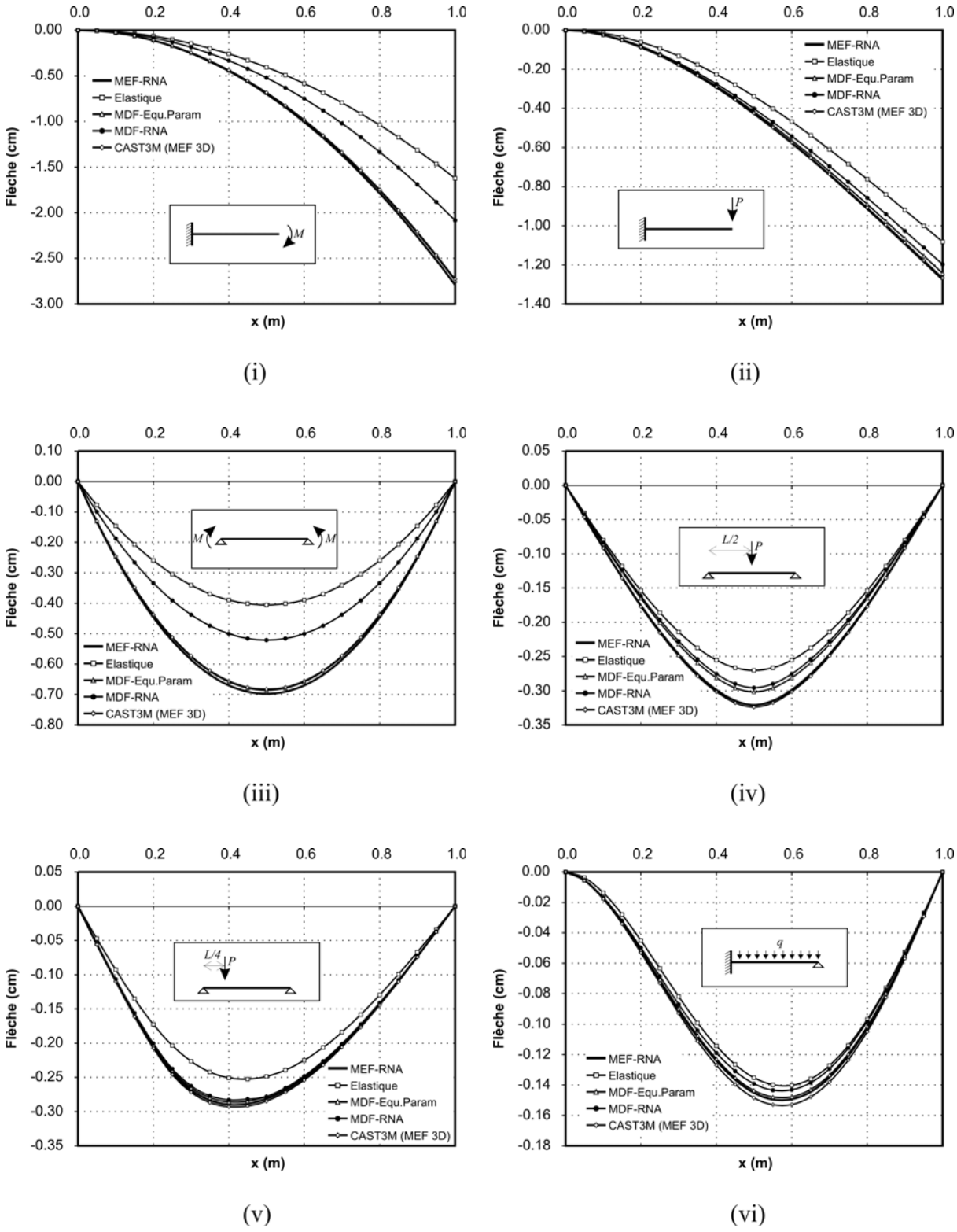


Figure 5.7 : Tracés des déformées des poutres à section en T de type C1.

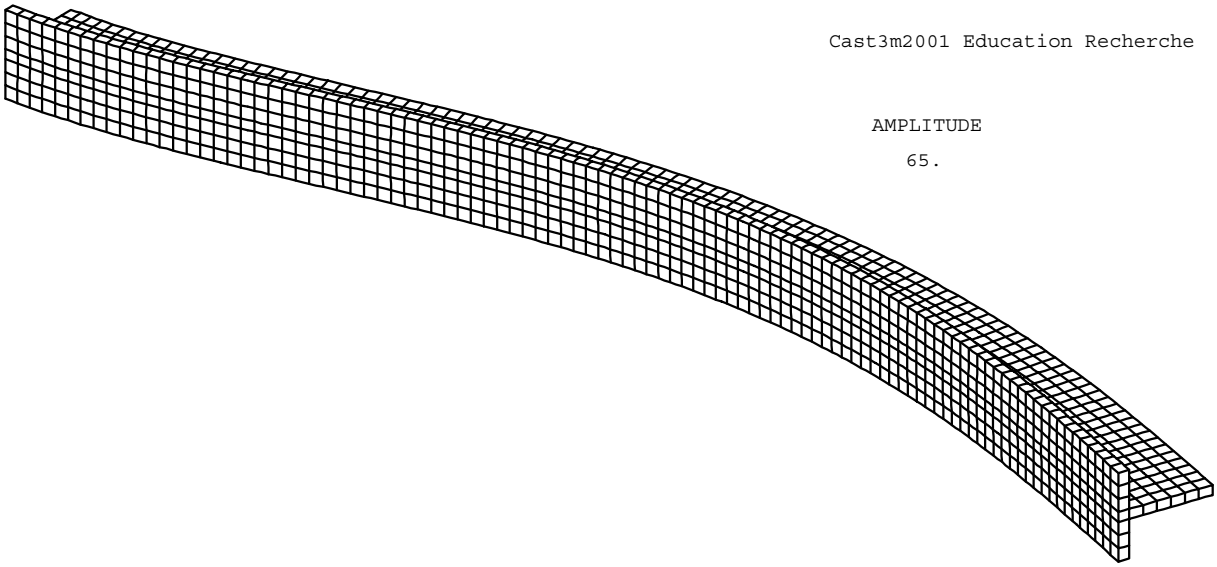


Figure 5.8 : Déformée de la poutre (5.1.h) obtenue dans CAST3M

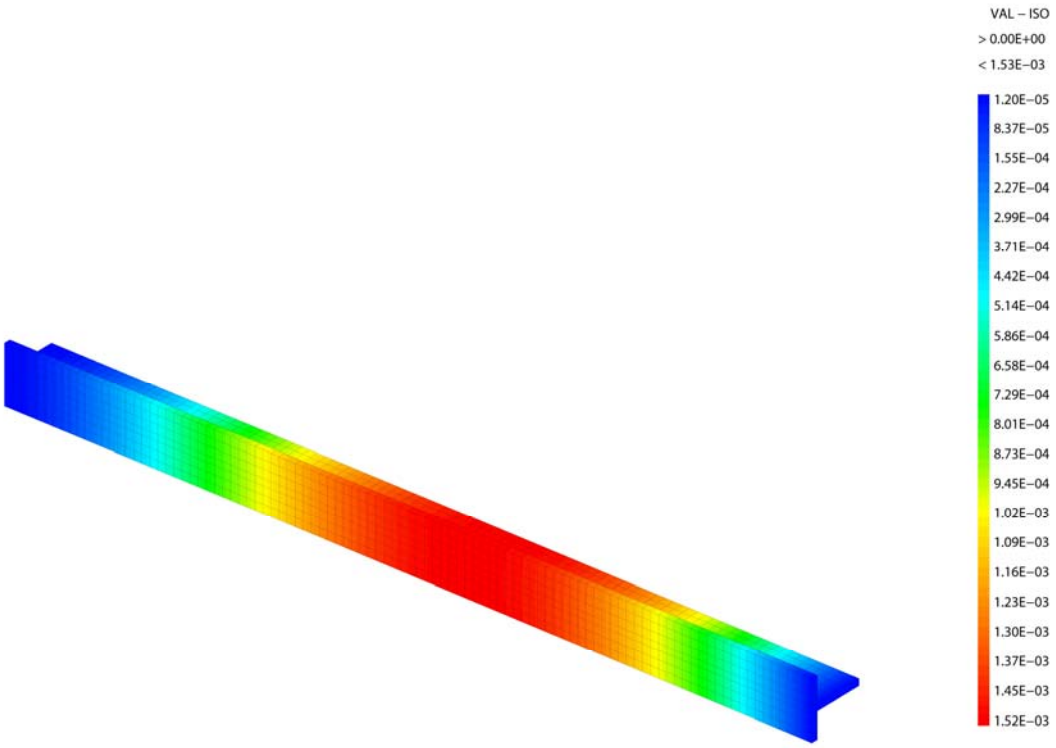


Figure 5.9 : La flèche de la poutre (5.1.h) obtenue dans CAST3M

5.4.4 Section en T de type A1

Pour valider le programme élaboré dans le cas des poutres à section triangulaires de type A1, nous avons retenu trois schémas statiques à mettre dans ce mémoire. Il s'agit de ceux illustrés par les figures (5.1.a), (5.1.b) et (5.1.d). Au cas de la figure (5.1.a), nous avons appliqué un moment concentré maximum de 0,694 KN.m, ce qui représente 90% du moment ultime de la section. La déformée induite par ce moment est dessinée dans la figure (5.10.i). La valeur de la flèche maximum obtenue par le programme élaboré est de 4.3912 cm. Celle calculée par le logiciel CAST3M est égale à 4.3826 cm. Ce qui représente une différence de 0,20%. La méthode des différences finies utilisant les réseaux de neurones n'a prédit qu'une flèche de 3.3309 cm, soit une différence de 24% par rapport au résultat de CAST3M.

La force appliquée à la console (5.1.b) est de 0,694 KN. Le moment exercé aux extrémités de la poutre simplement appuyée (5.1.d) est égal à 0,694 KN.m. Les déformée correspondantes sont dessinées aux figures (5.10.ii) et (5.10.iii), respectivement. Le tableau 5.4 récapitule les résultats obtenus. Il donne les flèches maxima obtenues par les diverses méthodes pour les différents cas étudiés. Les valeurs en italique sont le pourcentage de différence de chaque résultat par rapport à celui calculée par CAST3M.

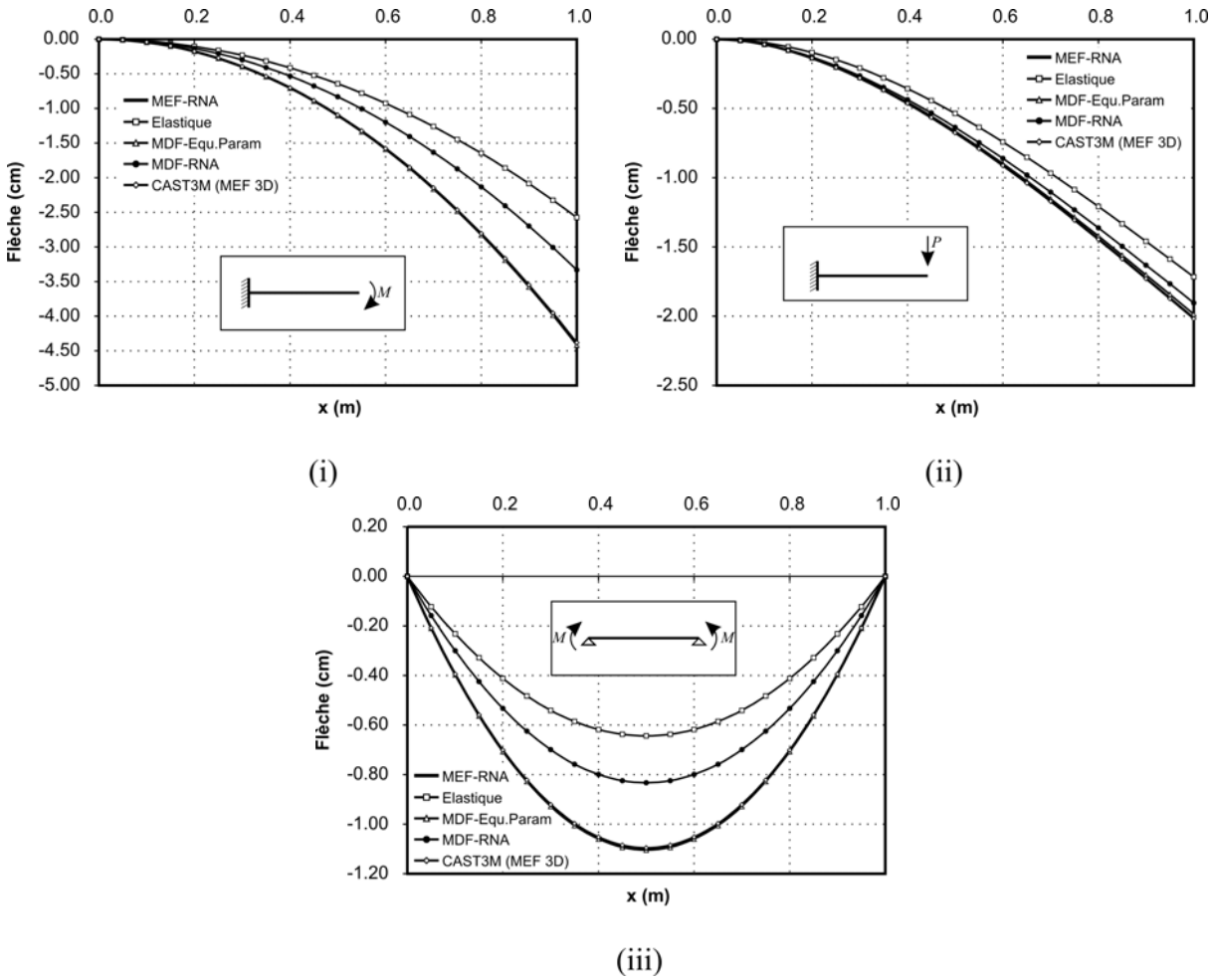


Figure 5.10 : Tracés des déformées des poutres à section en T de type A1.

Schéma Statique Figure N°	Charge Appliquée	**Flèche maximum** (cm)				
		MEF-RNA	Elastique	MDF. Equ.Param	MDF. RNA	**CAST3M. MEF.3D**
(5.1.a)	*M*=0.694 KN.m	4.3912 *0.20%*	2.5742 *41.26%*	4.4184 *0.82%*	3.3309 *24.00%*	**4.3826**
(5.1.b)	P=0.694 KN	2.0152 *0.15%*	1.7161 *14.96%*	1.9853 *1.63%*	1.9036 *5.68%*	**2.0181**
(5.1.d)	*M*=0.694 KN.m	1.0978 *0.18%*	0.6436 *41.27%*	1.1046 *0.80%*	0.8327 *24.01%*	**1.0958**

Tableau 5.4 : Récapitulatif des flèches maxima pour les poutres en T type A1.

5.4.5 Section en T de Type B

Les trois types de sections en T qui restent à présenter dans ce mémoire, à savoir les types B C2 et A2, n'ont pas été étudiés dans la référence [4]. Nous pensons que les résultats correspondant ont du être trop éloignés de ceux attendus. Effectivement, leurs relations moment - courbure diffèrent beaucoup de celle qui a été retenue comme courbe « moyenne » pour résumer le comportement des sections en T et qui a donc été « apprise » par les réseaux de neurones (voir le paragraphe 4.5). C'est ce qui a probablement motivé la conclusion de l'auteur de ce travail sur « la grande dispersion des résultats des réseaux de neurones » et ne lui a pas permis de prouver l'efficacité des réseaux neuromimétiques à interpoler la relation constitutive des poutres en flexion.

Les seuls résultats qui vont figurer dans ce qui suit sont ceux obtenus par le programme élaboré dans le cadre de ce travail, ceux donnés par la théorie élastique de flexion des poutres et ceux calculés à l'aide du logiciel CAST3M.

Pour ce qui est des exemples des poutres à section en T de type B, nous avons choisi trois schémas statiques à présenter dans ce mémoire. Il s'agit de ceux illustrés par les figures (5.1.a), (5.1.e) et (5.1.h). A la poutre montrée par la figure (5.1.a), nous avons appliqué un moment concentré maximum de 0,85 KN.m, ce qui représente environ 90% du moment ultime de la section. La déformée résultant de l'application de ce moment est dessinée dans la figure (5.11.i). La valeur de la flèche maximum obtenue par le programme élaboré est de 4,4108 cm. Celle calculée par le logiciel CAST3M est égale à 4,4119 cm. Nous avons ainsi une très petite différence de 0,03%.

La force concentrée appliquée à mi-portée de la poutre simplement appuyée (5.1.e) est de 3,4 KN. La charge uniformément répartie appliquée à la poutre hyperstatique encastrée - simplement appuyée illustrée à la figure (5.1.h) est de 6,8 KN/m. Les déformées correspondantes sont dessinées aux figures (5.11.ii) et (5.11.iii), respectivement. Le tableau 5.5 récapitule les flèches maxima obtenues par les diverses méthodes pour les différents cas étudiés. Nous rappelons que les valeurs en italique sont le pourcentage de différence de chaque résultat par rapport à celui prédit par CAST3M. La différence maximum obtenue est de seulement 1,55%.

D'un autre côté et afin de présenter une forme de résultat faisant intervenir l'état de contrainte, on a choisi de mettre en évidence la redistribution des efforts internes suite à la plastification d'une partie de la poutre. Nous représentons alors dans la figure (5.12) les diagrammes, dans les cas élastique et anélastique, du moment fléchissant de la poutre hyperstatique, encastrée - simplement appuyée illustrée à la figure (5.1.h), sous des forces réparties de 7.5 KN/m et de 8 KN/m.

L'expression du moment fléchissant calculée par la théorie de flexion élastique des poutres de Bernoulli est :

$$M(x) = -\frac{1}{8}q\left(L^2 - 5Lx + 4x^2\right) \qquad \forall\, x \in [0, L]$$

Nous notons évidemment la proportionnalité moment fléchissant - charge. Les moments maxima positif et négatif augmenteraient dans les mêmes proportions en cas d'augmentation de la charge. D'autre part, il parait évident qu'en élasticité linéaire, quelque soit q, le moment s'annule à la même position x_0.

En observant maintenant les résultats fournis par le programme élaboré, nous remarquons que le moment négatif, le plus grand en valeur absolue dans cet exemple, augmente moins vite que le moment positif suite à l'augmentation de q de 7,5 KN/m à 8 KN/m. D'un autre côté, ce changement de la charge a entraîné un décalage de l'abscisse du moment nul vers l'origine des axes, c'est-à-dire vers l'encastrement.

Autrement dit, le moment à l'encastrement calculé en élasto-plasticité est plus petit, en valeur absolue, que celui donné par la théorie élastique linéaire. Par contre le moment élasto-plastique positif, c'est-à-dire en travée, est plus grand que celui prédit par l'élasticité linéaire. C'est la redistribution des contraintes suite à la plastification de la section située à l'encastrement.

Schéma Statique Figure N°	Charge Appliquée	**Flèche maximum** (cm)		
		MEF-RNA	Elastique	**CAST3M. MEF.3D**
(5.1.a)	M=0.85 KN.m	4.4108 *0.03%*	2.6043 *40.97%*	**4.4119**
(5.1.e)	P=3.4 KN	0.5079 *1.27%*	0.4341 *15.63%*	**0.5145**
(5.1.h)	q=6.8 KN/m	0.2333 *1.55%*	0.2245 *5.24%*	**0.2369**

Tableau 5.5 : Récapitulatif des flèches maxima pour les sections en T type B.

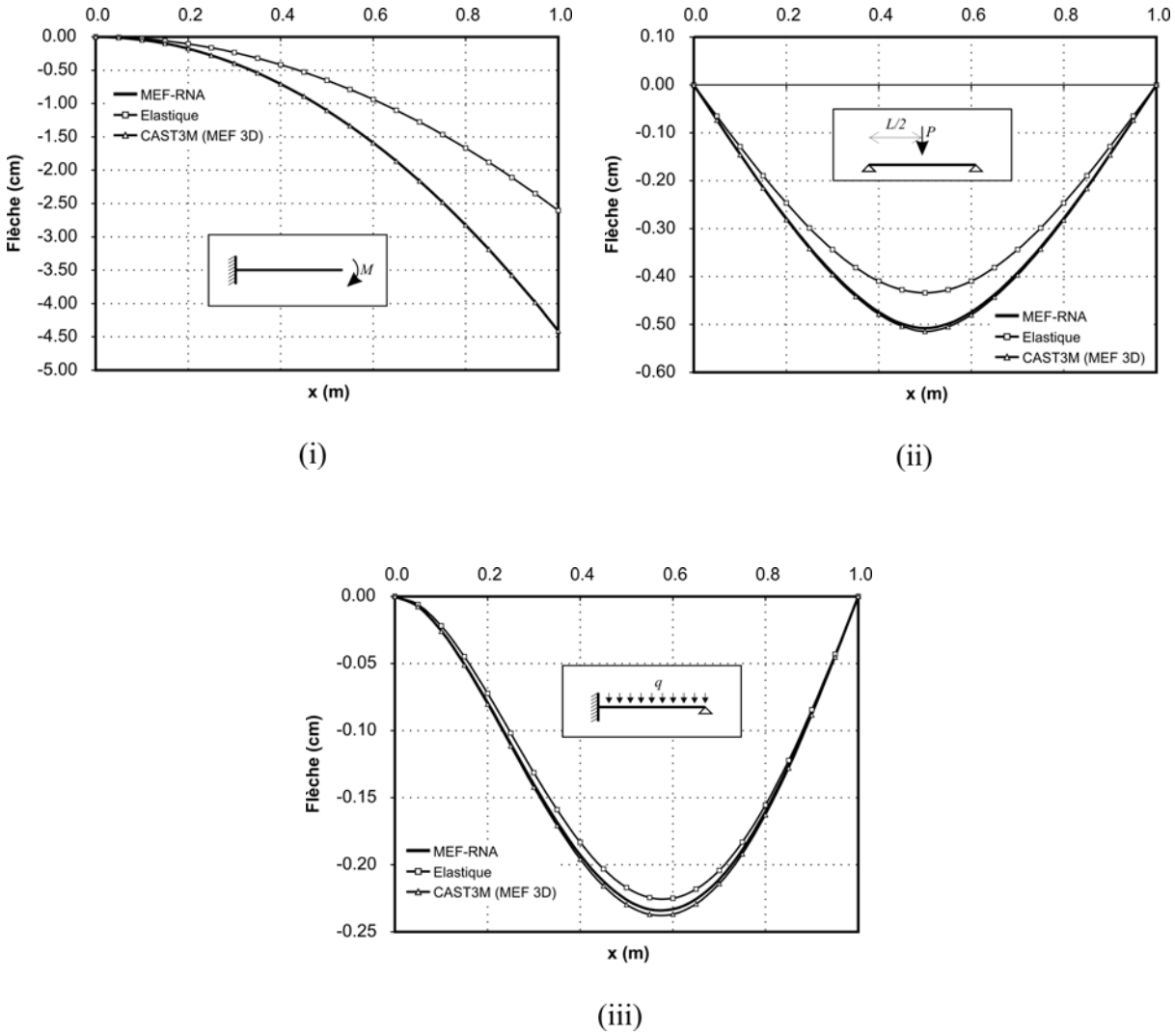


Figure 5.11 : Tracés des déformées des poutres à section en T type B.

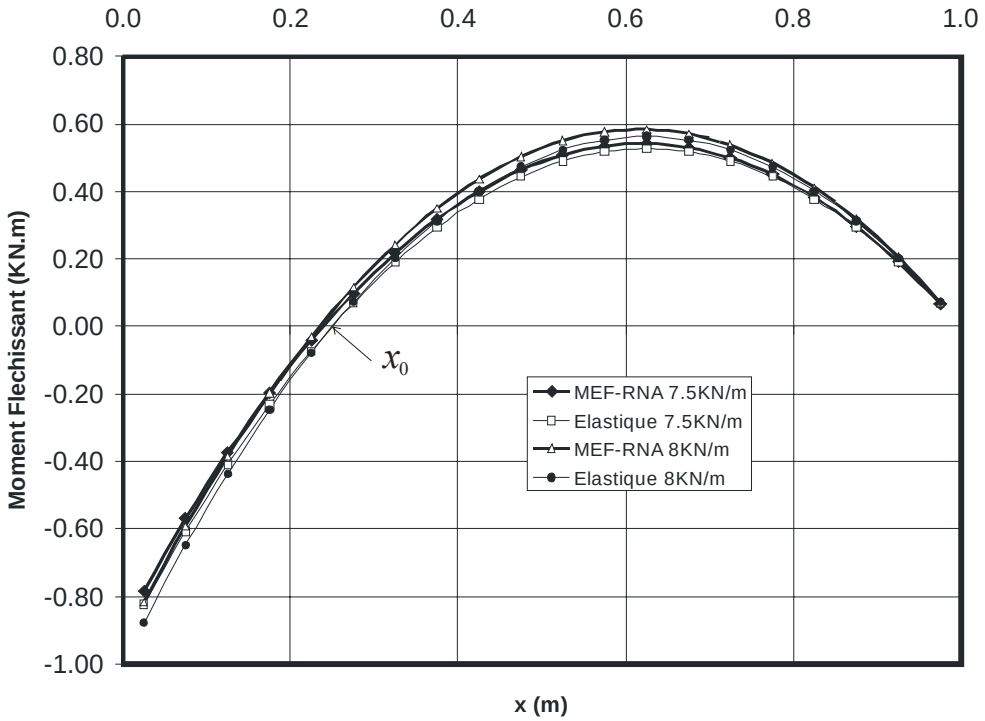


Figure 5.12 : Diagrammes du moment fléchissant.

5.4.6 Section en T de Type C2

Pour étudier les poutres à section en T de type C2, nous avons retenu trois schémas statiques. Il s'agit de ceux illustrés par les figures (5.1.b), (5.1.d) et (5.1.f). Au cas de la figure (5.1.b), nous avons appliqué une force concentrée maximum de 0,243 KN, ce qui représente 90% du moment ultime de la section. La déformée induite par ce moment est dessinée dans la figure (5.13.i). La valeur de la flèche maximum obtenue par le programme élaboré est de 3.6821 cm. Celle calculée par le logiciel CAST3M est égale à 3**.**6654 cm. Ce qui représente une différence de 0.45%.

Le moment exercé aux extrémités de la poutre simplement appuyée (5.1.d) est égal à 0.243 KN.m. La force excentrée exercée à la poutre simplement appuyée de la figure (5.1.f) est égale à 1.296 KN. Les déformée correspondantes sont dessinées aux figures (5.13.ii) et (5.13.iii), respectivement. Le tableau 5.6 récapitule les flèches maxima obtenues par les diverses méthodes pour les différents cas étudiés. Le pourcentage de différence de chaque résultat par rapport à celui prédit par CAST3M est écrit en italique. Il vaut 0.96% pour le cas de la figure (5.1.d) et 1.20% pour le cas de la figure (5.1.f). La différence maximum obtenue est de 1.20%.

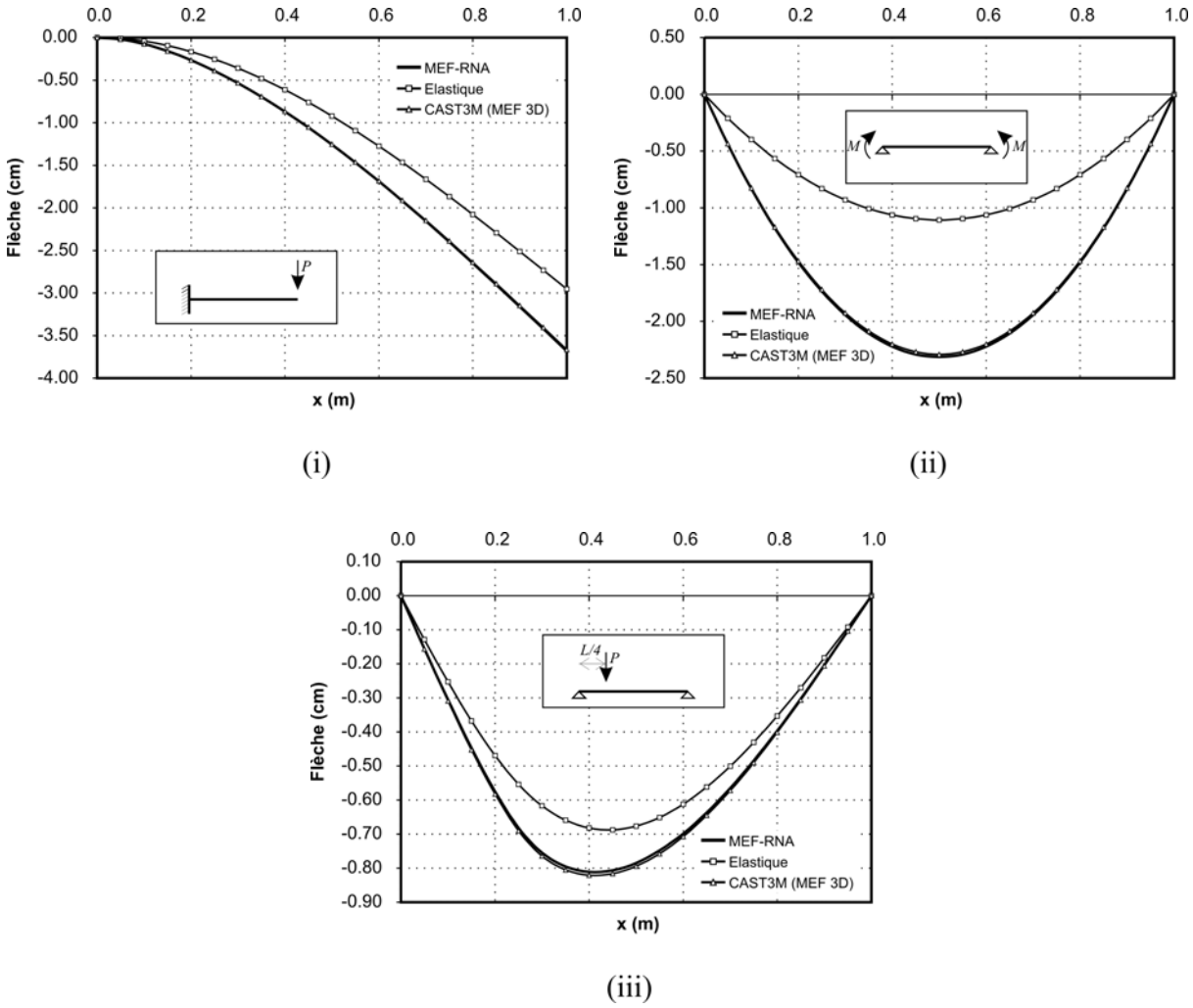


Figure 5.13 : Tracés des déformées des poutres à section en T type C2.

Schéma Statique Figure N°	Charge Appliquée	**Flèche maximum** (cm)		
		MEF-RNA	Elastique	**CAST3M. MEF.3D**
(5.1.b)	*P*=0.243 KN	3.6821 *0.45%*	2.9525 *19.45%*	**3.6654**
(5.1.d)	*M*=0.243 KN.m	2.3138 *0.96%*	1.1072 *51.69%*	**2.2918**
(5.1.f)	*P*=1.296 KN	0.8108 *1.20%*	0.6820 *16.89%*	**0.8207**

Tableau 5.6 : Récapitulatif des flèches maxima pour les sections en T type C2.

5.4.7 Section en T de Type A2

La section représentant ce type de formes en T est elle aussi montrée par la figure (5.3). Les trois schémas statiques illustrés par les figures (5.1.a), (5.1.d) et (5.1.e) sont retenus. A la poutre montrée par la figure (5.1.a), nous avons appliqué un moment concentré maximum de 1.436 KN.m, ce qui représente environ 90% du moment ultime de la section. La figure (5.14.i) donne la déformée induite par l'application de ce moment. La valeur de la flèche maximum obtenue par le programme élaboré est de 3.1071 cm. Celle calculée par le logiciel CAST3M est égale à 3.1365 cm. Nous avons ainsi une différence de 0.94%.

Les deux autres cas étudiés sont illustrés par les figures (5.1.d) et (5.1.e). Les déformées correspondantes sont dessinées respectivement dans les figures (5.14.ii) et (5.14.iii).

Le moment appliqué aux extrémités de la poutre simplement appuyée est de 1.436 KN.m. La valeur de la flèche maximum obtenue par le programme élaborée est égale a 0.7768 cm et celle calculée par CAST3M vaut 0.7845 cm. Le pourcentage de différence est donc de 0.99%.

La force concentrée appliquée à mi-travée de la poutre simplement appuyée du dernier cas traité est de 5.744 KN.m. La valeur de la flèche maximum obtenue par le programme élaborée est égale a 0.3878 cm et celle calculée par CAST3M vaut 0.3929 cm. La différence obtenue est de 1,31%. Sur le tableau 5.7 nous avons la flèche maximum obtenue par les diverses méthodes pour les différents cas étudiés. Nous remarquons que la différence maximum obtenue est de seulement 1,31%.

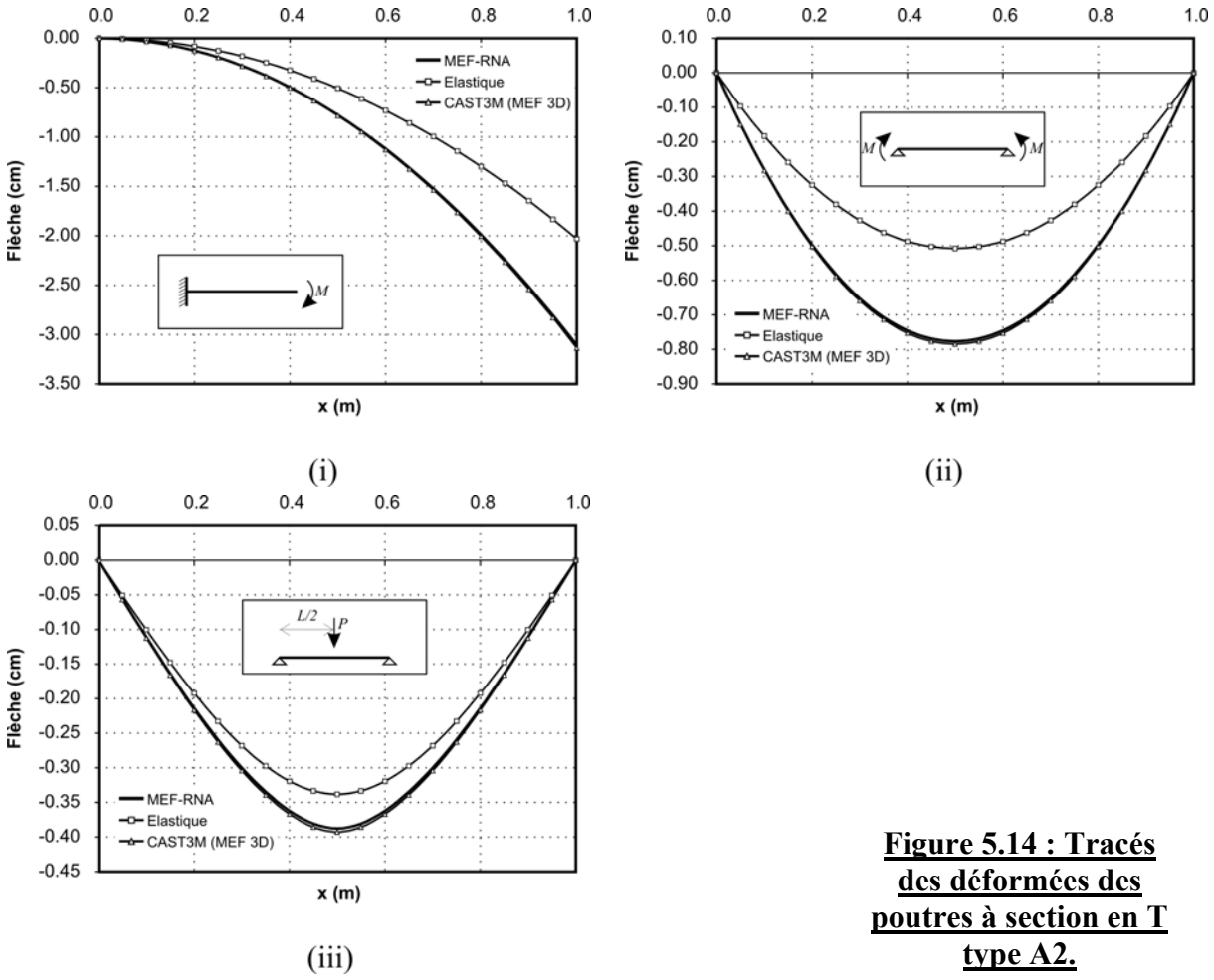


Figure 5.14 : Tracés des déformées des poutres à section en T type A2.

Schéma Statique Figure N°	Charge Appliquée	**Flèche maximum** (cm)		
		MEF-RNA	Elastique	**CAST3M. MEF.3D**
(5.1.a)	*M*=1.436 KN.m	3.1071 *0.94%*	2.0312 *35.24%*	**3.1365**
(5.1.d)	*M*=1.436 KN.m	0.7768 *0.99%*	0.5078 *35.27%*	**0.7845**
(5.1.e)	*P*=5.744 KN	0.3878 *1.31%*	0.3385 *13.86%*	**0.3929**

Tableau 5.7 : Récapitulatif des flèches maxima pour les sections en T type A2.

5.5 Conclusion

Dans ce chapitre, nous avons testé le code de calcul élaboré et la stratégie adoptée par des exemples simples. Il s'agit de poutres à une seule travée de longueur L unitaire, avec des conditions aux limites et des sollicitations représentatives des différents cas envisageables. Nous avons alors étudié des poutres isostatiques et hyperstatiques, chargées soit par un moment, soit par une force ponctuelle ou répartie. Les différentes configurations ont été illustrées.

L'utilisation du programme développé nécessite l'élaboration de fichiers de données. Un modèle a donc été détaillé. Pour valider les résultats obtenus, nous avons utilisé un code d'éléments finis très élaboré : CAST3M, version 2001. Les mêmes poutres y ont été modélisées en 3D et nous avons présenté, à titre d'exemples, deux modèles en éléments finis tridimensionnels, un par forme de section de poutre retenu.

Concernant les résultats trouvés, nous avons présenté, dans une première étape, ceux relatifs au comportement d'une poutre à section triangulaire. Les résultats obtenus à partir du programme élaboré sont, pour les différents cas traités, très proches de ceux donnés par le logiciel CAST3M. La différence est restée inférieure à 1,57%. La stratégie retenue pour interpoler la relation constitutive non linéaire à l'aide des réseaux de neurones artificiels semble efficace.

Nous avons ensuite présenté les résultats calculés pour des poutres ayant diverses formes de sections en T, à savoir des sections de type A1, A2, B, C1 et C2, conformément aux définitions données au chapitre 3. Nous avons exposé l'étude d'une section par type, et pour chacune d'elles nous avons considéré divers schémas statiques.

Les résultats du programme sont proches de ceux donnés par le logiciel CAST3M. La différence maximum obtenue est de seulement 2,34%. Nous pouvons alors avancer que le programme élaboré et la stratégie adoptée semblent bien fonctionner.

Chapitre 6

Conclusion générale

L'objectif de ce travail était de développer un code de calcul utilisant les réseaux de neurones artificiels à rétropropagation pour résoudre les équations régissant le comportement non linéaire des poutres sollicitées en flexion simple. La finalité d'une telle démarche est de prouver que les réseaux de neurones artificiels peuvent être utilisés efficacement dans un code de calcul en éléments finis en tant qu'outil d'interpolation des lois de comportement. Nous avons également fait allusion à la réduction du temps de calcul qui reste l'un des soucis majeurs en calcul et modélisation des structures malgré le développement continu des méthodes et des moyens informatiques. Le choix des poutres en flexion simple comme exemple d'éléments structuraux n'est théoriquement qu'une étape. L'objectif final est de généraliser la méthodologie aux cas plus compliqués.

Les réseaux de neurones artificiels se veulent une reproduction plus ou moins réaliste des neurones biologiques. Il s'agit d'une représentation informatique du système nerveux humain. Nous avons commencé par présenter leur modèle mathématique, les différentes fonctions d'activation usuelles, ainsi que les différentes règles d'apprentissage. La règle delta généralisée ou règle de la rétropropagation du gradient de l'erreur est la règle d'apprentissage utilisée dans ce travail, nous avons alors exposé son algorithme en détail. Nous avons ensuite passé en revue quelques travaux scientifiques dans le génie civil, élaborés en utilisant les réseaux neuromimétiques.

Ensuite, nous avons formalisé le problème à résoudre. Nous avons alors présenté les hypothèses simplificatrices de la théorie des poutres et nous avons posé les équations qui régissent leur comportement. Il s'agit des relations traduisant l'équilibre, des équations de compatibilité et des équations constitutives. Ces dernières prennent la forme de relations moment - courbure. Elles dépendent de la géométrie de la section et de la loi de comportement du matériau considéré. Nous avons supposé un comportement du matériau élasto-plastique parfait et nous avons alors exposé le développement de lois constitutives des poutres à sections triangulaires et en T, soit des sections présentant une seul axe de symétrie.

La méthode retenue pour résoudre les équations différentielles gouvernant la flexion simple des poutres est la méthode des éléments finis. Elle reste la méthode la plus puissante et la plus flexible surtout vis-à-vis d'une éventuelle généralisation de la procédure. Nous avons commencé par écrire un programme en langage Pascal permettant de calculer la flexion simple des poutres de Timoshenko. Nous y avons ensuite implanté les lois constitutives non

linéaires des poutres à sections triangulaires et en T sous une forme utilisant les réseaux de neurones artificiels. Pour en arriver là, il a fallu proposer une stratégie d'insertion et chercher les réseaux de neurones permettant d'effectuer différentes tâches d'interpolation nécessaires à cette stratégie.

L'utilisation du programme développé nécessite l'élaboration de fichiers de données. Leur création a été expliquée à travers un exemple. Ensuite, nous avons testé le programme élaboré et l'efficacité de la stratégie proposée. Nous avons pris différents exemples de poutres fléchies qui sont représentatifs des différentes situations envisageables. Nous avons alors étudié des poutres isostatiques et hyperstatiques, chargées soit par un moment, soit par une force ponctuelle ou répartie. Nous avons comparé les résultats obtenus à ceux trouvés dans la littérature et à ceux fournis par le logiciel CAST3M, un programme d'éléments finis très puissant. Nous y avons alors discrétisé les mêmes poutres par des éléments finis solides tridimensionnels.

Dans cette étape de validation du programme, nous avons d'abord présenté les résultats relatifs au comportement des poutres à section triangulaire. Les résultats obtenus à partir du programme élaboré sont, pour les différents cas traités, très proches de ceux donnés par le logiciel CAST3M. La différence est restée inférieure à 1,57%. La stratégie retenue pour interpoler la relation constitutive non linéaire à l'aide des réseaux de neurones artificiels a bien fonctionné.

Nous avons ensuite présenté les résultats calculés pour des poutres ayant des sections en T. Contrairement aux sections triangulaires, la relation constitutive des sections en T n'est pas unique. En fonction de la position de l'axe plastique et de l'évolution de la plastification avec la charge, on classe ces sections en cinq types, chacun ayant sa propre expression moment - courbure théorique. Nous avons alors présenté l'étude d'une section par type, et pour chacune d'elles nous avons considéré divers schémas statiques. Les résultats du programme sont restés très proches de ceux donnés par le logiciel CAST3M. La différence maximum obtenue est de seulement 2,34%. Le seul réseau de neurones utilisé pour interpoler les différentes expressions des différents cas a donc joué correctement son rôle.

Nous pouvons conclure que le programme élaboré et la stratégie adoptée fonctionnent correctement et donnent des résultats fiables. La différence par rapport à une méthode plus élaborée est restée tout le temps à des niveaux très faibles et avec des valeurs semblables, sans grande fluctuation.

D'un autre côté, nous pouvons affirmer que les réseaux de neurones artificiels ont fait preuve d'une grande capacité d'apprentissage avec relativement peu de neurones. En effet, si la relation moment - courbure des sections triangulaires peut sembler facile à apprendre, il en est autrement pour le cas des sections en T où on a observé des différences notables entre les relations constitutives d'une section en T à une autre.

Nous pouvons également conclure que les réseaux de neurones ont montré une grande capacité de généralisation. Effectivement, aucune des sections en T traitées n'a été présentée au réseau lors de la phase d'apprentissage. C'est-à-dire que le réseau n'a jamais appris directement leurs relations moment - courbure.

Une dernière conclusion concerne la démarche à adopter lorsqu'on souhaite utiliser les réseaux de neurones artificiels pour étudier un problème quelconque. Nous avons en effet perdu beaucoup de temps à chercher des réseaux compliqués pour réaliser nos objectifs alors que notre salut a été dans une démarche très simple. Dans ce cadre, nous pensons, qu'avec les réseaux de neurones, il faut avoir une attitude similaire à celle qu'on adopterait face à des

enfants. Il faut donc être très pédagogue, méthodique et leur présenter les choses le plus simplement possible.

Concernant les objectifs fixés, il reste néanmoins à prouver que la démarche retenue est plus rapide que les techniques traditionnelles. Cela n'a pas été possible dans cette étude car les temps d'exécution dans le cas de la méthode des éléments finis appliquée aux poutres sont très faibles. Nous ne pouvons pas les évaluer correctement par les moyens de mesure du temps disponibles, surtout que c'est le système d'exploitation qui affecte les priorités aux différentes applications. D'une exécution à une autre, nous avons des différences relatives notables entres les temps d'exécution d'un même programme.

A l'avenir, il serait intéressant d'étendre cette étude à d'autres types de sections, à divers matériaux constitutifs et à d'autres types d'éléments finis. On pourrait commencer, par exemple, par généraliser le programme élaboré à l'étude des portiques tridimensionnels. Une autre perspective intéressante serait d'utiliser des résultats expérimentaux dans la phase d'apprentissage des réseaux. On interpolerait ainsi une loi de comportement expérimentale. La méthode proposée n'en serait que plus valorisée.

Annexe A

Scripts et fonctions Matlab pour calculer les relations moment - courbure adimensionnelles.

A.1 Sections triangulaires

Dans cette annexe nous donnons le script écrit sur le logiciel Matlab afin de calculer la relation moment – courbure adimensionnelle des poutres à sections triangulaires. Il s'agit de la transcription sur Matlab des relations exposées dans le paragraphe 3.4. La relation est calculée sous la forme d'un couple de vecteurs, m – k, qui serviront comme pattern d'apprentissage au réseau de neurones. La dernière partie de ce script trace la courbe correspondante.

Fichier Plast.m

```
% Calcule la relation moment-courbure adimentionels pour
% les sections TRIANGULAIRES à comportement élasto-plastique parfait
% soumises à la flexion simple.

beta=sqrt(2)+2;
p=20;
%
% Phase 1 de Plastification en fibre supérieure
%
n101 = ((sqrt(3)-1)/2);
n1 = (0:0.01:n101)';
k1 = (3./(2-3.*n1+n1.^3))/p;
m1 = (3.*n1.^2+2.*n1+1)./(4.*(n1+2))*beta;
%
% Phase 2 de Plastification en fibre inférieure
%
n1ult = (p-1)*sqrt(3/(2*(3*p*p+1)));
```

```
n2 = (n101:0.01:n1ult)';
zz = 2 - n2 .* n2;
k2 = 2 .* (n2 + sqrt(zz/3)) ./ (p*(1 - 2 * (n2 .* n2)));
m2 = (4*sqrt(zz) .* (-2 * (n2.^5) + n2.^3 +4*n2.^2 - 3*n2+1));
m2 = m2-3*sqrt(3)*(2*n2.^2-4*n2+1);
m2 = m2 ./ (sqrt(zz) .* (4*n2.^2+1) + 3*sqrt(3) * n2);
m2 = m2 * beta / 4;
%
% Valeurs Ultimes
%
kult = sqrt(2*(3*p*p+1)/3) / p;
mult = beta * (1-3*sqrt(1.5)*p*(p*p+1)/(3*p*p+1)^1.5);
%
% Concatenation
%
n = [n1 ; n2; n1ult];
k = [k1 ; k2 ; kult];
m = [m1 ; m2 ; mult];
%
% Reste la phase élastique
%
zz = (0:0.02:0.98)';
ke = zz * 3 / 2 / p;
me = zz * beta / 8;
k = [ke ; k];
m = [me ; m];
plot(k,m,'r-+')
xlabel('k (courbure sans dimension)')
ylabel('m (moment sans dimension)')
title('Relation Moment-Courbure sans dimension')
text(kult,mult,'  C')
text(k1(1),m1(1),'  A')
text(k2(1),m2(1),'  B')
text(((kult+k2(1))/2),((mult+m2(1))/2),'Phase Plastique 2')
text(((k1(1)+k2(1))/2),((m1(1)+m2(1))/2),'  Phase Plastique 1')
text(k1(1)/2,m1(1)/2,'   Phase élastique')
hold on
plot([k1(1);k2(1);kult],[m1(1);m2(1);mult],'ks')
hold off
```

A.2 Sections en T

Dans cette partie nous présentons la fonction écrite sous Matlab pour calculer la relation moment – courbure adimensionnelle des poutres à sections en T. Les paramètres de cette fonction sont, respectivement, les variables sans dimensions associées aux épaisseurs de l'âme et de la semelle, et le paramètre de ductilité p.

Fichier TPlas.m

```
function [m,k] = Tplas(tau,zeta,p)
% Calcule la relation moment-courbure adimentionels pour
% les sections en Tés à comportement élasto-plastique parfait
% soumises à la flexion simple.

Npas=30;
```

```
sigma=tau*(1-zeta)+zeta;
gama =1-(1-tau)*(1-zeta)*(1-zeta);
pi   =1-(1-tau)*(1-zeta)*(1-zeta)*(1-zeta);

beta = calculbeta(tau,zeta);

% Phase élastique
[me, ke] = elastique(beta,sigma,gama,pi,p,Npas);

% Plastification de la fibre supérieure
[ms,ks] = Pfibresup(tau,beta,sigma,gama,pi,p,Npas);

% Plastification de la fibre inférieure
[mi,ki] = Pfibreinf(tau,zeta,beta,sigma,gama,pi,p,Npas);

m = [me;ms;mi];
k = [ke;ks;ki];


function beta = calculbeta(tau,zeta)
%
if (tau*(1-zeta)) < zeta
    % Classe C
    beta=1/(zeta*zeta+tau*(1-zeta)*(2-tau*(1-zeta)));
elseif (tau*(1-zeta)) > zeta
    % Classe A
    beta=tau/(tau^2*(1-zeta)^2+zeta*(2*tau-zeta));
elseif (tau*(1-zeta)) == zeta
    % Classe B
    beta=1/(2*(zeta*(2-zeta)-tau*(1-zeta)^2));
else
    error('Unexpected situation')
end


function [m,k] = elastique(beta,sigma,gama,pi,p,Npas)
pas = 1/Npas;
zz = (0:pas:1)';
k01 = 2 * sigma / gama / p;
m01 = beta * 2 * (4 * sigma * pi - 3 * gama * gama) / 3 / gama ;
m = zz * m01;
k = zz * k01;


function [m,k] = Pfibresup(tau,beta,sigma,gama,pi,p,Npas)
delta = sigma^2-4*tau*(gama-sigma);
n112 = (sigma - sqrt(delta)) / 2 / tau;
pas = n112/Npas;
n1 = ((0+pas):pas:n112)';
k = 2 .* sigma ./ p ./ (tau .* n1.^2 - 2*sigma .* n1 + gama);
num =       2 .* sigma .* tau .* n1.^3;
num = num - 3 .* tau .* gama .* n1.^2;
num = num + (4 * sigma * pi - 3 * gama * gama);
den = tau .* n1.^2 - 2 .* sigma .* n1 + gama;
m = 2/3 .* num ./ den .* beta;

function [m,k] = Pfibreinf(tau,zeta,beta,sigma,gama,pi,p,Npas)
%
C1=0;
```

```
C2=0;
B=0;
A1=0;
A2=0;
if (tau*(1-zeta)) < zeta
    % Classe C
    if ( tau >= (((p+1)*zeta-2)/(p-1)/(1-zeta)) )
        C1=1;
    else
        C2=1;
    end
elseif (tau*(1-zeta)) > zeta
    % Classe A
    if ( tau <= ((p+1)*zeta/(p-1)/(1-zeta)) )
        A1=1;
    else
        A2=1;
    end
elseif (tau*(1-zeta)) == zeta
    % Classe B
    B=1;
else
    error('Unexpected situation')
end



delta = sigma^2-4*tau*(gama-sigma);
n112 = (sigma - sqrt(delta)) / 2 / tau;

if A1 | B | C1
    lamda =(p*(1-sigma)+1)^2-(1-zeta)*(1-sigma)*((p+1)^2-tau*(p-1)^2);
    n123 = (p-1)*(p*(1-sigma)+1+sqrt(lamda))/((p+1)^2-tau*(p-1)^2);
    Npas2=Npas;
elseif A2
    n123=zeta/tau;
    Npas2=Npas/2;
else
    n123=1-zeta;
    Npas2=Npas/2;
end

pas = (n123-n112)/Npas2;
n1 = ((n112+pas):pas:n123)';

r = 1 + (1-tau) * (1-zeta)- 2 .* n1;
s = (1-tau) .* ( (1-zeta) - n1 ).^2;
r24s = r.^2 - 4 .* s;

k = 4 ./ p ./ (r+sqrt(r24s));

num =      -2 .* tau .* n1.^2;
num = num + 4 .* (sigma-zeta) .* n1;
num = num + sigma .* ( 1 + 3 * zeta - sigma);
num = num - zeta + sqrt(r24s) .* (sigma - 2 * zeta);
m = 2/3 .* num .* beta;

% Phase plastique n°3
if A2
```

```
    n1ult=(p-1)*sigma/(2*tau*p);
    pas = (n1ult-n123)/Npas2;
    n1 = ((n123+pas):pas:n1ult)';
    k3 = 2 * tau ./ (p .* (sigma - 2 .* tau .* n1));
    k=[k;k3];
    m3 =    -2 .* tau.^2 .* n1.^2 + 2 .* tau .* sigma .* n1;
    m3 = m3 + sigma*(sigma-3*zeta)+3*tau*zeta;
    m3 = 2/3 .* m3 .* beta ./ tau;
    m=[m;m3];
end
if C2
    n1ult=(p-1)*(2-sigma)/2/p;
    pas = (n1ult-n123)/Npas2;
    n1 = ((n123+pas):pas:n1ult)';
    k3 = 2 ./ ( p .* (2 - sigma -2 .* n1));
    k=[k;k3];
    m3 = -2 .* n1.^2 + 2 .* (2-sigma) .* n1 + sigma*(5+3*zeta-2*sigma) -
(2+3*zeta);
    m3 = 2/3 .* m3 .* beta;
    m=[m;m3];
end
```

A.3 Pattern d'apprentissage du réseau des sections en T

Le petit script, nommé TdonRN.m, donné ci-dessous nous a servi pour générer les données de l'apprentissage des réseaux de neurones des sections en T. Il s'agit des relations moment – courbure, calculées par la fonction Tplas ci-dessus, correspondant à 200 valeurs aléatoires des épaisseurs adimensionnelles de l'âme et de la semelle. Le paramètre de ductilité est fixé à 20.

Fichier TdonRN.m

```
M=[];
K=[];
ZETA=[];
TAU=[];
for i = 1:200
 x=0.03 + 0.47 * rand(1);
 y=0.03 + 0.47 * rand(1);
[m,k] = Tplas(x,y,20);
zeta=ones(size(m)) * y;
tau=ones(size(m)) * x;
M=[M;m];
K=[K;k];
ZETA=[ZETA;zeta];
TAU=[TAU;tau];
end
M=M';
K=K';
ZETA=ZETA';
TAU=TAU';
```

Annexe B

Le programme MEF élaboré pour les poutres à sections en T.

B.1 Programme principal

Dans ce qui suit nous donnons le code source du programme principal, T2.PAS, élaboré pour étudier la flexion simple des poutres à sections en T. Il est écrit en langage PASCAL et comporte une unité (UNIT), appelée NNTool, contenant les fonctions relatives à l'utilisation des réseaux de neurones. Cette unité est donnée dans l'annexe B.2 ci-après.

Fichier T2.PAS

```
Program BEAM;
USES NNTool;
Const
  MaxNDofN=2;
  MaxNMats=1;
  MaxNprop=7;
  MaxNelem=20;
  MaxNNodE=2;
  MaxNPoin=21;
  MaxNSvab=MaxNDofN*MaxNPoin;
  MaxNEvab=MaxNDofN*MaxNNodE;
  MaxSGRF=(MaxNSvab*MaxNSvab) div 2;
Type  {Real=extended;}
  MatricePropriete=array[1..MaxNMats,1..MaxNProp] of Real;
  MatriceConnectivite=array[1..MaxNElem,1..MaxNNodE] of Integer;
  VecteurListeMateriaux=array[1..MaxNElem] of Integer;
  matriceCoordonneeNoeuds=array[1..MaxNPoin]    of    Real;{Vecteur    car
UniDimen.}
  CodeAppuiDesDDL=array[1..MaxNSvab] of Integer;
  ValDeplImposeAuxDDL=array[1..MaxNSvab] of real;
  MatriceChargNodalSurElem=array[1..MaxNElem,1..MaxNEvab] of real;
  DeformationPlastique=array[1..MaxNElem] of real;{1 d,f.plat/,l,m}
```

```
  Contrainte=array[1..MaxNElem,1..MaxNDofN] of real;{2 contr/,l,m}
  MatriceDeplacementNoeuds=array[1..MaxNPoin,1..MaxNDofN] of real;
  RigiditeElementaire=array[1..MaxNEvab,1..MaxNEvab] of real;
  FichierRigiditeElementaire=File of RigiditeElementaire;
  ForceGlobale=ValDeplImposeAuxDDL;
  DeplaGlobal=ForceGlobale;
  RigiditeGlobale=array[1..MaxNSvab,1..MaxNSvab] of real;
  RigiditeGlobalePtr=^RigiditeGlobale;
  StoredGaussianReductionFactors=array[1..MaxSGRF] of real;
Var
 NomFicDon:String;
 Don,Res:Text;

 NPoin,NElem,NBoun,NMats,NProp,NNodE,NIncs,NAlgo,NDofN:Integer;
 NEvab,NSvab:Integer;
 Props:MatricePropriete;
 LNods:MatriceConnectivite;
 MatNo:VecteurListeMateriaux;
 Coord:matriceCoordonneeNoeuds;
 IFpre:CodeAppuiDesDDL;
 PeFix:ValDeplImposeAuxDDL;
 RLoad:MatriceChargNodalSurElem;

{Accumulative arrays}
 Plast:DeformationPlastique;
 Stres:Contrainte;
 ELoad,TLoad:MatriceChargNodalSurElem;{Force    residuelle    et    totale
appliqu,e}
 TDisp,TReac:MatriceDeplacementNoeuds;{Inconnues   totales   et   reactions
totales}
{Force globale=second membre}
 ASLod:ForceGlobale;
 ASTIF:RigiditeGlobalePtr;
 {Reactions,deplacements}
 React:ForceGlobale;
 XDisp:DeplaGlobal;
{Controle du calcul incr,mental}
 Niter,NoutP,NChek:integer;
 Facto,Toler,Pvalu:real;
 KResl:integer;
 Fixed:ValDeplImposeAuxDDL;
 {variables secondaires}
 iincs,iiter:integer;

 Fresv:StoredGaussianReductionFactors;




 Procedure DATA(Var Don,Res:Text;
                Var NPoin,NElem,NBoun,NMats,NProp,
                          NNodE,NIncs,NAlgo,NDofN:Integer;
                Var NEvab,NSvab:Integer;
                Var Props:MatricePropriete;
                Var LNods:MatriceConnectivite;
                Var MatNo:VecteurListeMateriaux;
                Var Coord:matriceCoordonneeNoeuds;
```

```
                Var IFpre:CodeAppuiDesDDL;
                Var PeFix:ValDeplImposeAuxDDL;
                Var RLoad:MatriceChargNodalSurElem);
  {Lecture de donn,es sur la g,om,trie,le chargement,
  les conditions aux limites,...}
  Var
    Title:String;
    imats,Jmats,iprop:integer;
    ielem,Jelem,inode:integer;
    ipoin,Jpoin:integer;
    isvab,ievab:integer;
    idofn:integer;
    iboun,NodFx,NPosn:integer;
    icode:array[1..MaxNDofN] of integer;
    value:array[1..MaxNDofN] of real;


   Begin {DATA}
     Readln(Don,Title);
     Writeln(Res,Title);
     Readln(Don,NPoin,NElem,NBoun,NMats,NProp,NNodE,NIncs,NAlgo,NDofN);
     Writeln(Res,'NPoin=',NPoin:5,'  NElem=',NElem:5,'  NBoun=',NBoun:5,
               '  NMats=',NMats:5);
     Writeln(Res,'NProp=',NProp:5,'  NNodE=',NNodE:5,'  Nincs=',NIncs:5,
               '  NAlgo=',NAlgo:5,'  NDofN=',NDofN:5);
     NEvab:=NDofN*NNodE; NSvab:=NDofN*NPoin;
     Writeln(Res,'   Proprit,s des Mat,riaux');
     For imats:=1 to NMats do
       begin
         Read(Don,Jmats);
         For iprop:=1 to NProp do Read(Don,Props[Jmats,iprop]);Readln(Don);
         Write(Res,Jmats:5);
         For            iprop:=1           to           NProp           do
Write(Res,Props[Jmats,iprop]:15);writeln(Res);
       end;
     Writeln(Res,'   EL    Noeuds   Mat.');
     For ielem:=1 to NElem do
       begin
         Read(Don,Jelem);
         For inode:=1 to NNodE do Read(Don,LNods[Jelem,inode]);
         Readln(Don,MatNo[Jelem]);
         write(Res,Jelem:5);
         For inode:=1 to NNodE do write(res,LNods[Jelem,inode]:5);
         writeln(Res,MatNo[Jelem]:5);
       end;
     Writeln(Res,'Noeud   Coordonnee');
     for ipoin:=1 to NPoin do
       begin
         Read(Don,Jpoin);Read(Don,Coord[Jpoin]);Readln(Don);
         Write(Res,Jpoin:5,'  ');Write(Res,Coord[Jpoin]:15);Writeln(Res);
       end;
     For isvab:=1 to NSvab do
       begin
         IFpre[isvab]:=0;
         PeFix[isvab]:=0.0;
       end;
     Write(Res,'Noeud App.  ');
     for idofn:=1 to NDofN do write(Res,'  DDL',idofn:1,' Code Val.Impos,e
');
```

```
    writeln(Res);
    for iboun:=1 to NBoun do
      begin
        read(Don,NodFx);
        for idofn:=1 to NDofN do read(Don,icode[idofn],value[idofn]);
        readln(Don);
        write(Res,NodFx:5,'        ');
        for idofn:=1 to NDofN do write(Res,'    ',icode[idofn]:5,
                                       value[idofn]:15);
        writeln(Res);
        NPosn:=(NodFx-1)*NDofN;
        for idofn:=1 to NDofN do
          begin
            NPosn:=NPosn+1;
            IFpre[NPosn]:=icode[idofn];
            PeFix[NPosn]:=value[idofn];
          end;
      end;{for iboun}
    writeln(res,'  El,ment        Forces Nodales');

    for ielem:=1 to NElem do
      for ievab:=1 to NEvab do RLoad[ielem,ievab]:=0.0;
    Repeat
      Read(Don,Jelem);
      for ievab:=1 to NEvab do read(Don,RLoad[Jelem,ievab]);
      Readln(Don);
    until (Jelem=NElem);
    for ielem:=1 to NElem do
      begin
        write(Res,ielem:5);
        for ievab:=1 to NEvab do write(Res,RLoad[ielem,ievab]:15);
        writeln(Res);
      end;
  end;{DATA}


Procedure Initial( NElem,NDofN,NEvab,NPoin:Integer;
                   Var Plast:DeformationPlastique;
                   Var Stres:Contrainte;
                   Var ELoad,TLoad:MatriceChargNodalSurElem;
                   Var TDisp,TReac:MatriceDeplacementNoeuds);
     {Initialise … z,ro les grandeurs accumulatives}
  Var
    ielem,idofn,ievab,ipoin:integer;
  begin
    for ielem:=1 to NElem do
      begin
        Plast[ielem]:=0.0;
        for idofn:=1 to NDofN do Stres[ielem,idofn]:=0.0;
        for ievab:=1 to NEvab do
          begin
            ELoad[ielem,ievab]:=0.0;
            TLoad[ielem,ievab]:=0.0;
          end;
      end;
    for ipoin:=1 to NPoin do
      for idofn:=1 to NDofN do
        begin
          TDisp[ipoin,idofn]:=0.0;
```

```
          TReac[ipoin,idofn]:=0.0;
        end;
  end;{Initial}

Procedure IncLoad(Var Don,Res:Text;
                      iincs,NElem,NEvab:integer;
                  Var Niter,NoutP:integer;
                  Var Facto,Toler:real;
                  Var ELoad,TLoad:MatriceChargNodalSurElem);
  {Lit les donn,es pour l'actuel incr,ment et actualise le vecteur charge}
  var
    ielem,ievab:integer;
  begin
    Readln(Don,NIter,NOutP,Facto,Toler);
    Writeln(Res,'IIncs=',iincs:5,'  NIter=',Niter:5,'  NOutP=',NOutP:5,
                '  Facto=',Facto:10:3,'  Toler',Toler:6:2);
    for ielem:=1 to NElem do
      for ievab:=1 to NEvab do
        begin
          ELoad[ielem,ievab]:=ELoad[ielem,ievab]+RLoad[ielem,ievab]*Facto;
          TLoad[ielem,ievab]:=TLoad[ielem,ievab]+RLoad[ielem,ievab]*Facto;
        end;
  end;{IncLoad}


Procedure NonAL(NAlgo,iincs,iiter,NSvab:integer;
                Facto:Real;
                Pefix:ValDeplImposeAuxDDL;
                Var KResl:integer;
                Var Fixed:ValDeplImposeAuxDDL);
  {Cr,e les indicateurs identifiant le type d'algorithme de r,solution}
  var
    isvab:integer;
  begin
    KResl:=2;
   {it,ration directe}
    If (NAlgo=1) then KResl:=1;
   {Rigidit, tangente}
    If (NAlgo=2) then KResl:=1;
   {Rigidit, Initiale:calculer une fois}
    If ((NAlgo=3) and (iincs=1) and (iiter=1)) then KResl:=1;
   {Comb1:Recalculer rigidit, … la premiere it,ration de chaque incr,ment}
    If ((NAlgo=4) and (iiter=1)) then KResl:=1;
   {Comb2:Recalculer … la seconde it,ration de chaque incr,ment, sauf au
      d,but o— il faut calculer la matrice de rigidit, initiale}
    If ((NAlgo=5) and (iincs=1) and (iiter=1)) then KResl:=1;
    If ((NAlgo=5) and (iiter=2)) then KResl:=1;
   {Les d,pl impos,s doivent ˆtre mis … la bonne valeur.
   Pour l'it,ration directe le pb est entiŠrement r,analys,;ces d,pl
impos,s
   sont introduits … la bonne valeur … chaque ,tape. Par contre pour les
   autres algorithmes qui sont cumulatifs et o— les d,placements sont
obtenus
   en cumulant les incr,ments obtenus … chaque it,ration, il faut
introduire
   la bonne valeur … la premiŠre it,ration de chaque incr,ment de charge
   et imposer des z,ros pour la suite des it,rations. Ainsi les
d,placements
   obtenus aux appuis … la convergence seront ,gaux aux d,placements
```

```
    impos,s. Les donn,es des d,pl impos,s sont stock,s dans PeFix au niveau
    de la proc,dure Data. Pour tenir compte de ce qui est dit ci-dessus,
    on les transfert aux proc,dures de r,solution via le vecteur Fixed.}
     If ((iiter=1) or (NAlgo=1))
        then for isvab:=1 to NSvab do Fixed[isvab]:=PeFix[isvab]*Facto
        else for isvab:=1 to NSvab do Fixed[isvab]:=0.0;
  end;{NonAL}

Function Beta(tau,zeta:real):real;
  begin
    if (tau>(zeta/(1-zeta))) then {Classe A}
       beta:=tau/(tau*tau*(1-zeta)*(1-zeta)+zeta*(2*tau-zeta))
    else if (tau=(zeta/(1-zeta))) then {Classe B}
           beta:=1/2/(zeta*(2-zeta)-tau*(1-zeta)*(1-zeta))
         else if (tau<(zeta/(1-zeta))) then {Classe C}
             beta:=1/(zeta*zeta+tau*(1-zeta)*(2-tau*(1-zeta)))
  end;

Procedure StiffB(NomFicDon:string;
                 NElem:integer;
                 MatNo:VecteurListeMateriaux;
                 Props:MatricePropriete;
                 LNods:MatriceConnectivite;
                 Coord:matriceCoordonneeNoeuds;
                 Plast:DeformationPlastique;
                 Stres:Contrainte);
  Var
    FRE:FichierRigiditeElementaire;
    ESTIF:RigiditeElementaire;
    ielem,lprop,node1,node2:integer;
    b,h,tw,tf,E,G,Sig0,Eleng,EIval,Svalu:real;
    tau,zeta,sigma,gama,pi,A,I:real;
    Mm1,Km1,CorrM,CorrK,p:real;
    valu1,valu2,valu3,valu4:real;
    istif,jstif:integer;

  Begin
    Assign(FRE,(NomFicDon+'.rig'));Rewrite(FRE);
    for ielem:=1 to NElem do
      begin
        lprop:=MatNo[ielem];
        b    :=Props[lprop,1];
        h    :=Props[lprop,2];
        tw   :=Props[lprop,3];
        tf   :=Props[lprop,4];
        E    :=Props[lprop,5];
        G    :=Props[lprop,6];
        Sig0 :=Props[lprop,7];
        node1:=LNods[ielem,1];
        node2:=LNods[ielem,2];

        tau  :=tw/b;
        zeta :=tf/h;
        sigma:=tau*(1-zeta)+zeta;
        gama :=1-(1-tau)*(1-zeta)*(1-zeta);
        pi   :=1-(1-tau)*(1-zeta)*(1-zeta)*(1-zeta);
        A    :=b*h*sigma;
        I    :=b*h*h*h/12*(4*sigma*pi-3*gama*gama)/sigma;{INUTILE !!}
```

```
        Eleng:=ABS(coord[node2]-coord[node1]);
        Svalu:=G*A/1.5;{GA/1.5}


        Mm1:=ABS(Stres[ielem,1]); {Moment actuellement enregistr,}
        CorrM:=b*h*h*Sig0/4/beta(tau,zeta);
        Mm1:=Mm1/CorrM; {moment sans dimnsion}
        Km1:=K(tau,zeta,Mm1);    {Courbure sans dimension}
        If Km1<0 then Km1:=0.0;
        p:=20; {HypothŠse prise dans le M,moire de DEA p42}
        CorrK:=p*Sig0/E/h;
        EIval:=EIt(tau,zeta,Km1)*CorrM/CorrK; {Raideur Tangente}

        {Formation de la matrice de rigidit,}
        valu1:=0.5*Svalu;
        valu2:=Svalu/Eleng;
        valu3:=EIval/Eleng;
        valu4:=0.25*Svalu*Eleng;
        ESTIF[1,1]:=valu2;
        ESTIF[1,2]:=valu1;
        ESTIF[1,3]:=-valu2;
        ESTIF[1,4]:=valu1;
        ESTIF[2,2]:=valu3+valu4;
        ESTIF[2,3]:=-valu1;
        ESTIF[2,4]:=-valu3+valu4;
        ESTIF[3,3]:=valu2;
        ESTIF[3,4]:=-valu1;
        ESTIF[4,4]:=valu3+valu4;
        for istif:=1 to 4 do
          for jstif:=istif to 4 do ESTIF[jstif,istif]:=ESTIF[istif,jstif];
        write(FRE,ESTIF);
      end;
    Close(FRE);
  End;{StiffB}


Procedure AssembForce(NSvab,NNodE,NDofN:integer;
                LNods:MatriceConnectivite;
                ELoad:MatriceChargNodalSurElem;
                var ASLod:ForceGlobale);

  Var
    isvab,ielem,inode,nodeI,idofn:integer;
    nrowS,nrowE:integer;
  Begin
    for isvab:=1 to NSvab do ASLod[isvab]:=0.0;
    for ielem:=1 to NElem do
      begin
        for inode:=1 to NNodE do
          begin
            nodeI:=LNods[ielem,inode];
            for idofn:=1 to NDofN do
              begin
                nrowS:=(nodeI-1)*NDofN+idofn;
                nrowE:=(inode-1)*NDofN+idofn;
              {Assembler les charges elementaires}
                ASLod[nrowS]:=ASLod[nrowS]+ELoad[ielem,nrowE];
              end;{idofn}
          end;{inode}
```

```
      end;{ielem}
  End;{AssembForce}


Procedure AssembRig(NomFicDon:String;
                    NSvab,NNodE,NDofN:integer;
                    LNods:MatriceConnectivite;
                    ASTIF:RigiditeGlobalePtr);
  Var
    FRE:FichierRigiditeElementaire;
    ESTIF:RigiditeElementaire;
    isvab,jsvab,ielem,inode,nodeI,idofn:integer;
    nrowS,nrowE,jnode,nodeJ,jdofn,ncolS,ncolE:integer;
  Begin
    Assign(FRE,(NomFicDon+'.rig'));Reset(FRE);
    {Initialiser la rigidit, globale}
    for isvab:=1 to NSvab do
          for jsvab:=1 to NSvab do ASTIF^[isvab,jsvab]:=0.0;
    for ielem:=1 to NElem do
      begin
        read(FRE,ESTIF);
        for inode:=1 to NNodE do
          begin
            nodeI:=LNods[ielem,inode];
            for idofn:=1 to NDofN do
              begin
                nrowS:=(nodeI-1)*NDofN+idofn;
                nrowE:=(inode-1)*NDofN+idofn;
                for jnode:=1 to NNodE do
                  begin
                    nodeJ:=LNods[ielem,jnode];
                    for jdofn:=1 to NDofN do
                      begin
                        ncolS:=(nodeJ-1)*NDofN+jdofn;
                        ncolE:=(jnode-1)*NDofN+jdofn;
                        {Assembler la rigidite globale}

ASTIF^[nrowS,ncolS]:=ASTIF^[nrowS,ncolS]+ESTIF[nrowE,ncolE];
                      end;{jdofn}
                  end;{jnode}
              end;{idofn}
          end;{inode}
      end;{ielem}
    Close(FRE);
  End;{AssembRig}


Procedure Greduc(NSvab:integer;
                 IFpre:CodeAppuiDesDDL;
                 Fixed:ValDeplImposeAuxDDL;
                 ASTIF:RigiditeGlobalePtr;
                 var Fresv:StoredGaussianReductionFactors;
                 var ASLod:ForceGlobale);
  {triangularisation de Gauss surmatrice et second membre}
  var
    kount,neqns,ieqns,irows,icols:integer;
    pivot,factr:real;
  Begin
    kount:=0;
```

```
    Neqns:=NSvab;
    for ieqns:=1 to Neqns do
      begin
        If (IFpre[ieqns]<>1) then {reduce equation}
          begin
            pivot:=ASTIF^[ieqns,ieqns];
            if (ABS(pivot)<1.0E-10) then
                                      begin
                                        writeln;
                                        writeln('Pivot incorrect');
                                        writeln('Equation=',ieqns);
                                        HALT(1);
                                      end;
            for irows:=(ieqns+1) to Neqns do
              begin
                kount:=kount+1;
                factr:=ASTIF^[irows,ieqns]/pivot;
                Fresv[kount]:=factr;
                if (factr<>0.0) then
                  begin
                    for icols:=ieqns to Neqns do
                      ASTIF^[irows,icols]:=ASTIF^[irows,icols]
                                           -factr*ASTIF^[ieqns,icols];
                    ASLod[irows]:=ASLod[irows]-Factr*ASLod[ieqns];
                  end;{if (factr<>0.0)}
              end;{irows}
          end {If (IFpre[ieqns]<>1)}
          else begin {Ajuster le second membre pour tenir compte des
                      deplacememts impos,s}
            for irows:=ieqns to Neqns do
              ASLod[irows]:=ASLod[irows]-ASTIF^[irows,ieqns]*Fixed[ieqns];
          end;{Else de If (IFpre[ieqns]<>1)}
      end;{ieqns}
  End;{Greduc}


Procedure Resolv(NSvab:integer;
                 IFpre:CodeAppuiDesDDL;
                 Fixed:ValDeplImposeAuxDDL;
                 ASTIF:RigiditeGlobalePtr;
                 Fresv:StoredGaussianReductionFactors;
                 var ASLod:ForceGlobale);
  {Si KResl=2, on ne forme et ne reduit que le second membre}
  var
    kount,Neqns,ieqns,irows:integer;
    factr:real;
  Begin
    kount:=0;
    Neqns:=NSvab;
    for ieqns:=1 to Neqns do
      begin
        If (IFpre[ieqns]<>1) then {reduce equation}
          begin
            for irows:=(ieqns+1) to Neqns do
              begin
                kount:=kount+1;
                factr:=Fresv[kount];
                if (factr<>0.0) then
                    ASLod[irows]:=ASLod[irows]-Factr*ASLod[ieqns];
```

```
              end;{irows}
            end {If (IFpre[ieqns]<>1)}
            else begin {Ajuster le second membre pour tenir compte des
                        deplacememts impos,s}
              for irows:=ieqns to Neqns do
                ASLod[irows]:=ASLod[irows]-ASTIF^[irows,ieqns]*Fixed[ieqns];
            end;{Else de If (IFpre[ieqns]<>1)}
        end;{ieqns}

  End;{Resolv}

Procedure BakSub(NSvab,NPoin,NElem,NNodE,NDofN:integer;
                 LNods:MatriceConnectivite;
                 Fixed:ValDeplImposeAuxDDL;
                 ASTIF:RigiditeGlobalePtr;
                 ASLod:ForceGlobale;
                 var React:ForceGlobale;
                 var XDisp:DeplaGlobal;
                 var TDisp,TReac:MatriceDeplacementNoeuds;
                 var TLoad:MatriceChargNodalSurElem);
  {Back-substitution = Resolution = calcul des inconnues}
  var
    Neqns,ieqns,nback,icols:integer;
    pivot,resid:real;
    kount,ipoin,idofn,ielem,inode,nloca:integer;
    nposn,ievab:integer;
  Begin
    Neqns:=NSvab;
    for ieqns:=1 to Neqns do React[ieqns]:=0.0;

    for nback:=Neqns downto 1 do
      begin
        pivot:=ASTIF^[nback,nback];
        resid:=ASLod[nback];
        If (nback<Neqns) then
            for icols:=(nback+1) to Neqns do
                  resid:=resid-ASTIF^[nback,icols]*XDisp[icols];
        If (IFpre[nback]=0) then XDisp[nback]:=resid/pivot
                            else begin
                              XDisp[nback]:=Fixed[nback];
                              React[nback]:=-resid;
                            end;
      end;{nback}

    {Incrementer les resultats dans variables totales}
    kount:=0;
    for ipoin:=1 to NPoin do
      for idofn:=1 to NDofN do
        begin
          kount:=kount+1;
          TDisp[ipoin,idofn]:=TDisp[ipoin,idofn]+XDisp[kount];
          TReac[ipoin,idofn]:=TReac[ipoin,idofn]+React[kount];
        end;{idofn}

    {Lors du calcul des forces residuelles, la contribution des reactions
    doit etre prise en compte du moment qu'on peut les voir
    comme des forces necessaires pour maitenir la valeur du d,placement
    impos,. Ainsi, les reactions calculees doivent etre ajoutee au
    vecteur des forces nodales appliqu,es … chaque iteration. Comme on a
```

```
    choisi d'organiser les donn,es des forces nodales par element
    et non par noeuds,il est necessaire de chercher un element qui
    contient ce noeud }
    for ipoin:=1 to NPoin do
      begin
        ielem:=0;
        repeat
          ielem:=ielem+1;
          inode:=0;
          repeat
            inode:=inode+1;
            nloca:=LNods[ielem,inode];
          until ((inode=NNodE) or (ipoin=nloca));
        until ((ielem=NElem) or (ipoin=nloca));
        for idofn:=1 to NDofN do
          begin
            nposn:=(ipoin-1)*NDofN+idofn;
            ievab:=(inode-1)*NDofN+idofn;
            Tload[ielem,ievab]:=Tload[ielem,ievab]+React[nposn];
          end;{idofn}
      end;{ipoin}
  End;{BakSub}


Procedure ReForB(NElem,NEvab,NDofN:integer;
                 MatNo:VecteurListeMateriaux;
                 Props:MatricePropriete;
                 LNods:MatriceConnectivite;
                 Coord:matriceCoordonneeNoeuds;
                 XDisp:DeplaGlobal;
                 var Stres:Contrainte;
                 var Plast:DeformationPlastique;
                 var ELoad:MatriceChargNodalSurElem);
  {Calcule les forces nodales internes equivalentes}
  var
    ielem,ievab,lprop,node1,node2:integer;
    {EIval,Svalu,Yield,Hards,Eleng:real;}
    wnod1,wnod2,Thta1,Thta2:real;
    Stran{,STlin,STcur,PreYS,Rfact,Reduc}:real;
    b,h,tw,tf,E,G,Sig0,Svalu,Eleng:Real;
    tau,zeta,sigma,A:real;
    p,CorrK,CorrM,Signe,PreST:Real;
    IncCourbSD,PreMomentSD,IncMomentSD:Real;

  Begin
    for ielem:=1 to NElem do
      for ievab:=1 to NEvab do ELoad[ielem,ievab]:=0.0;
    for ielem:=1 to NElem do
      begin
        lprop:=MatNo[ielem];
        b    :=Props[lprop,1];
        h    :=Props[lprop,2];
        tw   :=Props[lprop,3];
        tf   :=Props[lprop,4];
        E    :=Props[lprop,5];
        G    :=Props[lprop,6];
        Sig0 :=Props[lprop,7];
        node1:=LNods[ielem,1];
        node2:=LNods[ielem,2];
```

```
        tau  :=tw/b;
        zeta :=tf/h;
        sigma:=tau*(1-zeta)+zeta;
        A    :=b*h*sigma;

        Eleng:=ABS(Coord[node2]-Coord[node1]);
        Svalu:=G*A/1.5;{GA/1.5}

        p:=20; {HypothŠse prise dans le M,moire de DEA p42}
        CorrK:=p*Sig0/E/h;
        CorrM:=b*h*h*Sig0/4/beta(tau,zeta);

        {d,placements des noeuds (rem:increment)}
        wnod1:=XDisp[node1*NDofN-1];
        wnod2:=XDisp[node2*NDofN-1];
        Thta1:=XDisp[node1*NDofN];
        Thta2:=XDisp[node2*NDofN];
        {d,formations:STRAiN (rem:increment)}
        Stran:=(Thta1-Thta2)/Eleng;
        PreST:=Stres[ielem,1];

        If ((PreST) < 0) then Signe:=-1.0
        else If ((PreST) > 0) then Signe:=1.0
             Else If (Stran>0) then Signe:=1.0 Else Signe:=-1;

        IncCourbSD:=signe*(Stran/CorrK);
        PreMomentSD:=signe*(PreST/CorrM);
        IncMomentSD:=Mip1(tau,zeta,PreMomentSD,IncCourbSD);
        Stres[ielem,1]:=Stres[ielem,1]+Signe*IncMomentSD*CorrM;



        {La relation effort tranchant/distorsion est consideree tjs
        elastique. Aucune disposition articuliere n'est … prendre}
        Stres[ielem,2]:=Stres[ielem,2]+(Svalu/Eleng)*(Wnod2-Wnod1)
                        -0.5*Svalu*(Thta1+Thta2);
        {Calcul des forces nodales equivalentes}
        ELoad[ielem,1]:=ELoad[ielem,1]-Stres[ielem,2];
        ELoad[ielem,2]:=ELoad[ielem,2]+Stres[ielem,1]
                                      -0.5*Eleng*Stres[ielem,2];
        ELoad[ielem,3]:=ELoad[ielem,3]+Stres[ielem,2];
        ELoad[ielem,4]:=ELoad[ielem,4]-Stres[ielem,1]
                                      -0.5*Eleng*Stres[ielem,2];
      end;{For ielem}
  End;{ReForB}


Procedure Conund(var RES:text;
                 iiter,NSvab,NElem,NNodE,NDofN:integer;
                 Toler:real;
                 LNods:MatriceConnectivite;
                 TLoad:MatriceChargNodalSurElem;
                 var NChek:integer;
                 var ELoad:MatriceChargNodalSurElem;
                 var Pvalu:real);
  var
    Resid,Retot,refor,ratio:real;
    StFor,ToFor:ForceGlobale;
```

```
    isvab,ielem,ievab,inode,NodNo,idofn,nposn:integer;
  {Test la covergence}
  Begin
    NChek:=0;   {initialise … 0 l'indicateur de convergence}
    Resid:=0.0; {initialise … 0 la norme des forces residuelles}
    Retot:=0.0; {initialise … 0 la norme des forces totales appliquees}
    for isvab:=1 to NSvab do
      begin
        StFor[isvab]:=0.0;{vecteur des forces nodales equivalentes}
        ToFor[isvab]:=0.0;{vecteur des forces nodales appliquees}
      end;
    for ielem:=1 to NElem do
      begin
        ievab:=0;
        for inode:=1 to NNodE do
          begin
            NodNo:=LNods[ielem,inode];
            for idofn:=1 to NDofN do
              begin
                ievab:=ievab+1;
                nposn:=(NodNo-1)*NDofN+idofn;
                StFor[nposn]:=StFor[nposn]+ELoad[ielem,ievab];
                ToFor[nposn]:=ToFor[nposn]+TLoad[ielem,ievab];
              end;
          end;
      end;
    for isvab:=1 to NSvab do
      begin
        refor:=ToFor[isvab]-StFor[isvab];
        Resid:=Resid+refor*refor;
        Retot:=Retot+ToFor[isvab]*ToFor[isvab];
      end;
    for ielem:=1 to NElem do
      for ievab:=1 to NEvab do
        ELoad[ielem,ievab]:=TLoad[ielem,ievab]-ELoad[ielem,ievab];
    ratio:=100.0*sqrt(Resid/Retot);
    if (ratio>Toler) then NChek:=1;
    If (iiter>1) then if (ratio>Pvalu) then NChek:=999;
    Pvalu:=ratio;
    writeln(RES,'It,ration Nø:',iiter:5,' Code de convergence:',NChek:4,
                'Ratio des normes de residus:',ratio:15);
  End;{Conund}


Procedure Result(Var RES:text;
                     NdofN,Npoin,Nelem:integer;
                     TDisp,TReac:MatriceDeplacementNoeuds;
                     Stres:Contrainte;
                     Plast:DeformationPlastique);
  var
    ipoin,idofn,ielem:integer;
  Begin
    If   NdofN=1   then   writeln(RES,'              Noeud           Depla.
Reaction');
    If   NdofN=2   then   writeln(RES,'              Noeud           Depla.
Reaction',
                                  '       Depla.            Reaction');
    For ipoin:=1 to NPoin do
      begin
```

```
         write(RES,ipoin:10);
         for idofn:=1 to NdofN do
           write(RES,TDisp[ipoin,idofn]:14,'     ',TReac[ipoin,idofn]:14);
         writeln(RES);
       end;
     writeln(RES);
     If NdofN=1 then writeln(RES,'  ELEMENT     CONTRAIN     DEF.PLAST');
     If  NdofN=2  then  writeln(RES,'    ELEMENT                      CONTRAIN
DEF.PLAST');
     for ielem:=1 to Nelem do
       begin
         write(RES,ielem:10);
         for idofn:=1 to NdofN do
           write(RES,Stres[ielem,idofn]:14);
         writeln(RES,Plast[ielem]:14);
       end;
   End;{Result}

BEGIN {Principal BEAM}
  New(ASTIF);
  {Write('Fichier de donn,es (sans extension): ');Readln(NomFicDon);}
  NomFicDon:='TesB3bis'{'BeamData'};
  Assign(Don,NomFicDon);Reset(Don);
  Assign(Res,(NomFicDon+'.res'));Rewrite(Res);

  DATA(Don,Res,
       NPoin,NElem,NBoun,NMats,NProp,NNodE,NIncs,NAlgo,NDofN,
       NEvab,NSvab,Props,LNods,MatNo,Coord,IFpre,PeFix,RLoad);

  Initial(NElem,NDofN,NEvab,NPoin,
          Plast,Stres,ELoad,TLoad,TDisp,TReac);

  for iincs:=1 to NIncs do
    begin
      IncLoad(Don,Res,iincs,NElem,NEvab,
              Niter,NoutP,Facto,Toler,ELoad,TLoad);
      iiter:=0;
      Repeat
        iiter:=iiter+1;
        NonAL(NAlgo,iincs,iiter,NSvab,Facto,Pefix,
              KResl,Fixed);
        AssembForce(NSvab,NNodE,NDofN,LNods,ELoad,
                    ASLod);

        If (KResl=1) then
          begin
               StiffB(NomFicDon,NElem,MatNo,Props,LNods,Coord,Plast,Stres);
                                    {Output=ESTIF mais sur disk}
               AssembRig(NomFicDon,NSvab,NNodE,NDofN,LNods,
                         ASTIF);
               Greduc(NSvab,IFpre,Fixed,
                      ASTIF,Fresv,ASLod);
          end {If (KResl=1)}
          else Resolv(NSvab,IFpre,Fixed,ASTIF,Fresv,
                      ASLod);
        BakSub(NSvab,NPoin,NElem,NNodE,NDofN,LNods,Fixed,ASTIF,ASLod,
               React,XDisp,TDisp,TReac,TLoad);
        ReForB(NElem,NEvab,NDofN,MatNo,Props,LNods,Coord,XDisp,
               Stres,Plast,ELoad);
```

```
        Conund(RES,iiter,NSvab,NElem,NNodE,NDofN,Toler,LNods,TLoad,
               NChek,ELoad,Pvalu);
      until ((iiter=Niter) or (Nchek=0) or (Nchek=999));
      if ((iiter=Niter) and (Nchek<>0)) then
        begin
          writeln(RES,'LA SOLUTION N''A PAS CONVERGE !');
          Result(RES,NdofN,Npoin,Nelem,TDisp,TReac,Stres,Plast);
        end;
      If (Nchek=999) then
         begin
           writeln(RES,'LA SOLUTION DIVERGE ! ITERATION:',iiter:5);
           Result(RES,NdofN,Npoin,Nelem,TDisp,TReac,Stres,Plast);
           {Close(DON);Close(RES);Dispose(ASTIF);Halt(999); }
         end;
      If (Nchek=0) then
        begin
          writeln(RES,'CONVERGENCE A L''ITERATION:',iiter:5);
          Result(RES,NdofN,Npoin,Nelem,TDisp,TReac,Stres,Plast);
        end;
    end;{for iincs}

  Close(Don);
  Close(Res);
  Dispose(ASTIF);
END.
```

B.2 Unité NNtool.PAS

Toutes les fonctions relatives à l'utilisation des réseaux de neurones dans le programme élaboré ont été regroupées dans l'unité Pascal ci-dessous. Elle a néanmoins besoin du fichier contenant les poids et biais des coefficients synaptiques calculés sur Matlab. Ils sont donnés dans l'annexe B.3.

Fichier NNTool.PAS

```
Unit NNTool;

INTERFACE

TYPE Real=Extended;

Function K(tau,zeta,m:real):real;
Function M(tau,zeta,k:real):real;
Function EIt(tau,zeta,k:real):real; {Raideur Tangente}
Function Mip1(tau,zeta,Mi,Dk:real):real;



IMPLEMENTATION

Const
    NCou=3; {Nombre de couches, y compris celle de sortie}
    Ninp= 3; {Nombre d'inputs:tau,zeta et m (ou k)}
```

```
    N1  =10;
    N2  =10;
    N3  = 1;   {Nombre d'outputs}
    Nmax=10; {Taille max du vecteur de travail=max(Ninp,N1,N2,N3...)}
    nWgt=Ninp*N1+N1+N2*N1+N2+N3*N2+N3;
Type
    Archi= array[0..NCou] of byte;
    Weight=array[1..nWgt] of real;
    WeightPtr=^Weight;
    WorkVect=array[1..Nmax]of real;
Const
    NnC:Archi = (Ninp, N1 , N2 , N3);{Nombre de neurones par couche}
    TrC:Archi = (0, 2 , 2 , 1);{Type de fonction de transfert:
                                    1=purelin;2=logsig}
    MKfilename='MK310101.wgt';
    KMfilename='KM310101.wgt';
Var
    Fic:text;
    MKw,KMw:WeightPtr;
    i:integer;

Function logsig(x:real):real;
  begin
    logsig:=1/(1+exp(-x));
  end;{Logsig}

Function Lw(W:WeightPtr;ic,i,j:integer):real;
  var
    N,k:integer;
  begin
    N:=0;
    for k:=1 to ic-1 do N:=N+NnC[k]*NnC[k-1]+NnC[k];
    N:=N+(j-1)*NnC[ic]+i;
    Lw:=W^[N];
  end;

Function b1(W:WeightPtr;ic,i:integer):real;
  var
    N,k:integer;
  begin
    N:=0;
    for k:=1 to ic-1 do N:=N+NnC[k]*NnC[k-1]+NnC[k];
    N:=N+NnC[ic]*NnC[ic-1];
    N:=N+i;
    b1:=W^[N];
  end;

Function Sim(tau,zeta,xx:real;W:WeightPtr):real;
  var
  ic,i,j  : integer;
  X,Y : WorkVect;

  begin
    X[1]:=tau; X[2]:=zeta; X[3]:=xx;
    For ic:=1 to Ncou do
      begin
        For i:=1 to NnC[ic] do
          begin
            Y[i]:=0.0;
```

```
            For j:=1 to NnC[ic-1] do Y[i]:=Y[i]+Lw(W,ic,i,j)*X[j];
          end;
        For i:=1 to NnC[ic] do
          begin
            Y[i]:=Y[i]+b1(W,ic,i);
            Case TrC[ic] of
              1: X[i]:=Y[i];
              2: X[i]:=logsig(Y[i]);
            end;
          end;
      end;
    Sim:=X[1];
  end;{Sim}

Function K(tau,zeta,m:real):real;
  begin
    K:=Sim(tau,zeta,m,KMw);
  end;{K(m)}

Function M(tau,zeta,k:real):real;
  begin
    M:=Sim(tau,zeta,k,MKw);
  end;{K(m)}

Function EIt(tau,zeta,k:real):real; {Raideur Tangente}
  var Delta:real;
  begin
   Delta:=0.01;
   EIt:=(M(tau,zeta,(k+Delta))-M(tau,zeta,k))/Delta;
  end;

Function Mip1(tau,zeta,Mi,Dk:real):real;
  Var Ki,a,b,c,Miap:real;
  begin
    If Mi=0 then
      begin
        ki:=0.0;
      end else begin
        a:=0;b:=3;
        repeat
          c:=(b+a)/2;
          Miap:=M(tau,zeta,c);
          If Miap<Mi then
            begin
              a:=c;
            end else
            begin
              b:=c;
            end;
        until (Abs(Mi-Miap)/Mi)<1E-5;
        Ki:=c;
      end;
    Mip1:=M(tau,zeta,(Ki+Dk))-Mi;
  end;{Mip1}

BEGIN
{Lire les poids … partir des fichiers contenant tous les poids
 et biais fournis par MATLAB}
New(MKw);New(KMw);
```

```
Assign(fic,MKfilename);Reset(fic);
For i:=1 to nWgt do read(fic,MKw^[i]);
Close(fic);

Assign(fic,KMfilename);Reset(fic);
For i:=1 to nWgt do read(fic,KMw^[i]);
Close(fic);

END.
```

B.3 Fichier des Poids et Biais du réseau MK310101.

Dans ce qui suit nous donnons le fichier contenant les poids et biais des liaisons synaptiques calculés par Matlab pour le réseau MK310101, soit 161 valeurs (voir paragraphe 4.5).

Fichier MK310101.WGT

```
 4.2939525041872537e+000
 2.9444565890979106e+000
-1.6106771460003584e+000
 4.0303220712404436e-001
 1.0428944865075835e+000
-1.3667832097390718e+001
 4.3521134017946866e-001
 1.4161673321286678e+001
-9.7432696422657212e+000
 2.7259843378837658e+000
-6.4096416110891576e+000
-8.8029266075601846e+000
 7.4860483173835179e+000
 2.9670348702504095e+000
-1.0834959677767184e-001
 3.2730542563350873e+001
 9.9703937387292385e-001
-5.1481350006374953e+001
-4.4444751118239152e+000
-1.2796894081218118e+000
-1.8647670023419945e+001
-5.3090092830398046e+000
 1.5706856083038192e+000
 2.0690569326755748e+000
-6.9917132858683262e+000
 4.5507080500586481e+000
-2.4745113937612487e+001
-6.1857891424623113e-001
-4.4766824969155478e-001
 2.4158121657261130e+001
 2.7749529517802469e+000
 3.5866431873248956e+000
 1.3699721988037469e-001
-6.0172082043002473e-001
-3.1899335853507760e-001
-4.0206902403630034e+000
```

```
 3.6649566712260970e+000
 1.0957984622626911e+000
-1.4980989116486254e+000
-3.7862170475202634e+000
-2.4461503141287113e+000
-1.3412283876685782e+000
-2.6063486501343828e+001
 2.6300585240854217e+000
 9.5797057214269188e-001
-1.0392350416445981e+000
 1.6886519918790842e+001
 1.5361653409248224e+000
 1.2874015928961506e+000
 5.8021227083216616e-002
-9.4585182399780088e+000
 1.4247236325058024e+000
 7.7368688227379421e+000
-8.9455788629409341e+000
-4.1496644226549133e+000
 4.1192172482860183e+000
-1.6232134446271150e+001
 6.6580317699714211e+000
 9.5569588216123491e-001
-5.1641921782943162e-002
 4.1784413908669151e+000
-3.0646634312215908e+000
 2.7856638218174270e+000
 1.9237804233695766e+001
 7.8766062126165730e+000
-7.6764469100055868e+000
 7.7283350465726164e+000
-2.3812605764637652e+001
-1.1702630560534004e+001
 9.4729447028946950e-002
-3.3117618332618989e+001
 5.5817890407301425e+000
-1.4711114198117718e+001
-3.5849355940380271e+001
-1.7794402837226816e+001
 1.7655915153154425e+001
-1.7086826983248919e+001
 3.4614889112825033e+001
 1.8121251945131931e+001
-1.0165335496915786e+000
-6.1100505602159449e+000
-1.0629129527537104e-001
 5.9541970664093817e+000
-5.0944378252982467e+000
-7.7920296725676552e-001
 8.6550684061824390e-001
-3.6028499052528988e+000
 1.7502303685493679e+000
-1.5467729485389730e+000
 3.9984227762326632e+000
-4.8591560950390750e-001
-2.2068389920396936e-001
-3.3961235119077045e+001
 5.4169048281937297e-002
-7.0511108502526842e-001
```

```
 6.9103652954010819e-001
 6.1530572054850063e-001
-7.1206071995849063e-001
 9.4624525999354295e-002
 3.7979790058466335e-002
 2.6913362162571396e+001
-2.2498261642575250e+000
 3.3235506768502844e+001
 7.4191753797946642e+000
-1.6421507149163256e-001
 2.3048155667211900e-001
 2.4317994390505028e+001
-3.5893455924756359e+000
-1.0939912559976341e+000
-1.2518669921599124e-001
-2.5888819128724516e-001
-6.1683127595555005e-002
 1.6071131875562761e+000
 1.2929851078965819e+000
-3.3501744474352573e+000
 3.3487586901298965e+000
 5.5477318595111191e-001
 3.8075721610808579e+001
 3.4435613341199685e+000
-1.3878763344758346e-001
 5.7000658095475396e+000
-1.7330982668763106e+001
 3.0365474905852757e+001
 1.3179231147699110e+001
 2.4056962428388342e+001
-2.4137127210416821e+001
-1.5986217270342580e+001
-2.4380135024142081e+001
-2.7437069989715244e+001
 9.2546855720546652e+000
 6.3251161117763717e+000
 1.0108783555033533e+001
 1.1527711606939601e+001
 7.4926126820563521e+000
 5.4426007300021961e-001
-5.3354513166268869e-001
 1.5109819136197572e-002
-3.7008890324667258e+000
 1.0826905750055760e+000
-1.5056294132228594e+000
-5.8872502068076544e+000
 6.3240695024807403e+000
-1.7206449134730306e+001
-1.6813782954140648e+000
 6.4385213152133414e+000
-6.5185627341464079e+000
-1.5671809806285214e+001
 2.6421159760266071e+000
-2.7066929159691566e-001
-1.6772062013801565e+000
 2.4623420897718788e+001
 3.2680384506623433e+001
-5.0881486998518677e-001
 3.0070125289911601e+001
```

```
-1.3768757169366886e+001
-1.3497439740753844e+001
-6.2231462061106330e-002
 3.5313157479990469e+000
-4.2144862286824758e-001
-1.9799669220135327e+000
-2.1262476407139641e+001
```

Bibliographie

[1] ABDI H. *Les réseaux de neurones*. Presse Universitaire de Grenoble, 1994.

[2] BATHE K. J. *Numerical methods in finite elements analysis*. Prentice Hall, Englewood Cliffs, 1976.

[3] BEIROW B. & OSTERRIEDER P. Dynamic investigations of TV towers. In *Structural Engineering, Mechanics and Computation, Proc. International Conf., 2-4 April 2001, Cape Town, South Africa* (Elsevier, Amsterdam, 2001), A. Zingoni, Ed., Vol. 1, pp.629-636.

[4] COLLINET P. *Etude des poutres en flexion simple par les réseaux de neurones artificiels à rétropropagation et à apprentissage supervisé*. Mémoire de DEA, Université Blaise Pascal – Clermont-Ferrand 2, Clermont-Ferrand, septembre 1996.

[5] DEMUTH H. & BEALE M. *Neural Network Toolbox User's Guide, For Use with MATLAB* . The MathsWorks Inc., U.S.A, July 1992.

[6] DERRAS B. *Estimation du risque lié a l'effet de site et génération d'un spectre de réponse a la surface libre.* Mémoire de Magister, Université Abou Bekr Belkaid, Tlemcen, Algérie, juin 2004.

[7] DHATT G. & TOUZOT G. *Une présentation de la méthode des éléments finis*. Les Presses de l'Université Laval, Québec, Canada, 1981.

[8] DJAFOUR M. *Les méthodes numériques*. Notes de cours de Magister. Université Abou Bekr Belkaid, Tlemcen, Algérie, 2001.

[9] DJAFOUR M. & MEGNOUNIF A. *Elasticité générale*. Office des Publications Universitaires, Algerie, 1994.

[10] FRANCOIS D. PINEAU A. & ZAOUI A. *Comportement mécanique des matériaux. Élasticité et plasticité.* Edition Hermes, Paris, 1995.

[11] FREEMAN J. A. & SKAPURA D. M. *Neural Networks Algorithms, Applications, and Programming Techniques*. Addison – Wesley Publishing Company, U.S.A, July 1992.

[12] IZEBOUDJEN N. *Conception et implémentation en FPGA d'un classificateur neuronal des arythmies cardiaques*. Mémoire de Magister, Ecole Nationale Polytechnique, Alger, 1999.

[13] KO J. M., SUN Z. G. & NI Y. Q. A three-stage scheme for damage detection of Kap Shui Mun cable-stayed bridge. In *Structural Engineering, Mechanics and Computation, Proc. International Conf., 2-4 April 2001, Cape Town, South Africa* (Elsevier, Amsterdam, 2001), A. Zingoni, Ed., Vol. 1, pp. 111-122.

[14] LANGIS D. FAFARD M. & HENCHI K. *Identification structurale et réseaux neuromimétiques : calibration de modèles éléments finis.* Rapport GCS-97-09, Université Laval, Canada, Août 1997.

[15] MEGNOUNIF A. *Calcul des structures élasto-plastiques*. Polycopié, Université de Tlemcen, Algérie, Mars 1993.

[16] MAYORAZ F. CORNU T. & VULLIET L. Using neural networks to predict slope movements. *Proc. 7th Int. Symp. on Landslides., June 1996, Trondheim,* Vol. 1, pp. 295-300

[17] OWEN D. R. J. & HINTON E. *Finite elements in plasticity: Theory and practice*. Pineridge Press Limited, Swansea, U. K. 1980.

[18] PROPAGATOR. *Neural Network Development Software*. ARD Corporation, Columbia, USA, March, 1994.

[19] RAHMOUN Z. *Nouvelle approche de génération d'accélerogramme artificiel compatible avec un spectre de réponse en utilisant les réseaux de neurones.* Mémoire de Magister, Université Abou Bekr Belkaid, Tlemcen, juillet 2002.

[20] TIMOSHENKO S. P. & GOODINER J. N. *Théorie de l'élasticité*. Dunod, Paris, 1961.

[21] VASSILEVA S. T. Predicting earthquake ground motion descriptions through artificial neural networks for testing the constructions. In *Structural Engineering, Mechanics and Computation, Proc. International Conf., 2-4 April 2001, Cape Town, South Africa* (Elsevier, Amsterdam, 2001), A. Zingoni, Ed., Vol. 2, pp.927-934.

[22] ZIENCKIEWICZ O. C. *The finite element method*. Mc Graw Hill Book Co, New York, 1977.