RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement et de la Recherche Scientifique

École Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Département d'Electrotechnique

Laboratoire de Recherche en Electrotechnique

Mémoire de projet de fin d'étude en vue d'obtention du diplôme

Ingénieur d'état en Électrotechnique

Etude de l'Ecoulement de Puissance et Résolution par des Techniques Intelligentes

Présenté par:

Abderrahmane Abdoun
Feriel Achour

Sous la direction de :

Pr. Abdelhafid HELLAL (ENP)

Présenté et soutenu publiquement le 08/07/2021

Composition du jury:

Président	**Pr. K. Boughrara**	**ENP**
Examinateur	**Dr. R. Belkacemi**	**ENP**
Rapporteur	**Pr. A. Hellal**	**ENP**

ENP, 2021

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement et de la Recherche Scientifique

École Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Département d'Electrotechnique

Laboratoire de Recherche en Electrotechnique

Mémoire de projet de fin d'étude en vue d'obtention du diplôme

Ingénieur d'état en Électrotechnique

Etude de l'Ecoulement de Puissance et Résolution par des Techniques Intelligentes

Présenté par:

Abderrahmane Abdoun
Feriel Achour

Sous la direction de :

Pr. Abdelhafid HELLAL (ENP)

Présenté et soutenu publiquement le 08/07/2021

Composition du jury:

Président	**Pr. K. Boughrara**	**ENP**
Examinateur	**Dr. R. Belkacemi**	**ENP**
Rapporteur	**Pr. A. Hellal**	**ENP**

ENP, 2021

Dédicaces

En signe de respect et de reconnaissance, on dédie ce modeste travail à :

- Nos chères familles, celles qui nous ont dotés d'une éducation digne, leurs amour et sacrifices ont fait de nous ce que nous sommes aujourd'hui.

- Nos amis et à toutes personnes pour lesquelles nous avons une place dans leurs cœurs.

Nous vous remercions vivement pour votre présence et votre soutien.

Feriel et Abderrahmane

Remerciements

Nos remerciements et notre reconnaissance s'adressent au bon Dieu qui nous a donné la santé, le courage, la patience ainsi que la force pour réaliser ce modeste travail.

Nous exprimons nos gratitudes et notre reconnaissance à notre directeur de mémoire, Pr. **Abdelhafid HELLAL,** pour sa patience, son encadrement, sa qualité organisationnelle et de planification, qui nous ont permis de mener à bien ce projet de mémoire durant la préparation de notre travail.

Nous sommes très reconnaissants à l'égard de l'ensemble des membres du jury : Le Professeur **K. BOUGHRARA**, le Docteur **R. BELKACEMI** de l'Ecole Nationale Polytechnique, qui ont accepté de porter un regard critique sur ce projet de mémoire de fin d'études.

Enfin, nous adressons nos remerciements les plus sincères à nos familles pour leurs soutiens et leurs encouragements pour poursuivre et réaliser ce travail.

ملخص

تم في هذه المذكرة، اقتراح استخدام الشبكات العصبية الاصطناعية لحل المشكلة المعروفة لتدفق الطاقة الكهربائية، حيث تقوم بتقييم الحالة المستقرة لأنظمة الطاقة وهي أداة أساسية للتخطيط والتشغيل ومراقبة أنظمة الطاقة الحديثة . يتكون النموذج الرياضي لتدفق الطاقة الكهربائية من مجموعة من المعادلات الجبرية غير الخطية التي تم حلها تقليديًا باستخدام **Gauss-Seidel** أو **Newton-Raphson** أو إصداراتها الأخرى, من أجل الاستفادة من السرعة العالية للشبكات العصبية الاصطناعية مقارنة بأساليب تدفق الطاقة الكهربائية التقليدية . تم استخدام المدرك متعدد الطبقات، و هو فئة من الشبكات العصبية الاصطناعية المدربة على طريقة النسب المتدرجة لحساب مقادير الجهد وزوايا مشكلة تدفق الطاقة الكهربائية. تم اختبار منهجية المدرك متعدد الطبقات المقترحة بنجاح باستخدام شبكة نقل الكهرباء المكونة من 5 عقدات وشبكة 14 عقدة.

الكلمات المفتاحية: تدفق الطاقة الكهربائية ، الطرق التكرارية ، الشبكة العصبية الاصطناعية ، المدرك متعدد الطبقات ، تدفق الخط.

Abstract

In this thesis, the use of artificial neural networks (ANNs) is proposed to solve the well-known power flow (PF) problem. PF evaluates the steady state of power systems and is a fundamental tool for the planning, operation and control of modern power systems. The mathematical model of PF consists of a set of nonlinear algebraic equations solved conventionally with the Gauss-Seidel, Newton-Raphson or its decoupled versions. In order to take advantage of the higher speed of ANNs compared to conventional power flow methods, the multilayer perceptron, a class of feedforward artificial neural network trained with the gradient descent method was used for the computation of the voltage magnitudes and angles of the PF problem. The proposed multilayer perceptron methodology was successfully tested using the 5 nodes grid and IEEE 14 nodes grid.

Key words: load flow, iterative methods, artificial neural network, multilayer perceptron, line flow.

Résumé

Dans ce mémoire, l'utilisation de réseaux neuronaux artificiels (RNA) est proposée pour résoudre le problème bien connu de l'écoulement de puissance (EP). L'EP évalue l'état stable des réseaux électriques et constitue un outil fondamental pour la planification, l'exploitation et le contrôle des systèmes électriques modernes. Le modèle mathématique de l'EP comprend un ensemble d'équations algébriques non linéaires résolues de manière conventionnelle avec la méthode Gauss-Seidel, Newton-Raphson ou ses versions découplées. Afin de profiter de la vitesse supérieure des RNA par rapport aux méthodes conventionnelles de l'écoulement de puissance, le perceptron multicouche, une classe de réseau neuronal artificiel à propagation directe entraîné avec la méthode de descente de gradient a été utilisé pour le calcul des amplitudes et des angles des tensions du problème de l'EP. La méthodologie du perceptron multicouche proposée a été testée avec succès en utilisant le réseau de 5 nœuds et sur le réseau IEEE 14 nœuds.

Mots clés : écoulement de puissance, méthodes itératives, réseau de neurones artificiels, perceptron multicouche, puissance de transit.

Table des matières

Liste des tableaux

Liste des figures

Introduction générale

Un système d'énergie peut être exploité par le système de gestion de l'énergie dans les centres de contrôle [1]. Plusieurs fonctions d'analyse en ligne et hors ligne sont souvent exécutées dans les centres de contrôle. Parmi ces fonctions, on peut citer la configuration de la topologie, le contrôle automatique de la production, l'estimation de l'état, l'analyse de sécurité statique, les systèmes équivalents externes [1], etc. L'exécution de la plupart de ces fonctions est basée sur la résolution d'un grand nombre de problèmes de l'écoulement de puissance.

Le problème de l'écoulement de puissance peut être formulé mathématiquement comme un ensemble d'équations algébriques non linéaires. L'objectif principal du calcul de l'EP est de déterminer les conditions de fonctionnement du système électrique en régime permanent en termes d'amplitudes de tension et d'angles de phase à chaque nœud [1]. Une fois les tensions connues, d'autres quantités peuvent être calculées, telles que les flux de puissance réelle et réactive à travers les branches (lignes de transmission et transformateurs) [1][2], les puissances réactives générées aux nœuds de production, les pertes de puissance de transmission réelles et réactives, et ainsi de suite.

Les réseaux électriques modernes sont grands, massivement interconnectés et les tâches liées à leur planification, leur exploitation et leur contrôle sont de plus en plus complexes. En outre, les systèmes sont exploités avec leurs équipements très proches de leurs limites. Ce scénario a conduit à l'utilisation de systèmes informatiques sophistiqués pour effectuer les tâches de planification, d'exploitation et de contrôle [1]. Il est nécessaire de calculer l'état de fonctionnement d'un système (en résolvant le problème de l'écoulement de puissance) avec des centaines, voire des milliers de nœuds et de branches.

Habituellement, le problème de l'écoulement de puissance est résolu par les méthodes itératives de Newton-Raphson, Gauss-Seidel ou par la méthode de Découplé rapide. En général, l'état de fonctionnement d'un système est obtenu après quelques itérations, quelle que soit la méthode utilisée. Cependant, il existe certaines situations de fonctionnement, correspondant à des cas critiques, où l'ensemble des équations de l'écoulement de puissance devient mal conditionné, et un plus grand nombre d'itérations peut être nécessaire [1]. Il existe également des situations pour lesquelles les équations de l'EP n'ont pas de solution ou le processus itératif diverge.

Avec l'évolution de la technologie de surtout le domaine de computation, le monde s'est orienté vers l'introduction des systèmes intelligents ayant le pouvoir de faire fonctionner, analyser et évaluer l'état de ces systèmes en utilisant des techniques intelligentes moderne. Les Smart Grids sont donc une évolution logique dans le domaine des réseaux électriques, à travers par exemple, la gestion intelligente de la production, du transport, de la distribution de l'énergie électrique ainsi que du fonctionnement de manière globale.

De nombreuses techniques alternatives d'intelligence artificielle (IA) ont été développées récemment et appliquées avec succès dans de nombreux domaines de l'ingénierie [Muller]. L'un des outils d'IA disponibles est le réseau neuronal artificiel (RNA). Le RNA peut être considéré comme une alternative intéressante pour résoudre des problèmes non linéaires tels que le problème de l'écoulement de puissance, même si les premières applications réussies de le RNA ont été dans les

domaines de la reconnaissance des formes, de la classification et de la prévision. Les RNAs ont également la capacité d'obtenir une solution de problèmes qui n'ont pas de modèle analytique clair.

Le perceptron multicouche (PMC) est l'architecture de réseau de neurone la plus utilisée en ingénierie, y compris dans les applications des systèmes électriques. Elle sera donc également adoptée dans ce travail. Un PMC est capable de résoudre des problèmes complexes et non linéaires tels que le problème l'écoulement de puissance. Dans ce mémoire, le perceptron multicouche a été entraîné en utilisant la méthode de descente de gradient.

L'idée de ce travail est de proposer l'utilisation d'un PMC pour déterminer l'état de fonctionnement en régime permanent d'un système électrique, autrement dit un point de fonctionnement stable à un moment donné pour une charge donnée. En outre, le processus de calcul doit être plus rapide que les méthodes existantes, telles que la méthode de Newton-Raphson.

Deux étapes constituent le modèle de calcul de l'écoulement de puissance basé sur le perceptron multicouche. La première étape est l'apprentissage : le modèle PMC proposé est préparé à résoudre les problèmes de flux d'énergie en lui présentant un historique des données d'entrées-sorties générées précédemment. La seconde est l'étape de test.

La méthodologie proposée a été testée pour les systèmes 3 nœuds, 5 nœuds et IEEE 14 nœuds, en considérant des conditions d'exploitation normales (cas de base) et plusieurs scénarios de contingence, y compris différents modèles de charge/génération. Les résultats de simulation ont montré la bonne performance du PMC, prouvant sa capacité à résoudre le problème de l'écoulement de puissance.

Et finalement, les résultats des programmes développés et les simulations sur les différents modèles standards du réseau de transport ont été regroupés dans une application « Intelligent Load Flow Calculator » facile à utiliser et toute modification ou amélioration est possible.

L'organisation de ce travail consiste en :

- une introduction générale,
- un premier chapitre dédié aux méthodes d'écoulement de puissance conventionnelles,
- un deuxième chapitre consacré aux réseaux de neurones artificiels,
- un troisième chapitre destiné aux applications et à la simulation d'un modèle RNA à l'écoulement de puissance,
- un quatrième et dernier chapitre qui présente l'outil élaboré à travers un programme avec interface conviviale basée sur la plateforme App Designer de Matlab,
- et enfin une conclusion générale.

Chapitre 1 :

Calcul d'Ecoulement de Puissance par les Méthodes Conventionnelles

1. Chapitre 1 : Calcul d'Ecoulement de Puissance par les Méthodes Conventionnelles

1.1 Introduction

Un réseau électrique est un système d'éléments interconnectés conçu pour débiter de l'énergie électrique et la transporter sur de longues distances [3]. Comme tout système physique, des modèles ont été établis pour étudier ces réseaux et prévoir des solutions face à plusieurs problèmes. L'objectif des études dans les réseaux électriques est d'assurer une meilleure qualité d'énergie électrique ; maintenir des niveaux de tension près de leur valeur nominale [3], et vérifier l'équilibre entre la consommation et la production de l'énergie électrique pour satisfaire les clients. Ce chapitre s'intéresse au calcul d'écoulement de puissance (Load flow, Power flow) qui est la solution de la condition de fonctionnement statique d'un système de transmission d'énergie électrique [4], basée sur des méthodes de résolution numériques les plus utilisées : Méthode de Gauss–Seidel (GS), Newton–Raphson (NR) et de Découplée Rapide (FDLF).

L'écoulement de puissance a pour but de :

– Pouvoir exploiter le réseau électrique en fonction de la demande (charges), et déterminer le meilleur schéma d'exploitation.
– Déterminer la tension dans chaque nœud, le niveau de tension doit être maintenu avec une certaine tolérance [5].
– Minimiser les pertes sur les lignes de transmission, les transformateurs, … etc [5].
– Optimiser le coût de génération de la puissance.

1.2 Modélisation des réseaux électriques

Comme généralement un réseau est maillé, nous nous intéressons essentiellement au réseau de transport. Un réseau électrique est constitué :

- Des moyens de production (unités de production),
- Des lignes de transport et transformateurs de puissance,
- Des charges.

Notons que les charges ne sont qu'exceptionnellement des clients directement raccordés en THT. Elles représentent plus généralement un point de connexion au réseau de distribution (typiquement 63 kV), via un transformateur et rarement en 400 kV (ex : El Hadjar, Annaba) [6].

Notons également que les moyens de réglage de la puissance réactive (les FACTS ; capacités, réactances, compensateurs statiques de puissance réactive) peuvent être assimilés à des charges ou à des moyens de production ne consommant ou ne fournissant que de la puissance réactive.
D'une manière générale, modéliser un réseau d'énergie, c'est avant tout faire un certain nombre d'hypothèses simplificatrices qui conditionneront à la fois la complexité et le domaine de validité du modèle [6].

Les principales hypothèses retenues, dans le cadre des différentes études, sont les suivantes : seul le comportement en régime permanent à 50 Hz est étudié : le réseau est supposé linéaire. Un choix

important doit alors être fait : le calcul des transits de puissance se limite-t-il à un fonctionnement totalement équilibré du réseau. L'étude du réseau peut être menée à partir d'un schéma monophasé équivalent. Cette approche est bien souvent suffisante dans le cadre de l'exploitation d'un réseau d'énergie. Elle permet déjà de prédéterminer, pour un plan de production et un niveau de charge donnés, quelle doit être la charge de chacune des lignes du réseau en fonctionnement normal, et aussi quel doit être le plan de tension du réseau [6].

1.2.1 Réseau de transport

Le transport de l'énergie électrique se fait en plusieurs étapes. Au départ de la centrale de production, la tension délivrée par l'alternateur (ex : 20 kV), subit une élévation par un transformateur élévateur vers 400 kV, 225 kV ou une autre tension de transport, sur de longues lignes. A l'arrivée vers le système de distribution, des transformations baissent la tension sur plusieurs niveaux de moyenne et basse tension. Nous parlons ainsi, de niveaux THT, HT, MT, BT, TBT [3].

Pour pouvoir assurer un bon fonctionnement de notre réseau de transport, les tensions au niveau de chaque nœud doivent être dans la marge de $\pm$ 5% de la valeur nominale, en d'autre terme, entre 0.95 et 1.05 pu (per – unit). Certains réseaux acceptent des marges arrivants jusqu'à $\pm$10% au maximum de la tension nominale de chaque nœud (entre 0.9 et 1.1 pu) ; le point 1.2.3 aborde le sujet de système d'unités relatives (le per – unit). Dans le cas échéant, des systèmes de compensation d'énergie réactive peuvent assurer le bon contrôle de la tension des nœuds.

1.2.2 Courbe de prévision de charge

La courbe de charge est fournie par le fournisseur d'énergie. Elle définit l'évolution de la consommation d'énergie pendant une durée (24 heure, une semaine, un mois …etc.), et donc de la puissance électrique que la charge consomme pendant une durée fixe. Elle est constituée d'un relevé de la consommation électrique du client à intervalles réguliers (le pas de mesure). Il s'agit d'un indicateur de puissance qui montre la succession de pics et de creux qui varie selon la puissance électrique nécessaire.

Le graphique ci-dessous représente le profil journalier du réseau de transport d'énergie électrique belge relevé de la part du Groupe ELIA, nous pouvons voir en bleu l'évolution en temps réel de la consommation totale réelle et en orange la prévision de charge estimée par le fournisseur d'énergie par le biais des modèles mathématiques de prédiction.

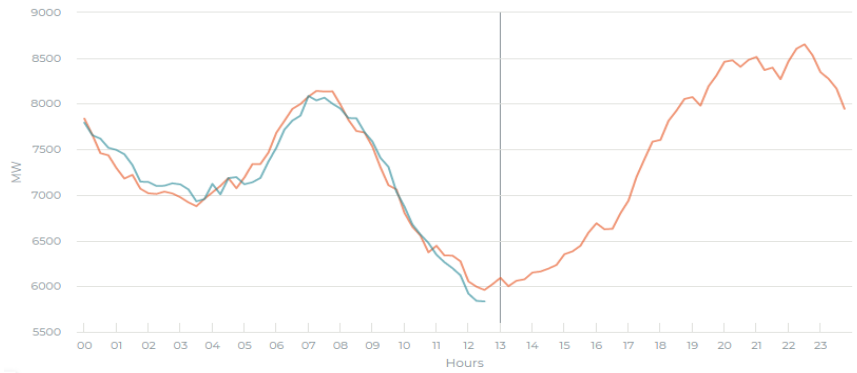


Figure 1.1 : profil journalier total du réseau de transport belge [7]

Les modèles de prédiction sont conçus pour pouvoir prédire la courbe de prévision de charge qui est supposée proche ou superposée à la courbe réelle de la consommation d'énergie électrique afin d'estimer la production d'énergie nécessaire pour satisfaire les clients consommateurs et assurer l'équilibre production – charge dans le réseau selon la saison, le jour, l'horaire, l'année …etc. Dans la pratique, le fournisseur d'énergie prélève une courbe de prévision de charge totale de tout le réseau, ce qui veut dire qu'à chaque instant donné, l'image du temps est une puissance active totale de consommation dans le réseau de transport dans tous les nœuds de charge.

Au niveau de la courbe de prévision de charge, chaque point présente un point de fonctionnement de notre réseau de transport obtenu après avoir effectué un calcul d'écoulement de puissance depuis un point de fonctionnement précédent. Si on décompose la courbe de charge pour chaque nœud, il sera plus aisé de calculer les paramètres spécifiant chaque charge à travers une méthode conventionnelle d'écoulement de puissance.

1.2.3 Système d'unités relatives (per-unit)

Les éléments d'un réseau de puissance fonctionnent dans des niveaux de tension où le kilovolt **(kV)** est l'unité qui convient le mieux pour les exprimer, tandis que le Méga – Volt – Ampères **(MVA)** exprime la puissance apparente.

Cependant toutes les grandeurs de tension, courant impédance et puissance sont exprimées en unités relatives (per-unit) par rapport aux grandeurs de base respectives données comme la puissance du réseau, etc. [6]. Les grandeurs en unités relatives sont calculées de la manière suivante :

$$\text{Valeur en pu} = \frac{\text{valeur réelle}}{\text{valeur de base choisie}}$$

Grâce à un choix judicieux des grandeurs de base, le schéma équivalent du transformateur est considérablement simplifié. Les tensions, courants, impédances et admittances convertis en per – unit ne changent pas quand elles sont rapportées du primaire au secondaire ou inversement. Cela donne un avantage considérable pour l'étude des réseaux importants comportant un grand nombre de transformateurs. D'autre part, les impédances en unités relatives des équipements électriques de même type sont comprises dans un intervalle de valeurs étroites. Ces grandeurs peuvent facilement faire l'objet de vérification de leur ordre de grandeur pour se prémunir de risque d'erreurs.

La méthode de calcul est définie de telle sorte que toutes les grandeurs au niveau de tous accès d'un réseau électrique soient proches de l'unité pour les bases appropriées. Cela facilite l'interprétation des résultats de calcul.

1.2.4 Schéma monophasé équivalent

La modélisation d'une ligne de transport triphasée se ramène à une transformation triphasée – monophasée, afin de simplifier l'étude.

Considérons le schéma d'une branche entre deux nœuds i et *j* :

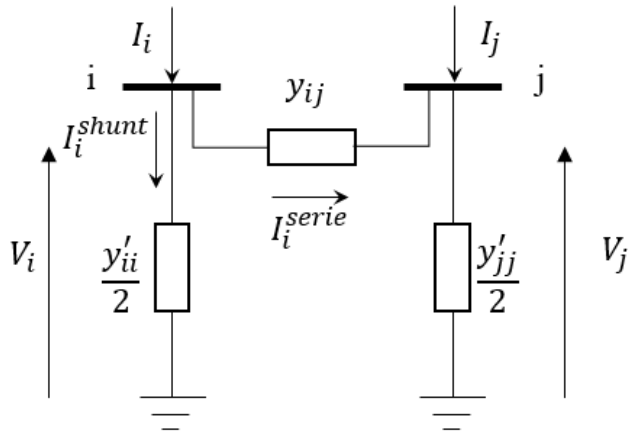


Figure 1.2 : Schéma monophasé équivalent en π [3]

Avec :

I_i : Courant injecté au nœud *i* ;

I_i^{serie} : Courant transitant du nœud *i* au nœud *j* ;

I_i^{shunt} : Courant de fuite au nœud *i* ;

y_{ij} : Admittance de la ligne *i-j* ;

$y'_{ii}/2$: Admittance shunt de nœuds i par rapport à la référence ;

V_i : Tension au nœud *i* ;

Les tensions, admittances et puissances au niveau d'un nœud i s'écrivent comme suit :

$$\overline{V_i} = e_i + j\, f_i = |V_i| e^{j\delta_i} \tag{1.1}$$

$$Y_{ij} = G_{ij} - j\, B_{ij} = \left|Y_{ij}\right| e^{j(\delta_i - \delta_j)} \tag{1.2}$$

$$S_i = \overline{V_i} \sum_{\substack{j=1 \\ j \neq i}}^{n} (\overline{V_i} - \overline{V_j}) Y_{ij} = P_i + j\, Q_i \tag{1.3}$$

Dans le cas où la ligne est raccordée à un transformateur à prise fixe de rapport de transformation de valeur 'a' (transformer tap), ce transformateur est considéré comme un dipôle de disposition [6], Il est représenté par un schéma monophasé équivalent similaire au modèle de la ligne :

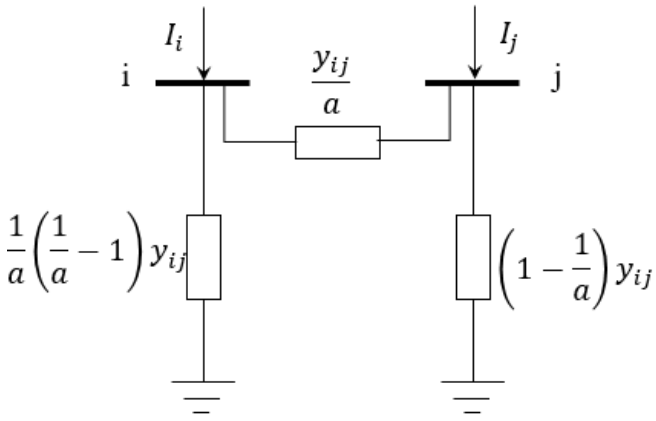


Figure 1.3 : Modèle monophasé d'un transformateur à prise fixe

1.2.5 Mise en équation du réseau et classement des nœuds

La matrice admittance schématise la topologie d'un réseau électrique, elle contient toutes les informations concernant notre réseau. Il peut être décrit sous la forme matricielle compacte ci – dessous :

$$[\boldsymbol{I}] = [\boldsymbol{Y}][\boldsymbol{V}] \tag{1.4}$$

En forme étalée :

$$\begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ \vdots \\ I_n \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & \cdots & \cdots & Y_{1n} \\ Y_{21} & Y_{22} & \cdots & \cdots & Y_{2n} \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ Y_{n1} & Y_{n2} & \cdots & \cdots & Y_{nn} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ \vdots \\ V_n \end{bmatrix} \tag{1.5}$$

Où **:**

[I] : Vecteur des courants injectés aux nœuds du réseau.

[V] : Vecteur des tensions aux nœuds du réseau.

[Y] : Matrice d'admittance du réseau.

n : Nombre de nœuds dans le réseau électrique.

La construction de la matrice admittance se fait algébriquement par remplir les éléments diagonaux et non – diagonaux depuis le modèle d'une ligne, selon les formulations qui suivent :

$$\begin{cases} Y_{ii} = \sum_{i=1}^{n} y_{ij} + \frac{y'_{ii}}{2} + y_{0i} \\ Y_{ij} = -y_{ij} \end{cases} \tag{1.6}$$

y_{i0} : Admittance shunt de réglage statique (compensation : FACTS, élément passif) raccordé à un nœud i.

Pour résoudre ce système d'équations, on doit imposer à chaque nœud soit la tension ou le courant injecté. Pratiquement ce problème est plus compliqué, car il faut définir les conditions de fonctionnement du réseau. Ces conditions affectent les grandeurs électriques relatives aux nœuds du réseau tel que : la puissance active P, la puissance réactive Q, l'amplitude $|V|$ et le déphasage δ de tension. On distingue alors trois types de nœuds :

Tableau 1. 1 : Paramètres spécifiés et paramètres inconnus par type de nœud

Type de nœud	Paramètres spécifiés	Paramètres inconnus
Nœuds de génération (PV)	P et $\|V\|$	Q et δ
Nœuds de charge (PQ)	P et Q	$\|V\|$ et δ
Nœud d'équilibre ou balancier (Slack bus)	$\|V\|$ et δ	P et Q

1.3 Méthodes de calcul d'écoulement de puissance

Le calcul d'écoulement de puissance est fondamental dans les réseaux électriques, leurs modèles mathématiques en général, sont présentés sous forme de systèmes d'équations non linéaires, la solution de ce type de systèmes peut être trouvée en utilisant des méthodes itératives [9]. Chaque méthode donne une solution approchée selon ses propriétés de convergence. Il existe également des méthodes directes qui convergent en un nombre réduit d'itérations par rapport aux méthodes itératives [9]. Ces méthodes directes ne sont pas souvent utilisées, destinées pour résoudre des systèmes de petite taille.

Le but de ces méthodes est de déterminer au niveau de chaque nœud 4 paramètres : l'amplitude de tension $|V|$, l'angle δ, les puissances actives P et réactives Q [8]. Dans un problème d'EP, les tensions des nœuds PQ et Slack sont initialisées à des valeurs de tensions données selon l'état de fonctionnement du réseau précédent, dans le cas échéant, l'initialisation se met à $V_i^0 = 1 + j\ 0\ pu$.

Les méthodes principales utilisées sont :

- Méthode de Gauss–Seidel.

- Méthode de Newton–Raphson.

- Méthode Découplée Rapide (FDLF).

1.3.1 Méthode de Gauss–Seidel (GS)

Cette méthode est inspirée de la méthode de Gauss et s'est évoluée dans le temps grâce à des travaux de recherche effectués. La méthode de Gauss–Seidel (GS) est utilisée pour la résolution des systèmes de petite taille.

La formule de la tension dans les nœuds PQ est donnée par :

$$V_i^{k+1} = \frac{1}{Y_{ii}}\left(\frac{P_i - jQ_i^k}{\left(V_i^k\right)^*} - \sum_{j=1}^{i-1} Y_{ij}V_j^{k+1} - \sum_{j=i+1}^{n} Y_{ij}V_j^k\right) \tag{1.7}$$

Les angles et les puissances seront déduits par le biais de la formule (1.7).

Dans chaque itération, le vecteur de tension obtenu qui remplacera l'ancienne valeur, c'est-à-dire : V_i^{k+1} remplace V_i^k. Le processus se termine une fois que :

$$\Delta V_i^{k+1} = V_i^{k+1} - V_i^k < \epsilon \tag{1.8}$$

Où ϵ représente la tolérance.

La méthode de Gauss–Seidel accélérée (par relaxation) est apparue, pour un gain de temps de calcul et un nombre d'itérations minimal, en introduisant un coefficient d'accélération α tel que :

$$V_{i_{acc}}^{k+1} = V_i^k + \alpha\, \Delta V_i^k \tag{1.9}$$

Le processus de calcul se fait en trois étapes :

Etape 1 : Nœuds PQ

Après la sélection des nœuds PQ, la formule générale pour déterminer l'amplitude $|V_i|$ et la phase δ_i de la tension au nœud i est l'équation (1.5) [8], d'où on aura :

$$|V_i|^{(k+1)} = abs\left(V_i^{k+1}\right) \tag{1.10}$$

$$\delta_i^{(k+1)} = \arg\left(V_i^{k+1}\right) \tag{1.11}$$

Avec i étant le numéro du nœud PQ.

Etape 2 : Nœuds PV

La puissance active et l'amplitude de tension sont connues pour les nœuds de génération, donc il reste qu'à déterminer la puissance réactive et la phase de tension.

La formule du courant de génération des nœuds PV i est :

$$I_{g_i} = \frac{P_i - jQ_i}{V_i^*} = Y_{i1}V_1 + Y_{i2}V_2 + \cdots + Y_{in}V_n \tag{1.12}$$

Ce qui implique :

$$P_i - jQ_i^{k+1} = V_i^{k^*}\left[Y_{i1}V_1 \sum_{j=1}^{i-1} Y_{ij}V_j^{k+1} - \sum_{j=i+1}^{n} Y_{ij}V_j^k\right] \quad (1.13)$$

La partie imaginaire de l'équation précédente est la puissance réactive du nœud de génération i :

$$Q_i^{k+1} = -Imag\left[V_i^{k^*}\left[Y_{i1}V_1 \sum_{j=1}^{i-1} Y_{ij}V_j^{k+1} - \sum_{j=i+1}^{n} Y_{ij}V_j^k\right]\right] \quad (1.14)$$

En connaissant maintenant la puissance apparente du nœud PV, nous pourrons directement tirer la valeur de la phase de tension via la formule ci – dessous :

$$\delta_i^{(k+1)} = Arg\left[\frac{1}{Y_{ii}}\left(\frac{P_i^{spec} - jQ_i^k}{\left(V_i^k\right)^*} - \sum_{j=1}^{i-1} Y_{ij}V_j^{k+1} - \sum_{j=i+1}^{n} Y_{ij}V_i^k\right)\right] \quad (1.15)$$

Etape 3: Nœud balancier (Slack bus)

Une fois que toutes les tensions des nœuds PV et PQ sont solutionnés, la puissance du nœud Slack bus est calculée :

$$\frac{P_1 - jQ_1}{V_1^*} = Y_{11}V_1 + Y_{12}V_2 + Y_{13}V_3 + \cdots + Y_{1n}V_n = \sum_{j=1}^{n} Y_{1j}\,V_j \quad (1.16)$$

Où :

$$P_1 - jQ_1 = Y_{11}V_1V_1^* + Y_{12}V_2V_1^* + Y_{13}V_3V_1^* + \cdots + Y_{1n}V_nV_1^* \quad (1.17)$$

On trouve bien donc :

$$P_1 = real\left(V_1^* \sum_{j=1}^{n} Y_{1j}\,V_j\right) \quad (1.18)$$

$$Q_1 = -Imag\left(V_1^* \sum_{j=1}^{n} Y_{1j}\,V_j\right) \quad (1.19)$$

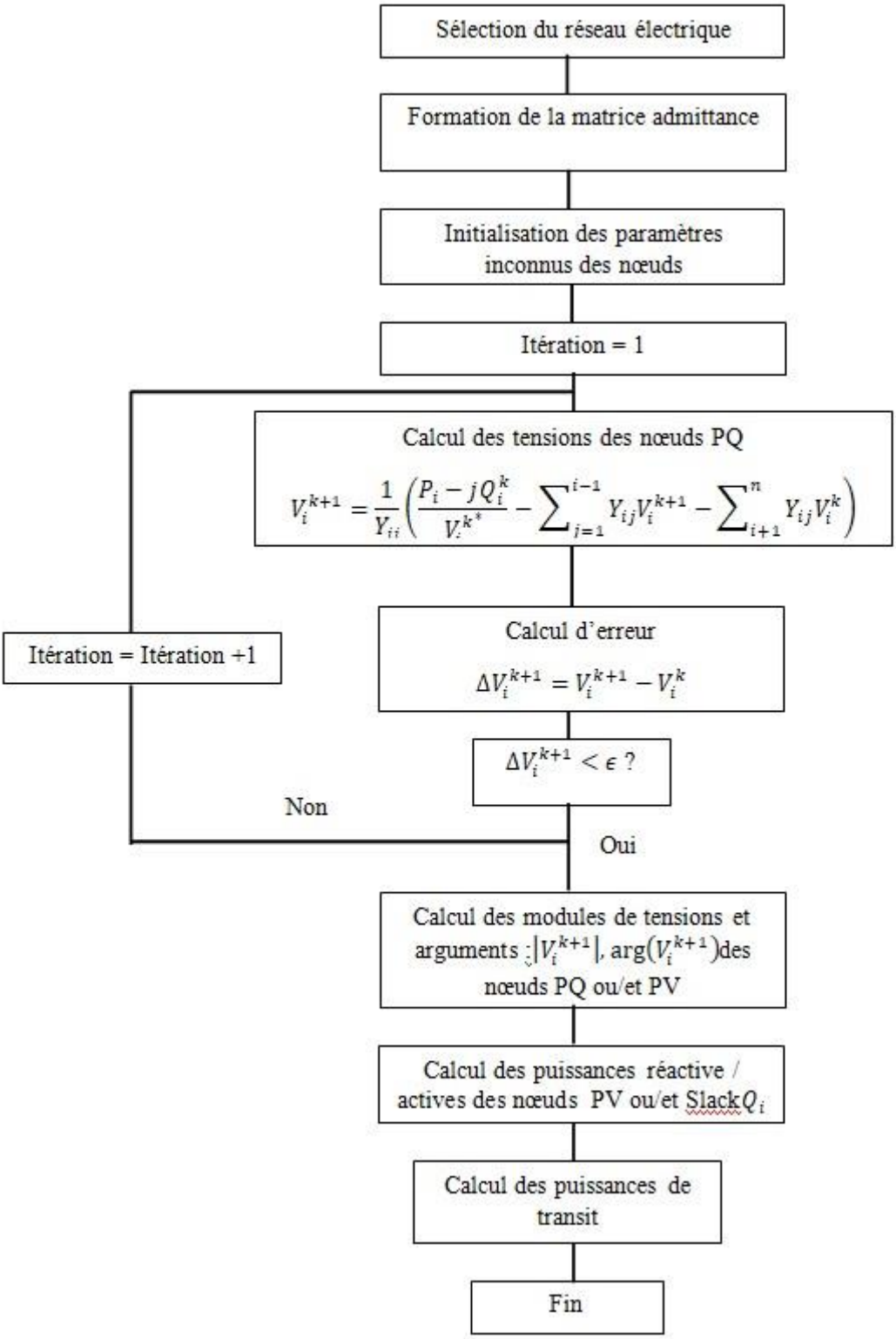


Figure 1.4: Organigramme descriptif de la méthode de Gauss-Seidel

1.3.2 Méthode de Newton–Raphson (NR)

La méthode de Newton-Raphson est très utilisée en pratique pour l'écoulement de puissance et pour n'importe quel système, de grande ou petite taille. Le choix des valeurs initiales de tension est important. Plus, elles sont proches de la solution, plus vite est la convergence.

Pour un réseau de *n* nœuds, le système est de dimension de 2(*n*-1) x 2(*n* -1).

Le problème se transforme en cette forme :

$$X^{k+1} = X^k - J^{-1}(X^k)\, F(X^k) \tag{1.20}$$

Comme il s'agit d'un calcul avec des valeurs complexes, nous avons deux façons pour résoudre le problème :

NR en coordonnées rectangulaires

Il s'agit d'écrire les nombres complexes sous forme algébrique afin de séparer les parties réelles de celle d'imaginaires et rendre le calcul dans l'espace réel. Les formulations des puissances, tensions et d'admittance seront :

$$P_i = \sum_{j=1}^{n} \left[\mathrm{e_i} \left(G_{ij} e_j + B_{ij} f_j \right) + f_i \left(G_{ij} f_j - B_{ij} e_j \right) \right] \tag{1.21}$$

$$Q_i = \sum_{j=1}^{n} \left[f_i \left(G_{ij} e_j + B_{ij} f_j \right) - e_j \left(G_{ij} f_j - B_{ij} e_j \right) \right] \tag{1.22}$$

$$|V_i|^2 = e_i^2 + f_i^2 \tag{1.23}$$

$$\Delta P_i^{(k)} = P_{i\,spec} - P_{i\,cal} \tag{1.24}$$

$$\Delta Q_i^{(k)} = Q_{i\,spec} - Q_{i\,cal} \tag{1.25}$$

$$\Delta |V_i|^2 = \left| V_{i\,spec} \right|^2 - |V_{i\,cal}|^2 \tag{1.26}$$

L'erreur s'écrit en forme compacte :

$$\begin{bmatrix} \Delta e \\ \Delta f \end{bmatrix} = - J^{-1} \begin{bmatrix} \Delta P \\ \Delta Q \\ \Delta |V|^2 \end{bmatrix} \tag{1.27}$$

Tel que :

$$J = \begin{bmatrix} J_1 & J_2 \\ J_3 & J_4 \\ J_5 & J_6 \end{bmatrix} \tag{1.28}$$

$$J = \begin{bmatrix} \dfrac{\partial P}{\partial e} & \dfrac{\partial P}{\partial f} \\ \dfrac{\partial Q}{\partial e} & \dfrac{\partial Q}{\partial f} \\ \dfrac{\partial |V|^2}{\partial e} & \dfrac{\partial |V|^2}{\partial f} \end{bmatrix} \tag{1.29}$$

$Pour\ J_1$:

$$\frac{\partial P_i}{\partial e_i} = 2e_i G_{ii} + \sum_{j \neq i}^{n} \left(e_i G_{ij} + f_j B_{ij}\right) \tag{1.30}$$

$$\frac{\partial P_i}{\partial e_j} = e_i G_{ij} + f_j B_{ij} \tag{1.31}$$

$Pour\ J_2$:

$$\frac{\partial P_i}{\partial f_i} = 2f_i B_{ii} + \sum_{j \neq i}^{n} \left(f_i G_{ij} - e_j B_{ij}\right) \tag{1.32}$$

$$\frac{\partial P_i}{\partial f_j} = f_i G_{ij} - e_j B_{ij} \tag{1.33}$$

$Pour\ J_3$:

$$\frac{\partial Q_i}{\partial e_i} = 2e_i B_{ii} - \sum_{j \neq i}^{n} \left(f_i G_{ij} - f_j B_{ij}\right) \tag{1.34}$$

$$\frac{\partial Q_i}{\partial e_j} = e_i B_{ij} + f_j G_{ij} \tag{1.35}$$

$Pour\ J_4$:

$$\frac{\partial Q_i}{\partial f_i} = 2f_i B_{ii} + \sum_{j \neq i}^{n} \left(e_i G_{ij} + f_j B_{ij}\right) \tag{1.36}$$

$$\frac{\partial Q_i}{\partial f_j} = -e_i G_{ij} + f_j B_{ij} \tag{1.37}$$

$Pour\ J_5$:

$$\frac{\partial |V_i|^2}{\partial e_i} = 2e_i \tag{1.38}$$

$$\frac{\partial |V_i|^2}{\partial e_j} = 0 \tag{1.39}$$

$Pour\ J_6$:

$$\frac{\partial |V_i|^2}{\partial f_i} = 2f_i \tag{1.40}$$

$$\frac{\partial |V_i|^2}{\partial f_j} = 0 \tag{1.41}$$

Les valeurs obtenues sont utilisées pour estimer les tensions des nœuds pour l'itération suivante :

$$\begin{bmatrix} e_i^{k+1} \\ f_i^{k+1} \end{bmatrix} = \begin{bmatrix} e_i^k \\ f_i^k \end{bmatrix} + \begin{bmatrix} \Delta e_i^k \\ \Delta f_i^k \end{bmatrix} \tag{1.42}$$

NR en coordonnées polaires

Le principe est exactement le même sauf que les variables seront donc, l'amplitude de tension de chaque nœud $|V_i|$ et son argument δ_i. L'équation (22) devient :

$$V_i = |V_i| e^{j\delta_i} \tag{1.43}$$

$$Y_{ik} = |Y_{ik}| e^{j\theta ik} \tag{1.44}$$

$$P_i = |V_i| \sum_{k=1}^{n} |V_k||Y_{ik}| \cos(\theta_{ik} + \delta_k - \delta_i) \tag{1.45}$$

$$Q_i = -|V_i| \sum_{k=1}^{n} |V_k||Y_{ik}| \sin(\theta_{ik} + \delta_k - \delta_i) \tag{1.46}$$

L'erreur s'écrit en forme compacte :

$$\begin{bmatrix} \Delta|V| \\ \Delta\delta \end{bmatrix} = -J^{-1} \begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} \tag{1.47}$$

$$J = \begin{bmatrix} \dfrac{\partial P}{\partial |V|} & \dfrac{\partial P}{\partial \delta} \\ \dfrac{\partial Q}{\partial |V|} & \dfrac{\partial Q}{\partial \delta} \end{bmatrix} \tag{1.48}$$

$$\begin{bmatrix} |V_i|^{k+1} \\ \delta_i^{k+1} \end{bmatrix} = \begin{bmatrix} |V_i|^k \\ \delta_i^k \end{bmatrix} + \begin{bmatrix} \Delta|V_i|^k \\ \Delta\delta_i^k \end{bmatrix} \tag{1.49}$$

Le processus est répété jusqu'à que les valeurs des mismatches ($\Delta P_i, \Delta Q_i et\ \Delta|V_i|$) soient inférieures à une certaine tolérance fixée [8].

1.3.3 Méthode Découplée Rapide (FDLF)

En anglais : Fast Decoupled Load Flow, c'est une méthode améliorée et simplifiée à base de Newton–Raphson [8]. A partir de la méthode de NR qui a montré que la moitié des éléments de la matrice jacobienne en coordonnées polaires représentent un couplage faible [4], des approximations sont faites pour simplifier le modèle.

L'équation de découplage donne :

$$\begin{bmatrix} \frac{\Delta P}{|V|} \\ \frac{\Delta Q}{|V|} \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & C \end{bmatrix} \begin{bmatrix} \Delta\delta \\ \Delta|V| \end{bmatrix} \tag{1.50}$$

A et *C* étant des sous-matrices jacobiennes. Le découplage de Newton réduit la non-linéarité du système [4].

La FDLF est venue par des mises à jour et des approximations. L'équation (1.50) devient :

$$\begin{bmatrix} \frac{\Delta P}{|V|} \\ \frac{\Delta Q}{|V|} \end{bmatrix} = \begin{bmatrix} -B' & 0 \\ 0 & -B'' \end{bmatrix} \begin{bmatrix} \Delta\delta \\ \Delta|V| \end{bmatrix} \tag{1.51}$$

B' et B'' représentent des sous matrices de la matrice susceptance *B* après avoir fixé certaines hypothèses.

Le calcul avec cette méthode est similaire de point de vue algorithmique avec la méthode NR en coordonnées polaires, le découplage assure que la matrice jacobienne soit creuse à moitié (remplisse par des zéros pour des raisons de simplification, elle est généralement utilisée dans le cas de problème de surcharge (outage) ou pour des régions qui disposent des lignes dont le quotient inductance – résistance très faible (R/X >>>0).

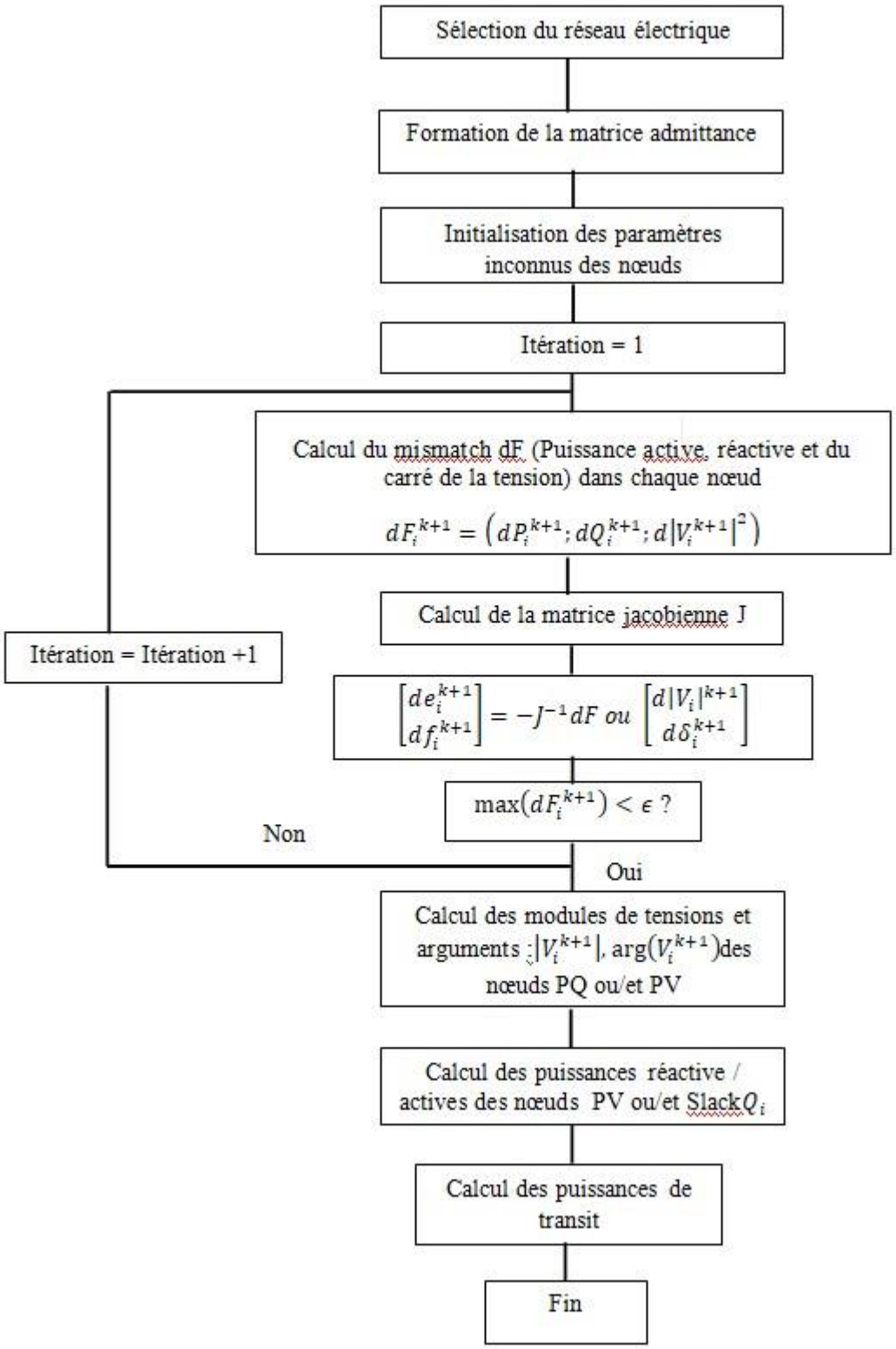


Figure 1.5: Organigramme descriptif de la méthode Newton-Raphson (NR)

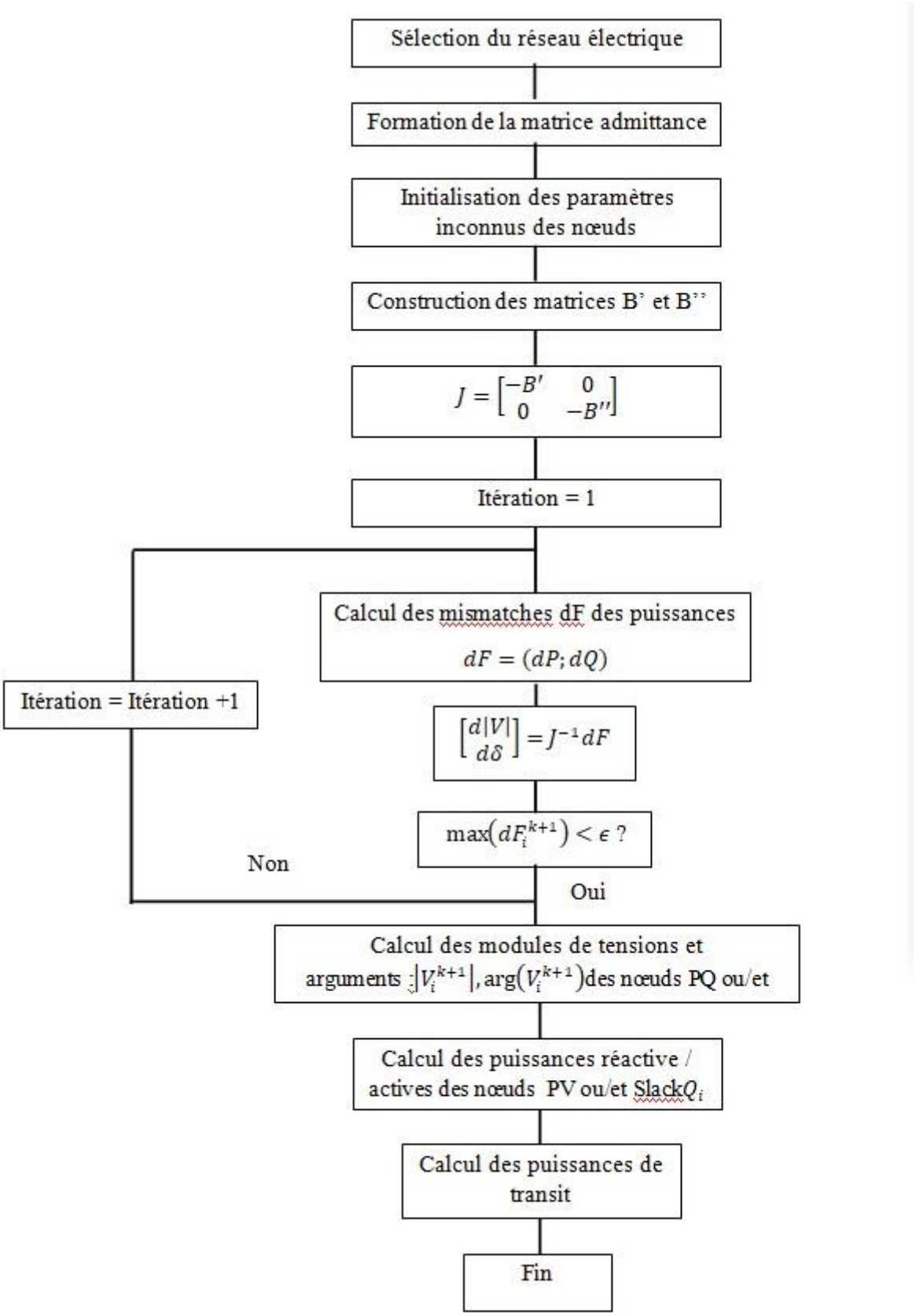


Figure 1.6 : Organigramme descriptif de la méthode Fast Decoupled (FDLF)

1.4 Performances sur les machines à calcul

Les machines à calcul sont limitées par leur stockage et rapidité, mais la technologie en électronique évolue très vide et augmente considérablement leur performance. Une application de calcul numérique destinée pour l'étude d'un système physique doit vérifier les critères suivants : rapidité, stockage minimale, fiabilité, flexibilité et simplicité.

Malheureusement, chaque méthode a ses avantages et ses inconvénients, et cela dépend du type de système. La méthode de Gauss–Seidel converge linéairement tandis que Newton–Raphson converge de façon quadratique, ce qui fait que cette dernière converge rapidement, d'ailleurs c'est la méthode la plus utilisée en pratique. L'inconvénient de cette dernière nécessite un espace de mémoire plus grand [10].

Le stockage nécessaire en résolvant un système de réseau électrique par la méthode FDLF est réduit de moitié par rapport à la méthode de Newton–Raphson, sa convergence est également rapide, mais l'application de cette méthode est généralement utilisée dans quelques nœuds qui présentent une convergence lente par Newton–Raphson, les points de surcharges, les systèmes présentant un quotient R/X important [9].

La méthode de Gauss – Seidel nécessite un nombre d'itération assez grand en raison de sa faible convergence, le calcul devient très lent pour une bonne précision pour trouver la bonne valeur de la tension, et donc ça demande plusieurs itérations pour atteindre la solution. Pour la méthode de NR et sa dérivée FDLF, le nombre d'itération est minimum dû à la convergence rapide de ces méthodes.

1.5 Simulations

Afin de finaliser ce chapitre, nous allons s'intéresser à l'application des méthodes conventionnelles déjà mentionnées pour effectuer un calcul d'écoulement de puissance sur des réseaux électriques standards qu'on les a déjà pris comme des exemples de simulation et validation. Leurs données sont mentionnées dans l'annexe, elles représentent les informations de chaque nœud d'une part, et les données de lignes d'autre part. Elles représentent également un point de fonctionnement du réseau électrique à un instant donné. Le choix a été effectué pour le réseau de 14 nœuds (IEEE 14), afin de montrer la validité des quatre méthodes pour n'importe quel point de fonctionnement et pour n'importe quelle topologie du réseau.

Commençons la résolution par la méthode de Gauss – Seidel, ci – dessous les résultats obtenus donnant le nouveau point de fonctionnement de chaque nœud (Load flow table) ainsi les puissances de transit (Line flow table) :

1.5.1 Ecoulement de puissance par la méthode de Gauss – Seidel

Grid data: 14 bus | Method Choice: GS | Run Load Flow | Run Line Flow | Return

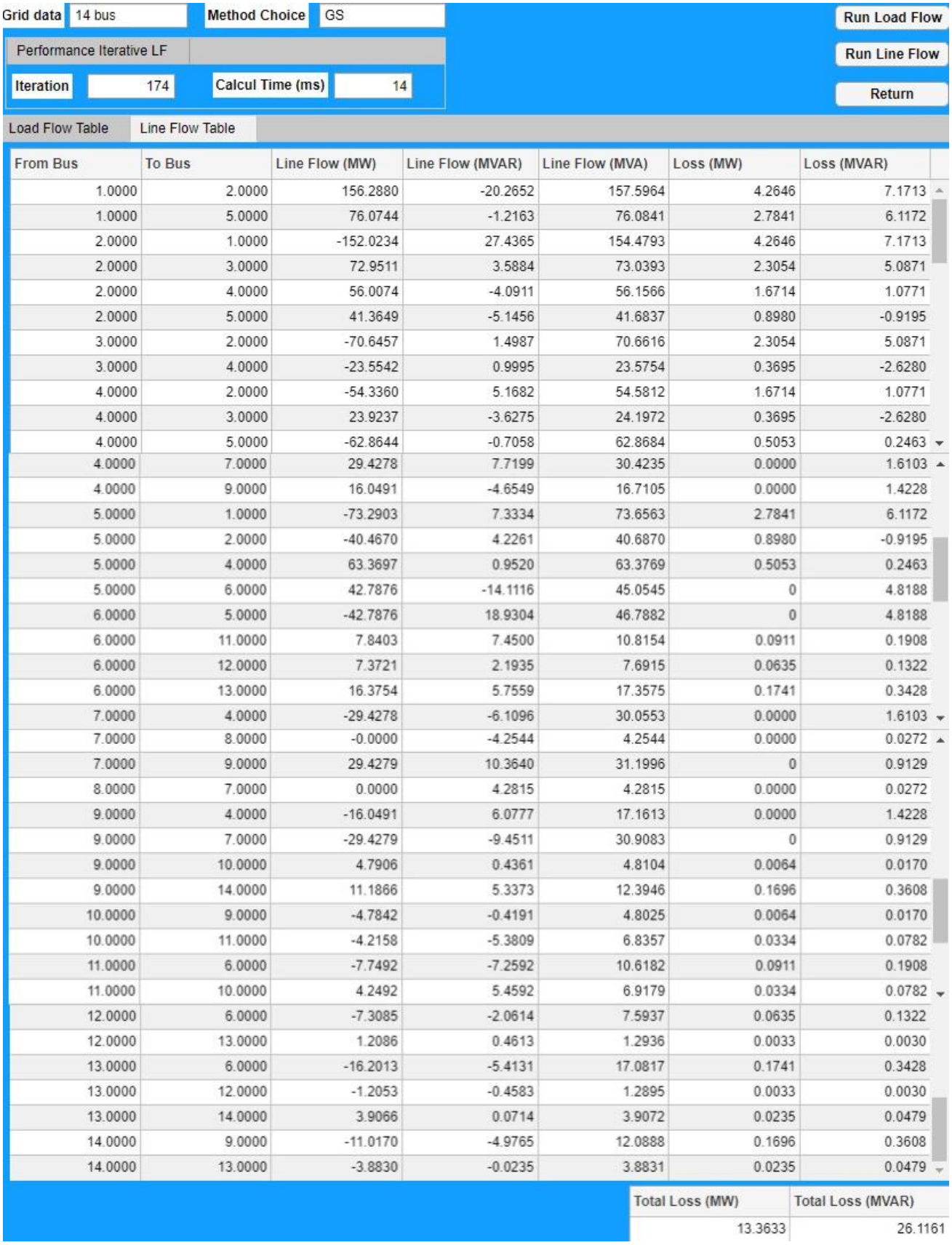
Performance Iterative LF

Iteration: 174 | Calcul Time (ms): 14

Load Flow Table | Line Flow Table

Bus No	Code No	Voltage Magnitude [pu]	Angle [rad]	Load [MW]	Load [MVAR]	Generation [MW]	Generation [MVAR]
1.0000	1.0000	1.0600	0	0	0	232.3623	-21.4814
2.0000	2.0000	1.0450	-0.0866	21.7000	12.7000	40.0000	34.4881
3.0000	2.0000	1.0100	-0.2212	94.2000	19.0000	0	21.4982
4.0000	0	1.0218	-0.1804	47.8000	-3.9000	0	0
5.0000	0	1.0304	-0.1552	7.6000	1.6000	0	0
6.0000	2.0000	1.0700	-0.2532	11.2000	7.5000	0	41.8298
7.0000	0	1.0831	-0.2322	0	0	0	0
8.0000	2.0000	1.0900	-0.2322	0	0	0	4.2815
9.0000	0	1.0730	-0.2601	29.5000	16.6000	0	0
10.0000	0	1.0712	-0.2635	9.0000	5.8000	0	0
11.0000	0	1.0841	-0.2603	3.5000	1.8000	0	0
12.0000	0	1.0564	-0.2675	6.1000	1.6000	0	0
13.0000	0	1.0530	-0.2687	13.5000	5.8000	0	0
14.0000	0	1.0465	-0.2810	14.9000	5.0000	0	0

Figure 1.7: Ecoulement de Puissance du réseau IEEE 14 nœuds par GS



Grid data: 14 bus | Method Choice: GS | Run Load Flow | Run Line Flow | Return

Performance Iterative LF

Iteration: 174 | Calcul Time (ms): 14

Load Flow Table | Line Flow Table

From Bus	To Bus	Line Flow (MW)	Line Flow (MVAR)	Line Flow (MVA)	Loss (MW)	Loss (MVAR)
1.0000	2.0000	156.2880	-20.2652	157.5964	4.2646	7.1713
1.0000	5.0000	76.0744	-1.2163	76.0841	2.7841	6.1172
2.0000	1.0000	-152.0234	27.4365	154.4793	4.2646	7.1713
2.0000	3.0000	72.9511	3.5884	73.0393	2.3054	5.0871
2.0000	4.0000	56.0074	-4.0911	56.1566	1.6714	1.0771
2.0000	5.0000	41.3649	-5.1456	41.6837	0.8980	-0.9195
3.0000	2.0000	-70.6457	1.4987	70.6616	2.3054	5.0871
3.0000	4.0000	-23.5542	0.9995	23.5754	0.3695	-2.6280
4.0000	2.0000	-54.3360	5.1682	54.5812	1.6714	1.0771
4.0000	3.0000	23.9237	-3.6275	24.1972	0.3695	-2.6280
4.0000	5.0000	-62.8644	-0.7058	62.8684	0.5053	0.2463
4.0000	7.0000	29.4278	7.7199	30.4235	0.0000	1.6103
4.0000	9.0000	16.0491	-4.6549	16.7105	0.0000	1.4228
5.0000	1.0000	-73.2903	7.3334	73.6563	2.7841	6.1172
5.0000	2.0000	-40.4670	4.2261	40.6870	0.8980	-0.9195
5.0000	4.0000	63.3697	0.9520	63.3769	0.5053	0.2463
5.0000	6.0000	42.7876	-14.1116	45.0545	0	4.8188
6.0000	5.0000	-42.7876	18.9304	46.7882	0	4.8188
6.0000	11.0000	7.8403	7.4500	10.8154	0.0911	0.1908
6.0000	12.0000	7.3721	2.1935	7.6915	0.0635	0.1322
6.0000	13.0000	16.3754	5.7559	17.3575	0.1741	0.3428
7.0000	4.0000	-29.4278	-6.1096	30.0553	0.0000	1.6103
7.0000	8.0000	-0.0000	-4.2544	4.2544	0.0000	0.0272
7.0000	9.0000	29.4279	10.3640	31.1996	0	0.9129
8.0000	7.0000	0.0000	4.2815	4.2815	0.0000	0.0272
9.0000	4.0000	-16.0491	6.0777	17.1613	0.0000	1.4228
9.0000	7.0000	-29.4279	-9.4511	30.9083	0	0.9129
9.0000	10.0000	4.7906	0.4361	4.8104	0.0064	0.0170
9.0000	14.0000	11.1866	5.3373	12.3946	0.1696	0.3608
10.0000	9.0000	-4.7842	-0.4191	4.8025	0.0064	0.0170
10.0000	11.0000	-4.2158	-5.3809	6.8357	0.0334	0.0782
11.0000	6.0000	-7.7492	-7.2592	10.6182	0.0911	0.1908
11.0000	10.0000	4.2492	5.4592	6.9179	0.0334	0.0782
12.0000	6.0000	-7.3085	-2.0614	7.5937	0.0635	0.1322
12.0000	13.0000	1.2086	0.4613	1.2936	0.0033	0.0030
13.0000	6.0000	-16.2013	-5.4131	17.0817	0.1741	0.3428
13.0000	12.0000	-1.2053	-0.4583	1.2895	0.0033	0.0030
13.0000	14.0000	3.9066	0.0714	3.9072	0.0235	0.0479
14.0000	9.0000	-11.0170	-4.9765	12.0888	0.1696	0.3608
14.0000	13.0000	-3.8830	-0.0235	3.8831	0.0235	0.0479

Total Loss (MW)	Total Loss (MVAR)
13.3633	26.1161

Figure 1.8: Calcul de puissance de transit par la méthode GS

1.5.2 Ecoulement de puissance avec la méthode Newton Raphson – Coordonnées Rectangulaires

Grid data: 14 bus | Method Choice: NR - CR | Run Load Flow | Run Line Flow | Return

Performance Iterative LF

Iteration: 5 | Calcul Time (ms): 1

Load Flow Table | Line Flow Table

Bus No	Code No	Voltage Magnitude [pu]	Angle [rad]	Load [MW]	Load [MVAR]	Generation [MW]	Generation [MVAR]
1.0000	1.0000	1.0600	0	0	0	232.3633	-21.4816
2.0000	2.0000	1.0450	-0.0866	21.7000	12.7000	40.0000	34.4884
3.0000	2.0000	1.0100	-0.2212	94.2000	19.0000	0	21.4983
4.0000	0	1.0218	-0.1804	47.8000	-3.9000	0	0
5.0000	0	1.0304	-0.1552	7.6000	1.6000	0	0
6.0000	2.0000	1.0700	-0.2532	11.2000	7.5000	0	41.8298
7.0000	0	1.0831	-0.2322	0	0	0	0
8.0000	2.0000	1.0900	-0.2322	0	0	0	4.2816
9.0000	0	1.0730	-0.2601	29.5000	16.6000	0	0
10.0000	0	1.0712	-0.2635	9.0000	5.8000	0	0
11.0000	0	1.0841	-0.2603	3.5000	1.8000	0	0
12.0000	0	1.0564	-0.2675	6.1000	1.6000	0	0
13.0000	0	1.0530	-0.2687	13.5000	5.8000	0	0
14.0000	0	1.0465	-0.2810	14.9000	5.0000	0	0

Figure 1.9: Ecoulement de Puissance du réseau IEEE 14 nœuds par NRCR

Grid data: 14 bus | Method Choice: NR - CR | Run Load Flow | Run Line Flow | Return

Performance Iterative LF

Iteration: 5 | Calcul Time (ms): 1

Load Flow Table | Line Flow Table

From Bus	To Bus	Line Flow (MW)	Line Flow (MVAR)	Line Flow (MVA)	Loss (MW)	Loss (MVAR)
1.0000	2.0000	156.2887	-20.2653	157.5971	4.2647	7.1714
1.0000	5.0000	76.0747	-1.2162	76.0844	2.7841	6.1173
2.0000	1.0000	-152.0240	27.4367	154.4800	4.2647	7.1714
2.0000	3.0000	72.9512	3.5884	73.0394	2.3054	5.0872
2.0000	4.0000	56.0076	-4.0911	56.1569	1.6715	1.0771
2.0000	5.0000	41.3651	-5.1456	41.6839	0.8980	-0.9195
3.0000	2.0000	-70.6458	1.4988	70.6617	2.3054	5.0872
3.0000	4.0000	-23.5542	0.9995	23.5754	0.3695	-2.6280
4.0000	2.0000	-54.3362	5.1682	54.5814	1.6715	1.0771
4.0000	3.0000	23.9236	-3.6275	24.1971	0.3695	-2.6280
4.0000	5.0000	-62.8646	-0.7057	62.8686	0.5053	0.2463
4.0000	7.0000	29.4280	7.7199	30.4237	0	1.6104
4.0000	9.0000	16.0492	-4.6549	16.7106	0.0000	1.4228
5.0000	1.0000	-73.2906	7.3335	73.6566	2.7841	6.1173
5.0000	2.0000	-40.4671	4.2261	40.6872	0.8980	-0.9195
5.0000	4.0000	63.3699	0.9520	63.3771	0.5053	0.2463
5.0000	6.0000	42.7878	-14.1116	45.0548	0	4.8189
6.0000	5.0000	-42.7878	18.9305	46.7884	0	4.8189
6.0000	11.0000	7.8403	7.4500	10.8154	0.0911	0.1908
6.0000	12.0000	7.3721	2.1935	7.6915	0.0635	0.1322
6.0000	13.0000	16.3754	5.7559	17.3575	0.1741	0.3428
7.0000	4.0000	-29.4280	-6.1095	30.0555	0	1.6104
7.0000	8.0000	-0.0000	-4.2544	4.2544	0	0.0272
7.0000	9.0000	29.4280	10.3639	31.1997	0	0.9129
8.0000	7.0000	0.0000	4.2816	4.2816	0	0.0272
9.0000	4.0000	-16.0492	6.0777	17.1614	0.0000	1.4228
9.0000	7.0000	-29.4280	-9.4511	30.9084	0	0.9129
9.0000	10.0000	4.7906	0.4361	4.8104	0.0064	0.0170
9.0000	14.0000	11.1866	5.3373	12.3946	0.1696	0.3608
10.0000	9.0000	-4.7842	-0.4191	4.8025	0.0064	0.0170
10.0000	11.0000	-4.2158	-5.3809	6.8357	0.0334	0.0782
11.0000	6.0000	-7.7492	-7.2591	10.6182	0.0911	0.1908
11.0000	10.0000	4.2492	5.4591	6.9179	0.0334	0.0782
12.0000	6.0000	-7.3086	-2.0613	7.5937	0.0635	0.1322
12.0000	13.0000	1.2086	0.4613	1.2936	0.0033	0.0030
13.0000	6.0000	-16.2013	-5.4130	17.0817	0.1741	0.3428
13.0000	12.0000	-1.2053	-0.4583	1.2895	0.0033	0.0030
13.0000	14.0000	3.9066	0.0714	3.9072	0.0235	0.0479
14.0000	9.0000	-11.0170	-4.9765	12.0888	0.1696	0.3608
14.0000	13.0000	-3.8830	-0.0235	3.8831	0.0235	0.0479

Total Loss (MW)	Total Loss (MVAR)
13.3633	26.1165

Figure 1.10: Calcul de puissance de transit par la méthode NRCR

1.5.3 Ecoulement de puissance avec la méthode Newton Raphson – Coordonnées Polaires

Grid data: 14 bus | Method Choice: NR - CP | Run Load Flow | Run Line Flow | Return

Performance Iterative LF

Iteration: 4 | Calcul Time (ms): 1

Load Flow Table | Line Flow Table

Bus No	Code No	Voltage Magnitude [pu]	Angle [rad]	Load [MW]	Load [MVAR]	Generation [MW]	Generation [MVAR]
1.0000	1.0000	1.0600	0	0	0	232.3633	-21.4816
2.0000	2.0000	1.0450	-0.0866	21.7000	12.7000	40.0000	34.4884
3.0000	2.0000	1.0100	-0.2212	94.2000	19.0000	0	21.4983
4.0000	0	1.0218	-0.1804	47.8000	-3.9000	0	0
5.0000	0	1.0304	-0.1552	7.6000	1.6000	0	0
6.0000	2.0000	1.0700	-0.2532	11.2000	7.5000	0	41.8298
7.0000	0	1.0831	-0.2322	0	0	0	0
8.0000	2.0000	1.0900	-0.2322	0	0	0	4.2816
9.0000	0	1.0730	-0.2601	29.5000	16.6000	0	0
10.0000	0	1.0712	-0.2635	9.0000	5.8000	0	0
11.0000	0	1.0841	-0.2603	3.5000	1.8000	0	0
12.0000	0	1.0564	-0.2675	6.1000	1.6000	0	0
13.0000	0	1.0530	-0.2687	13.5000	5.8000	0	0
14.0000	0	1.0465	-0.2810	14.9000	5.0000	0	0

Figure 1.11: Ecoulement de Puissance du réseau IEEE 14 nœuds par NRCP

Grid data: 14 bus | Method Choice: NR - CP | Run Load Flow | Run Line Flow | Return

Performance Iterative LF

Iteration: 4 | Calcul Time (ms): 1

Load Flow Table | Line Flow Table

From Bus	To Bus	Line Flow (MW)	Line Flow (MVAR)	Line Flow (MVA)	Loss (MW)	Loss (MVAR)
1.0000	2.0000	156.2887	-20.2653	157.5971	4.2647	7.1714
1.0000	5.0000	76.0747	-1.2162	76.0844	2.7841	6.1173
2.0000	1.0000	-152.0240	27.4367	154.4800	4.2647	7.1714
2.0000	3.0000	72.9512	3.5884	73.0394	2.3054	5.0872
2.0000	4.0000	56.0076	-4.0911	56.1569	1.6715	1.0771
2.0000	5.0000	41.3651	-5.1456	41.6839	0.8980	-0.9195
3.0000	2.0000	-70.6458	1.4988	70.6617	2.3054	5.0872
3.0000	4.0000	-23.5542	0.9995	23.5754	0.3695	-2.6280
4.0000	2.0000	-54.3362	5.1682	54.5814	1.6715	1.0771
4.0000	3.0000	23.9236	-3.6275	24.1971	0.3695	-2.6280
4.0000	5.0000	-62.8646	-0.7057	62.8686	0.5053	0.2463
4.0000	7.0000	29.4280	7.7199	30.4237	-0.0000	1.6104
4.0000	9.0000	16.0492	-4.6549	16.7106	-0.0000	1.4228
5.0000	1.0000	-73.2906	7.3335	73.6566	2.7841	6.1173
5.0000	2.0000	-40.4671	4.2261	40.6872	0.8980	-0.9195
5.0000	4.0000	63.3699	0.9520	63.3771	0.5053	0.2463
5.0000	6.0000	42.7878	-14.1116	45.0548	0	4.8189
6.0000	5.0000	-42.7878	18.9305	46.7884	0	4.8189
6.0000	11.0000	7.8403	7.4500	10.8154	0.0911	0.1908
6.0000	12.0000	7.3721	2.1935	7.6915	0.0635	0.1322
6.0000	13.0000	16.3754	5.7559	17.3575	0.1741	0.3428
7.0000	4.0000	-29.4280	-6.1095	30.0555	-0.0000	1.6104
7.0000	8.0000	0.0000	-4.2544	4.2544	0	0.0272
7.0000	9.0000	29.4280	10.3639	31.1997	0	0.9129
8.0000	7.0000	-0.0000	4.2816	4.2816	0	0.0272
9.0000	4.0000	-16.0492	6.0777	17.1614	-0.0000	1.4228
9.0000	7.0000	-29.4280	-9.4511	30.9084	0	0.9129
9.0000	10.0000	4.7906	0.4361	4.8104	0.0064	0.0170
9.0000	14.0000	11.1866	5.3373	12.3946	0.1696	0.3608
10.0000	9.0000	-4.7842	-0.4191	4.8025	0.0064	0.0170
10.0000	11.0000	-4.2158	-5.3809	6.8357	0.0334	0.0782
11.0000	6.0000	-7.7492	-7.2591	10.6182	0.0911	0.1908
11.0000	10.0000	4.2492	5.4591	6.9179	0.0334	0.0782
12.0000	6.0000	-7.3086	-2.0613	7.5937	0.0635	0.1322
12.0000	13.0000	1.2086	0.4613	1.2936	0.0033	0.0030
13.0000	6.0000	-16.2013	-5.4130	17.0817	0.1741	0.3428
13.0000	12.0000	-1.2053	-0.4583	1.2895	0.0033	0.0030
13.0000	14.0000	3.9066	0.0714	3.9072	0.0235	0.0479
14.0000	9.0000	-11.0170	-4.9765	12.0888	0.1696	0.3608
14.0000	13.0000	-3.8830	-0.0235	3.8831	0.0235	0.0479

Total Loss (MW)	Total Loss (MVAR)
13.3633	26.1165

Figure 1.12 : Calcul de puissance de transit par la méthode NRCP

1.5.4 Ecoulement de puissance par la méthode de découplé rapide (FDLF)

Grid data 14 bus | Method Choice FDLF | Run Load Flow | Run Line Flow | Return

Performance Iterative LF

Iteration 53 | Calcul Time (ms) 2

Load Flow Table | Line Flow Table

Bus No	Code No	Voltage Magnitude [pu]	Angle [rad]	Load [MW]	Load [MVAR]	Generation [MW]	Generation [MVAR]
1.0000	1.0000	1.0600	0	0	0	232.3633	-21.4816
2.0000	2.0000	1.0450	-0.0866	21.7000	12.7000	40.0000	34.4884
3.0000	2.0000	1.0100	-0.2212	94.2000	19.0000	0	21.4983
4.0000	0	1.0218	-0.1804	47.8000	-3.9000	0	0
5.0000	0	1.0304	-0.1552	7.6000	1.6000	0	0
6.0000	2.0000	1.0700	-0.2532	11.2000	7.5000	0	41.8298
7.0000	0	1.0831	-0.2322	0	0	0	0
8.0000	2.0000	1.0900	-0.2322	0	0	0	4.2816
9.0000	0	1.0730	-0.2601	29.5000	16.6000	0	0
10.0000	0	1.0712	-0.2635	9.0000	5.8000	0	0
11.0000	0	1.0841	-0.2603	3.5000	1.8000	0	0
12.0000	0	1.0564	-0.2675	6.1000	1.6000	0	0
13.0000	0	1.0530	-0.2687	13.5000	5.8000	0	0
14.0000	0	1.0465	-0.2810	14.9000	5.0000	0	0

Figure 1.13: Ecoulement de Puissance du réseau IEEE 14 nœuds par FDLF

Grid data: 14 bus | Method Choice: FDLF | Run Load Flow | Run Line Flow | Return

Performance Iterative LF

Iteration: 53 | Calcul Time (ms): 2

Load Flow Table | Line Flow Table

From Bus	To Bus	Line Flow (MW)	Line Flow (MVAR)	Line Flow (MVA)	Loss (MW)	Loss (MVAR)
1.0000	2.0000	156.2887	-20.2653	157.5971	4.2647	7.1714
1.0000	5.0000	76.0747	-1.2162	76.0844	2.7841	6.1173
2.0000	1.0000	-152.0240	27.4367	154.4800	4.2647	7.1714
2.0000	3.0000	72.9512	3.5884	73.0394	2.3054	5.0872
2.0000	4.0000	56.0076	-4.0911	56.1569	1.6715	1.0771
2.0000	5.0000	41.3651	-5.1456	41.6839	0.8980	-0.9195
3.0000	2.0000	-70.6458	1.4988	70.6617	2.3054	5.0872
3.0000	4.0000	-23.5542	0.9995	23.5754	0.3695	-2.6280
4.0000	2.0000	-54.3362	5.1682	54.5814	1.6715	1.0771
4.0000	3.0000	23.9236	-3.6275	24.1971	0.3695	-2.6280
4.0000	5.0000	-62.8646	-0.7057	62.8686	0.5053	0.2463
4.0000	7.0000	29.4280	7.7199	30.4237	0.0000	1.6104
4.0000	9.0000	16.0492	-4.6549	16.7106	0.0000	1.4228
5.0000	1.0000	-73.2906	7.3335	73.6566	2.7841	6.1173
5.0000	2.0000	-40.4671	4.2261	40.6872	0.8980	-0.9195
5.0000	4.0000	63.3699	0.9520	63.3771	0.5053	0.2463
5.0000	6.0000	42.7878	-14.1116	45.0548	0	4.8189
6.0000	5.0000	-42.7878	18.9305	46.7884	0	4.8189
6.0000	11.0000	7.8403	7.4500	10.8154	0.0911	0.1908
6.0000	12.0000	7.3721	2.1935	7.6915	0.0635	0.1322
6.0000	13.0000	16.3754	5.7559	17.3575	0.1741	0.3428
7.0000	4.0000	-29.4280	-6.1095	30.0555	0.0000	1.6104
7.0000	8.0000	-0.0000	-4.2544	4.2544	0	0.0272
7.0000	9.0000	29.4280	10.3639	31.1997	0	0.9129
8.0000	7.0000	0.0000	4.2816	4.2816	0	0.0272
9.0000	4.0000	-16.0492	6.0777	17.1614	0.0000	1.4228
9.0000	7.0000	-29.4280	-9.4511	30.9084	0	0.9129
9.0000	10.0000	4.7906	0.4361	4.8104	0.0064	0.0170
9.0000	14.0000	11.1866	5.3373	12.3946	0.1696	0.3608
10.0000	9.0000	-4.7842	-0.4191	4.8025	0.0064	0.0170
10.0000	11.0000	-4.2158	-5.3809	6.8357	0.0334	0.0782
11.0000	6.0000	-7.7492	-7.2591	10.6182	0.0911	0.1908
11.0000	10.0000	4.2492	5.4591	6.9179	0.0334	0.0782
12.0000	6.0000	-7.3085	-2.0613	7.5937	0.0635	0.1322
12.0000	13.0000	1.2086	0.4614	1.2937	0.0033	0.0030
13.0000	6.0000	-16.2013	-5.4131	17.0817	0.1741	0.3428
13.0000	12.0000	-1.2053	-0.4584	1.2895	0.0033	0.0030
13.0000	14.0000	3.9066	0.0714	3.9072	0.0235	0.0479
14.0000	9.0000	-11.0170	-4.9765	12.0888	0.1696	0.3608
14.0000	13.0000	-3.8830	-0.0235	3.8831	0.0235	0.0479

Total Loss (MW)	Total Loss (MVAR)
13.3633	26.1165

Figure 1.14: Calcul de puissance de transit par la méthode FDLF

1.5.5 Comparaison des méthodes en amplitudes de tension et angles

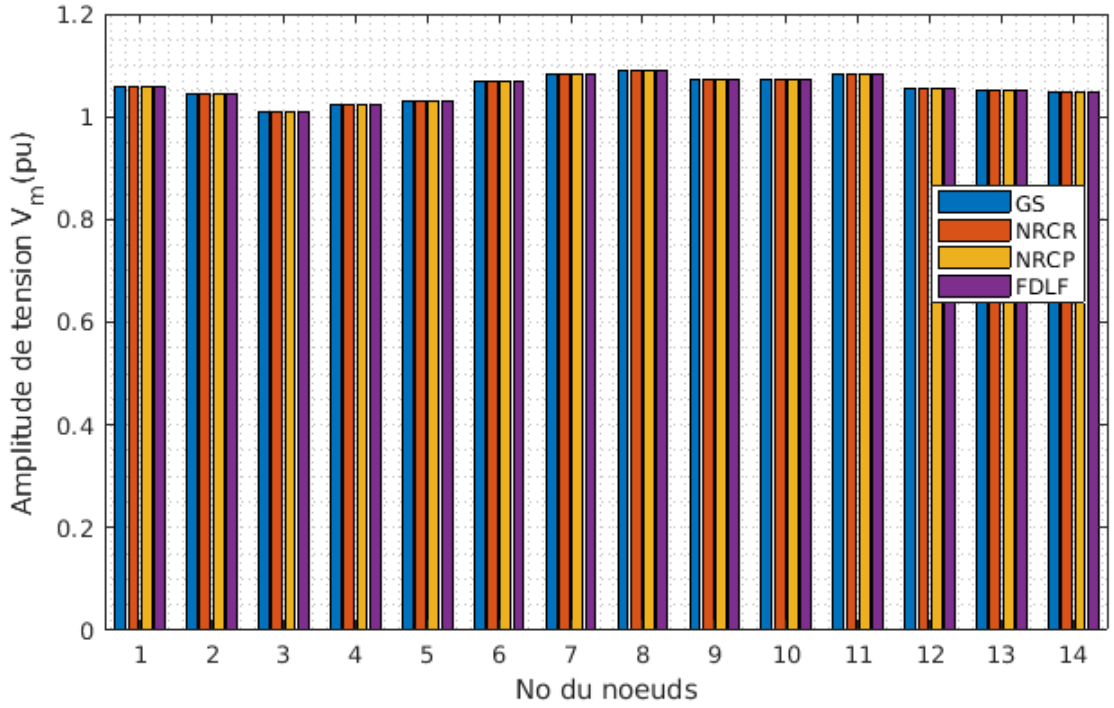


Figure 1.15: Comparaison des quatre méthodes en amplitude de tension

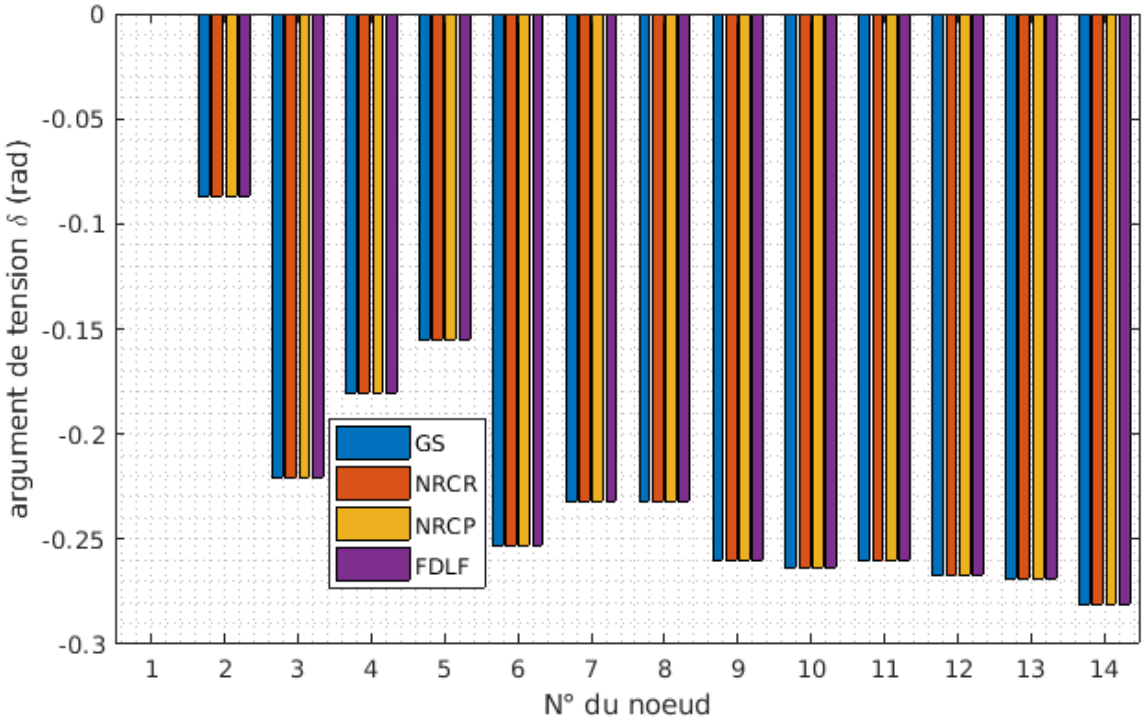


Figure 1.16: Comparaison des quatre méthodes en angle

1.6 Interprétations et discussions

Les résultats ci-dessus montrent que la méthode de Gauss-Seidel prend plus de temps et plus d'itérations que les autres méthodes (174 itérations en 14 ms) pour les mêmes valeurs de tension en module $|V|$, d'angle $|\delta|$, de puissance active et réactive trouvés, car la correction de l'erreur est liée à l'actualisation de la tension et que la convergence de la méthode de Gauss Seidel est linéaire.

De même, les méthodes de Newton Raphson en coordonnées polaires et rectangulaires donnent les mêmes résultats que la méthode GS pour un nombre inférieur d'itérations (environ 5 itérations et en un laps de temps court de 1 ms), car la correction d'erreur est liée à l'actualisation des puissances et leur convergence est quadratique.

La méthode découplée rapide donne les mêmes résultats que ceux obtenus par la méthode NR et GS en un temps petit mais avec un nombre d'itérations plus grand (53 itérations en 2 ms), la matrice jacobienne est sparte, le temps de calcul est donc réduit mais qui a aussi un inconvénient sur le nombre d'itération pour atteindre une bonne précision.

Depuis les figures (1.7), (1.9), (1.11) et (1.13) et les figures comparatives (1.15) et (1.16), nous avons présenté les modules des tensions pour tous les nœuds ainsi que les angles de tension. On voit bien que les tensions sont comprises entre 0.9 pu.et 1.1 pu, ce qui nous donne une information sur la stabilité du réseau 14 nœuds en tension ; le fonctionnement de chaque nœud de ce réseau est normal dans les marges admissibles. Les quatre méthodes convergent vers les mêmes solutions de l'écoulement de puissance, ce qui montre la validité de ces méthodes. Pour les angles de tension, les valeurs ne dépassent pas les 17° en absolu, ce qui assure une stabilité d'angle au niveau des nœuds, spécialement des nœuds de génération PV.

Pour les 4 méthodes utilisées, les valeurs de puissances de transit sont pratiquement les mêmes, les pertes actives dans le réseau sont de 13.36 MW (dans les environs de 4% puissance générée totale) perdues dans les lignes sous forme d'effet Joule, tandis que pour les puissances réactives globales sont autour de 26.11 MVAR, cette perte réactive peut être compensée via l'utilisation des équipement FACTS ou d'autres équipement de compensation.

Chapitre 2 :

Réseau de Neurones Artificiels - Perceptron Multicouche PMC -

2. Chapitre 2 : Réseau de Neurones Artificiels - Perceptron Multicouche PMC -

2.1 Introduction

Durant ces dernières années, l'intelligence artificielle a eu une réputation très reconnue dans le contexte d'identification, analyse, traitement et résolution des problèmes de nos jours. Son application est très vaste en traitement d'information et dans l'étude des systèmes physiques, en implémentant des techniques de résolution intelligentes en fonction de l'état du système. Ces techniques sont inspirées des phénomènes naturels qui se passent autour de nous ou de la nature humaine et destinées à la résolution des problèmes d'ordre complexe et d'optimisation.

Les techniques d'intelligence artificielle sont basées sur l'acquisition, l'apprentissage, le traitement et le stockage des informations. Ces opérations sont traduites par une série d'instructions et de calculs.

Les heuristiques, les métaheuristiques et les réseaux de neurones, sont des techniques intelligentes qui ont faits preuve dans le domaine de Data Science et l'analyse de données, que ça soit pour des problèmes de classification (ex : reconnaissance des images) ou de régression (fitting) pour simuler des modèles mathématiques analytiques ou numériques.

Le réseau de neurones artificiels (RNA) est l'un des outils intelligents utilisés généralement dans le cadre d'apprentissage profond (Deep Learning), d'apprentissage ML (Machine Learning) ou d'autres types d'apprentissage en fonction de la nature du problème à résoudre.

Dans ce chapitre, nous allons définir et analyser un des types de réseaux de neurones artificiels (RNA) ; les Perceptrons Multicouches (PMC).

2.2 Les Réseaux de Neurones Artificiels (RNA)

Les réseaux de neurones présentent un ensemble des cellules connectées entre elles par des liaisons, affectées de poids (synapses) et activer par une fonction dite : fonction d'activation avec un biais donné pour limiter les grandeurs. Ces cellules sont similaires aux neurones biologiques, elles peuvent acquérir, stocker et utiliser des informations et des connaissances empiriques [11]. La possibilité d'apprendre sur la base d'exemples constitue l'une des nombreuses fonctionnalités des réseaux de neurones qui permettent à l'utilisateur de modéliser ses données et établir des règles précises qui vont guider les relations sous-jacentes entre différents attributs de données. L'utilisateur des réseaux de neurones collecte des données représentatives puis il fait appel aux algorithmes d'apprentissage, qui vont apprendre automatiquement la structure des données. La conception et le fonctionnement d'un réseau de neurones consistent à collecter en premier les données d'entrée, élaborer la structure du RNA selon l'application choisie, passer à la phase de l'apprentissage en corrigeant les poids à chaque itération et enfin faire une généralisation en appliquant des tests afin de valider le modèle établi. L'avantage des RNAs est leur pouvoir d'obtenir la solution des problèmes qui n'ont pas un modèle analytique claire.

.

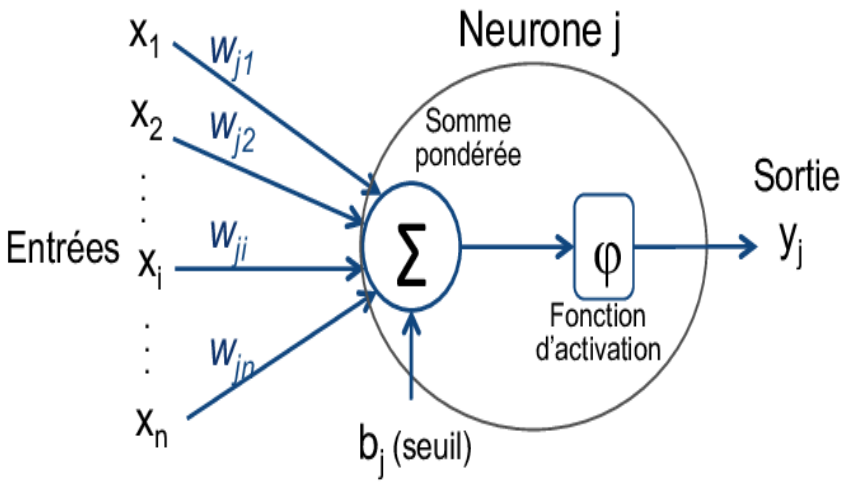


Figure 2.1 : Structure d'un neurone artificiel [12]

L'architecture d'un réseau de neurones artificiels est constituée d'une couche d'entrée (intrants), une couche de sortie (extrants) et une ou plusieurs couches cachées selon le type d'application.

2.2.1 Les intrants et les extrants

Au sein d'un réseau de neurones artificiels, le traitement de l'information suit toujours la même séquence : les informations sont transmises en parallèle sous la forme de signaux aux neurones depuis la couche d'entrée, où elles sont traitées. L'ensemble des intrants et des extrants constituent une base de données pour laquelle on peut entrainer un réseau de neurone.

2.2.2 Les poids

À chaque neurone est attribué un « poids » particulier, et donc une importance différente. Associé à la fonction dite de transfert ou activation, le poids permet de déterminer quelles informations peuvent entrer dans le système. La mise à jour des poids modifie en principe la structure de la fonction d'activation en termes de pente de la fonction (gradient) et image des intrants, et donc nous aurons une amélioration du réseau de neurones. C'est utile, mais si on veut générer une sortie dans une zone saturée de la fonction d'activation, avec un input donné, la modification de la pente de la fonction d'activation n'est plus valable, et à ce moment, le paramètre « biais » qui intervient.

2.2.3 Le biais

Le bais est un paramètre essentiel dans l'apprentissage d'un réseau de neurones artificiels, pour une meilleure performance d'un réseau de neurones. Dans un RNA, un biais est un neurone associé à chaque neurone de la couche cachée ou/et de la couche de sortie, il prend généralement la valeur 1.0. En plus de la mise à jour des poids, il sert à décaler la fonction d'activation pour obtenir des valeurs de sortie de tel sorte de ne pas falsifier l'apprentissage. Par abus de langage et de formulation, la correction des biais se traduit par la correction des connexions liantes ces neurones spéciaux des neurones standards (une mise à jour des pondérations particulière).

2.2.4 La fonction d'activation

Les neurones d'un réseau possèdent des fonctions d'activation qui vont transformer les signaux émis par les neurones de la couche précédente à l'aide d'une fonction mathématique. Les entrées du PMC ne possèdent généralement aucune fonction d'activation. En d'autres termes, ils utilisent la fonction

identité, ce qui signifie que les signaux d'entrée ne sont nullement transformés. Les fonctions d'activation des neurones de sortie sont, dans la plupart des cas, la fonction identité mais ce choix peut varier d'une tâche à l'autre.

Plusieurs fonctions d'activation qui existent en fonction du domaine étudié algébriquement et physiquement, elles sont d'ordre non linéaire qui régit à la résolution des systèmes complexes linéaires ou non linéaire, les plus utilisées souvent sont :

La fonction Rectified Linear Unit (ReLU) : est la fonction d'activation la plus simple en terme de linéarité, la plus utilisée récemment dans des modèles d'apprentissage profond (Deep Learning). C'est la fonction identité pour des entrées positives et nulle sinon. Elle est donnée par cette formulation :

$$f(X) = ReLU(X) = max(0, X) = \begin{cases} X & si\ X > 0 \\ 0 & Sinon \end{cases} \quad (2.1)$$

Cette fonction permet d'effectuer un filtre sur nos données. Elle laisse passer les valeurs positives **(**x > 0) dans les couches suivantes du réseau de neurones**.** Selon l'ordre de grandeurs de sorties nous pouvons activer la couche de sortie et ça permet d'obtenir des meilleurs résultats et aussi elle est utilisée dans les couches intermédiaires.

Les avantages de l'emploi de cette fonction sont comme suit :

- Les valeurs négatives seront converties vers 0 et donc on n'aura pas de présence des valeurs négatives.
- La valeur maximale de la sortie va jusqu'à l'infini, la précision de la prédiction de sortie est maximale.
- Apprentissage rapide par rapport aux fonctions d'activation précédentes.

Un problème peut se poser sur la dérivabilité de la fonction ReLU au voisinage de 0, généralement on met la dérivée de la fonction ReLU au point 0 à 0.

Il existe également d'autres fonctions de la même famille que la ReLU, dédiées pour certaines applications qui prohibent d'avoir un gradient nul pour des valeurs négatives.

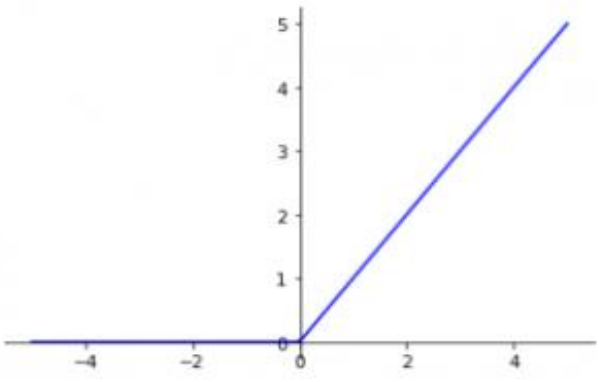


Figure 2.2 : fonction relu –Rectified Linear Unit

La fonction identité (Pureline activation) : la fonction identité est une fonction linéaire dont les valeurs de sortie peuvent varier de $-\infty$ jusqu'à $+\infty$. L'activation du neurone est transmise directement en sortie vers la couche suivante ou la valeur d'extrants. Dans le jargon de domaine des PMCs, l'application de cette fonction sur un neurone n'est pas considérée comme une activation.

$$\boldsymbol{f(X) = X} \qquad (\mathbf{2.2})$$

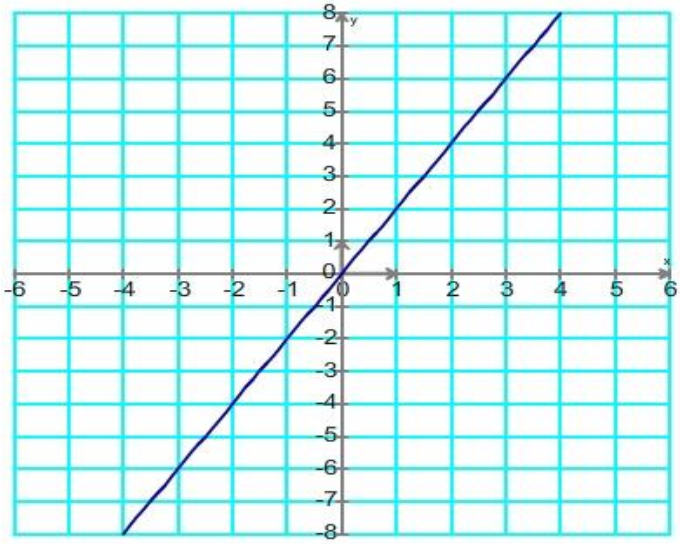


Figure 2.3: Fonction linéaire

Dans ce cas, les problèmes de gradient et de dérivabilité dispersaient et l'algorithme d'apprentissage devient rapide.

La fonction logistique (sigmoïde) : pour une grandeur réelle, la sortie est dans la plage de 0 à 1, utilisée généralement dans des problèmes de classification probabilistique ou de régression dans l'ordre de grandeurs défini précédemment, cette fonction est définie par :

$$\boldsymbol{f(X) = \frac{1}{1 + \exp(-X)}} \qquad (\mathbf{2.3})$$

L'inconvénient de ce type de fonction de transfert est qu'elle n'est pas symétrique, non centrée au voisinage de 0. L'apprentissage avec cette fonction est lent pour des valeurs très grandes en absolue et le gradient qui tend pratiquement vers 0, le neurone rentre dans ce cas, dans un état de réanimation.

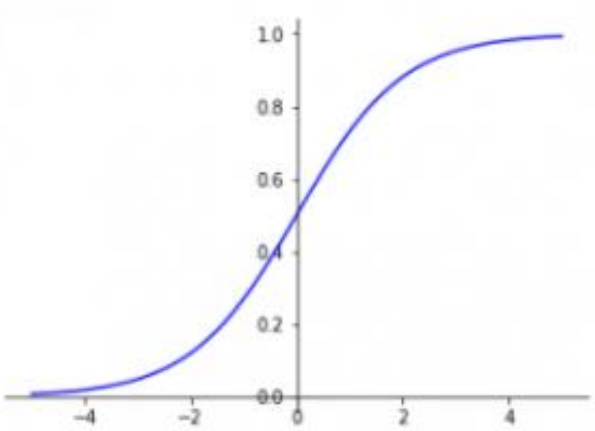


Figure 2.4: Fonction sigmoïde

La fonction tangente hyperbolique : la sortie de la fonction varie entre -1 et 1 pour une valeur d'entrée donnée. Dans la littérature, cette fonction est plus performante comparant à la fonction logistique en raison de sa large variation de sortie. Son gradient est très important pour des valeurs au voisinage de 0 ou pour des valeurs faibles.

$$\boldsymbol{f(X) = \tanh(X) = \frac{\exp(X) - \exp(-X)}{\exp(X) + \exp(-X)}} \qquad (2.4)$$

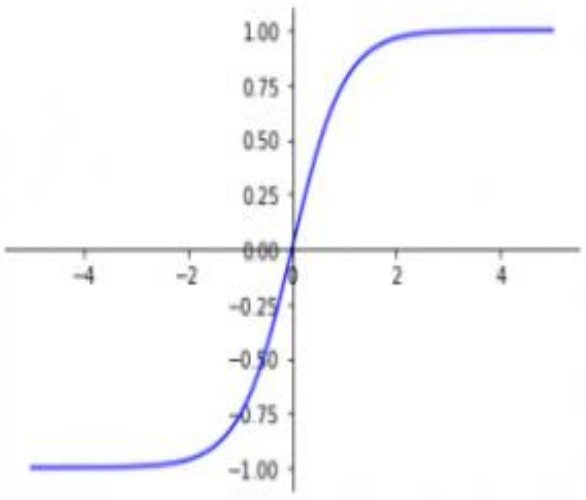


Figure 2.5: Fonction tangente hyperbolique

Le problème se pose également dans le calcul de gradient comme mentionné auparavant sur la fonction logistique. Le gradient de la tangente hyperbolique est pratiquement nul pour des valeurs importantes, ce qui influence sur l'apprentissage du réseau de neurones, la vitesse de la phase d'apprentissage devient lente pour atteindre l'optimum du modèle. La précision dans ce cas est très bonne comparant à la fonction logistique.

2.3 Le Perceptron Multicouche (PMC)

Un perceptron multicouche (PMC) est une classe de réseau neuronal artificiel (RNA) à propagation directe. Le terme PMC est utilisé de manière ambiguë, parfois pour désigner n'importe quel RNA à propagation directe. C'est un type de réseau de neurones supervisé, considéré comme étant le plus simple et connu. Chaque neurone est connecté à tous les neurones de la couche précédente et la suivante. Selon Minskey et Papert en 1969, il était prouvé qu'un perceptron ne peut généraliser sur des exemples appris localement [11], c'est pourquoi on passe au perceptron multicouche.

Le perceptron multicouche est un type de perceptron normalisé avec une à deux couche cachée. En pratique, si on augmente le nombre de couches cachées, la quantité de calcul monte d'une façon exponentielle, d'où l'utilité des réseaux de perceptron multicouche [11]. Il sert à résoudre des problèmes non linéaires et complexes. Le PMC est utilisé dans la recherche, vu son pouvoir de résoudre d'une manière stochastique des problèmes de régression, classification et pour des applications diverses dans le domaine de Data Science.

2.3.1 Architecture d'un perceptron multicouche

Un PMC présente une couche d'entrée non activée (fonction d'activation identité), les couches cachées et la couche de sortie sont activées par des fonctions d'activation selon le type d'application.

La couche cachée contient des nœuds (unités) de réseau non observables. Chaque unité cachée est une fonction de la somme pondérée des entrées. Cette fonction est la fonction d'activation et les valeurs des pondérations sont déterminées par l'algorithme d'estimation. Si le réseau contient une seconde couche cachée, chaque unité cachée de la seconde couche est une fonction de la somme pondérée des unités de la première couche cachée. Pour une bonne architecture et meilleure précision, il est recommandé d'homogénéiser les couches cachées, autrement dit, il faut bien utiliser la même fonction d'activation pour toutes les couches cachées.

Dans ce qui suit, nous allons élaborer les PMCs à une seule couche cachée, visualiser ses performances et ses caractéristiques, comment faire apprendre un PMC par la méthode de descente de gradient par rétro – propagation et étudier les « hyper – paramètres » pour avoir une architecture performante.

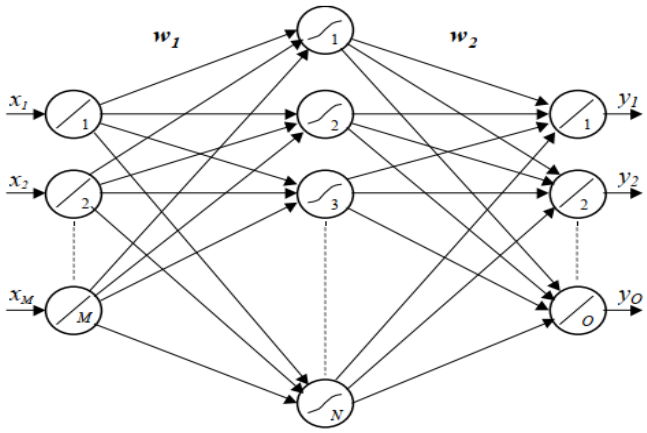


Figure 2.6: Architecture d'un perceptron multicouche à une couche cachée [1]

2.3.2 Base de données

Afin de faire fonctionner un PMC, on doit l'alimenter avec des données d'entrée, extraites d'une base de données existante. L'apprentissage est dit supervisé, en raison de l'utilisation des valeurs de sortie désirées depuis la même base de données, comme étant des références afin de comparer la sortie référence avec la sortie prédite. Cela nous permet de mettre à jour les pondérations et les biais. Selon la problématique traitée et la méthode intelligente utilisée, une partie de la base de données sera utilisée pour assurer la phase d'apprentissage du perceptron multicouche, et une partie pour valider et tester les performances du PMC. Dans certaines applications, une base de données est subdivisée en deux (à la limite trois paquets) :

Deux paquets : données d'apprentissage et de test.

Trois paquets : données d'apprentissage, de validation et de test.

Un exemple (Echantillon)

Chaque élément d'une base de données présente un échantillon qu'on peut l'utiliser uniquement pour l'apprentissage ou pour le test.

Un exemple contient un vecteur d'entrée X d'une dimension donnée et un vecteur de sortie Y d'une autre dimension.

Les vecteurs d'entrée X de dimension $n \times 1$ et de sortie Y de dimension $n' \times 1$ sont définis comme suit :

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_{n'} \end{bmatrix} \tag{2.5}$$

2.3.2.1 Les données d'apprentissage

C'est une association de plusieurs exemples dans un paquet de données, ce dernier présente généralement 70% à 80% de la base de données globale afin d'effectuer un apprentissage par rétro – propagation en utilisant l'algorithme de descente de gradient et faire apprendre le PMC.

Dans un paquet de nombre d'exemples m, un exemple est représenté par X^i où i est la position de l'exemple dans le paquet d'apprentissage :

$$X_{train} = [X^1 \quad X^2 \quad \dots \quad \dots \quad X^m] \tag{2.6}$$

$$Y_{train} = [Y^1 \quad Y^2 \quad \dots \quad \dots \quad Y^m] \tag{2.7}$$

La taille de X_{train} et Y_{train} est de $n \times m$, respectivement $n' \times m$.

2.3.2.2 Les données de test

Les données restantes sont dédiées pour tester et valider la performance du perceptron multicouche.

$$X_{test} = [X^{m+1} \quad X^{m+2} \quad \dots \quad \dots \quad X^{l}] \tag{2.8}$$

$$Y_{test} = [Y^{m+1} \quad Y^{m+2} \quad \dots \quad \dots \quad Y^{l}] \tag{2.9}$$

La taille de X_{test} et Y_{test} est de $n \times (l - m - 1)$ respectivement $n' \times (l - m - 1)$.

Il est recommandé de travailler avec des entrées souvent normalisées pour avoir un bon fonctionnement du PMC. La normalisation des données assure en quelque sorte leur bonne distribution si les valeurs sont très faibles ou très grandes contenant le vecteur d'entrée.

2.3.3 Initialisation des poids et des biais

L'initialisation d'un PMC avec des bonnes valeurs de pondérations peut faire la différence entre le réseau de neurones qui converge dans un délai raisonnable et la fonction de perte du réseau PMC qui ne va nul part même après plusieurs itérations.

L'algorithme de descente de gradient nécessite un point de départ dans l'espace des valeurs de poids possibles à partir duquel commencer le processus d'optimisation. L'initialisation des poids est une procédure permettant de fixer les poids d'un réseau neuronal à de petites valeurs aléatoires qui définissent le point de départ de l'optimisation (apprentissage ou formation) du modèle de réseau neuronal.

Si les poids sont trop faibles, l'ordre de grandeur des sorties calculées depuis la couche cachée diminue fortement. L'entrée finit par tomber à une valeur vraiment faible et ne peut plus être utile. Considérons la fonction tangente hyperbolique, si nous l'utilisons comme fonction d'activation, l'approximation dans cette plage de valeurs devient linéaire lorsque nous nous rapprochons de zéro. Cela signifie essentiellement qu'on ne profite pas de la non-linéarité de la fonction.

Si les poids sont trop importants, la variance des données d'entrée a tendance à augmenter rapidement dans chaque couche. Finalement, elle devient si grande qu'elle devient inutile. La fonction tangente hyperbolique a tendance à devenir plate pour les grandes valeurs, ce qui signifie que les activations seront saturées et que les gradients se rapprochent de zéro.

L'initialisation du réseau avec les bons poids est une étape très importante si nous voulons que notre réseau neuronal fonctionne correctement. Nous devons nous assurer que les poids sont dans une fourchette raisonnable avant de construire le PMC, en d'autre terme, le réseau de neurones doit être normalisé en valeur pour le faire fonctionner convenablement.

Historiquement, l'initialisation des poids suit des heuristiques simples qui prennent des petites valeurs aléatoires dans l'intervalle [-0.3, 0.3], [0, 1] ou [-1, 1] selon l'architecture désirée, ces heuristiques continuent à bien fonctionner en général [13].

L'approche standard actuelle pour l'initialisation des poids des couches et des nœuds de réseaux neuronaux qui utilisent la fonction d'activation Sigmoïde ou Tangente hyperbolique est appelée initialisation de **G. Xavier** (2010).

2.3.3.1 L'initialisation de Xavier

La méthode d'initialisation Xavier est calculée comme un ensemble de nombres aléatoires avec une distribution de probabilité uniforme (U) dans la plage $-\left(\frac{1}{\sqrt{\mathrm{N_{in}}}}\right)$ et $\left(\frac{1}{\sqrt{\mathrm{N_{in}}}}\right)$ (Equation 2.10), où N_{in} est le nombre d'entrées, ou bien avec une distribution normale de médiane égale à 0 et une variance de $\sqrt{\frac{2}{N_{in}}}$ (Equation 2.11) [1].

$$\mathrm{W} \sim \mathrm{U}\left[-\left(\frac{1}{\sqrt{N_{in}}}\right), \left(\frac{1}{\sqrt{N_{in}}}\right)\right] \tag{2.10}$$

$$\mathrm{W} \sim \mathcal{N}\left[0, \sqrt{\frac{2}{\mathrm{N_{in}}}}\right] \tag{2.11}$$

2.3.3.2 Initialisation normalisée de Xavier

L'initialisation de Xavier normalisée est calculée comme un nombre aléatoire avec une distribution de probabilité uniforme (U) entre la plage $-\sqrt{\frac{6}{N_{in}+N_{out}}}$ et $\sqrt{\frac{6}{N_{in}+N_{out}}}$ (Equation 2.12), où N_{in} est le nombre d'entrées du noeud (par exemple, le nombre de nœuds dans la couche précédente et N_{out} est le nombre de sorties de la couche (par exemple, le nombre de nœuds dans la couche actuelle), ou suivant une loi normale de médiane 0 et de variance $\sqrt{\frac{2}{\mathrm{N_{in}}+N_{out}}}$ (Equation 2.13).

$$\mathrm{W} \sim \mathrm{U}\left[-\sqrt{\frac{6}{N_{in}+N_{out}}}, \sqrt{\frac{6}{N_{in}+N_{out}}}\right] \tag{2.12}$$

$$\mathrm{W} \sim \mathcal{N}\left[0, \sqrt{\frac{2}{\mathrm{N_{in}}+N_{out}}}\right] \tag{2.13}$$

Dans la plupart des méthodes d'apprentissage, les biais sont généralement mis à 0 [13].

Remarque :

Nous évitons d'initialiser les pondérations à une seule valeur constante, en raison d'impossibilité de briser la symétrie de la matrice de poids, du coup les gradients seront les mêmes durant toute la phase d'apprentissage, le modèle établi du PMC est équivalent à un seul neurone.

2.3.3.3 Initialisation de Kaiming He

L'initialisation de Kaiming, ou initialisation de He (2015), est une méthode d'initialisation pour les réseaux neuronaux qui tient compte de la non-linéarité et parfois la linéarité des fonctions d'activation, telles que les activations ReLU et la fonction identité.

Une méthode d'initialisation appropriée doit éviter de réduire ou d'amplifier de manière exponentielle les amplitudes des signaux d'entrée**.**

Kaiming He suggère que l'initialisation de Xavier et d'autres schémas ne sont pas appropriés pour ReLU et ses extensions. Il a proposé une petite modification de l'initialisation de Xavier pour la rendre appropriée à l'utilisation de ReLU, à voir la fonction d'identité maintenant communément appelée « Initialisation de Kaiming » (spécifiquement des nombres aléatoires selon la loi uniforme dans la plage de $\pm\sqrt{\frac{2}{N_h}}$ où N_h est le nombre de nœuds dans la couche cachée ou suivant une loi normale de médiane 0 et une variance de $\sqrt{\frac{2}{N_h}}$.

2.3.3.4 Initialisation normalisée de Kaiming He

L'astuce s'est améliorée dans le cadre de la recherche et d'autre mise à niveau a été effectuée pour maitriser bien la bonne conception du PMC, la normalisation de l'initialisation de He se traduit par une des deux lois de probabilité :

$$\mathrm{W} \sim \mathrm{U}\left[-\left(\sqrt{\frac{6}{N_h}}\right),\left(\sqrt{\frac{6}{N_h}}\right)\right] \tag{2.14}$$

$$\mathrm{W} \sim \mathcal{N}\left[0,\sqrt{\frac{6}{\mathrm{N_h}}}\right] \tag{2.15}$$

2.3.4 Propagation

Une fois que le réseau de neurones (PMC) est bien construit, le calcul commence par une phase dite : phase de propagation, ceci détermine en final la sortie prédite depuis la couche d'entrée vers la couche de sortie. Si l'apprentissage se fait pour un exemple i, la propagation se traduit en forme d'équations.

La propagation depuis la couche d'entrée vers la couche cachée :

$$u^{in^i} = w_1\, X^i + b_1 \tag{2.16}$$

Après activation avec une fonction f :

$$u^{out^i} = f\left(u^{in^i}\right) \tag{2.17}$$

Au niveau de la couche de sortie :

$$Y^{in^i} = w_2\, u^{out_1^i} + b_2 \tag{2.18}$$

Une fois la sortie sera activée par une fonction g, on aura :

$$Y^{out^i} = g\left(Y_2^i\right) \tag{2.19}$$

2.3.5 Erreur quadratique moyenne, fonction coût et fonction perte

L'étude des performances de bon apprentissage du PMC et sa validité se résume sur un calcul d'erreur entre la sortie obtenue après propagation et la sortie désirée depuis une base de données. Si l'erreur est minimale, cela veut dire que le modèle du PMC s'adapte bien et donne de bons résultats. Plusieurs formes d'erreur sont utilisées pour juger la bonne adaptation du PMC au problème traité, une des formes la plus utilisée est bien l'erreur quadratique moyenne (Mean Square Error MSE). Elle est basée sur la théorie de régression par la méthode de moindre carré. L'objectif est de trouver une bonne droite de régression de tel sorte qu'il faut avoir une somme de carré de l'erreur entre les sorties désirées et les sorties calculée qui converge vers 0.

L'expression de l'erreur quadratique moyenne est donnée par :

$$E = \frac{1}{2}(Y_{calc}^{out} - Y_{dés}^{out})^2 \tag{2.20}$$

Dans le domaine d'apprentissage Machine Learning ou Deep Learning, cette erreur est appelée **la fonction de perte (Loss function)**.

L'apprentissage par la descente du gradient s'effectue par le calcul de gradient de l'erreur par rapport aux poids et aux biais, mais l'erreur prise dans ce cas est une somme des erreurs quadratiques moyennes (moyenne des pertes de l'ensemble d'apprentissage), cette somme d'erreurs est appelée **la fonction coût (Cost function)**, elle est définie par la formulation suivante **:**

$$J(w, b) = \sum_{i=1}^{m} \frac{1}{2m}(Y_{calc}^{out} - Y_{dés}^{out})^2 \tag{2.21}$$

m : Le nombre d'exemple d'apprentissage.

Le concept de validation du modèle de PMC est similaire à l'apprentissage avec calcul de la fonction coût pour les données de test.

La fonction coût nous donne une information si le modèle a sous – appris (Underfitting) ou sur – appris (Overfitting) en fonction de nombre d'itérations (Epochs). Un modèle sous-appris est un modèle dont les performances sont bonnes sur l'ensemble de données d'apprentissage et mauvaises sur l'ensemble de données de test. Cela peut être diagnostiqué à partir d'un graphique où la perte d'apprentissage est inférieure à la perte de validation, et où la perte de validation a une tendance qui suggère que des améliorations supplémentaires sont possibles en jouant sur les hyper – paramètres comme le nombre d'itération.

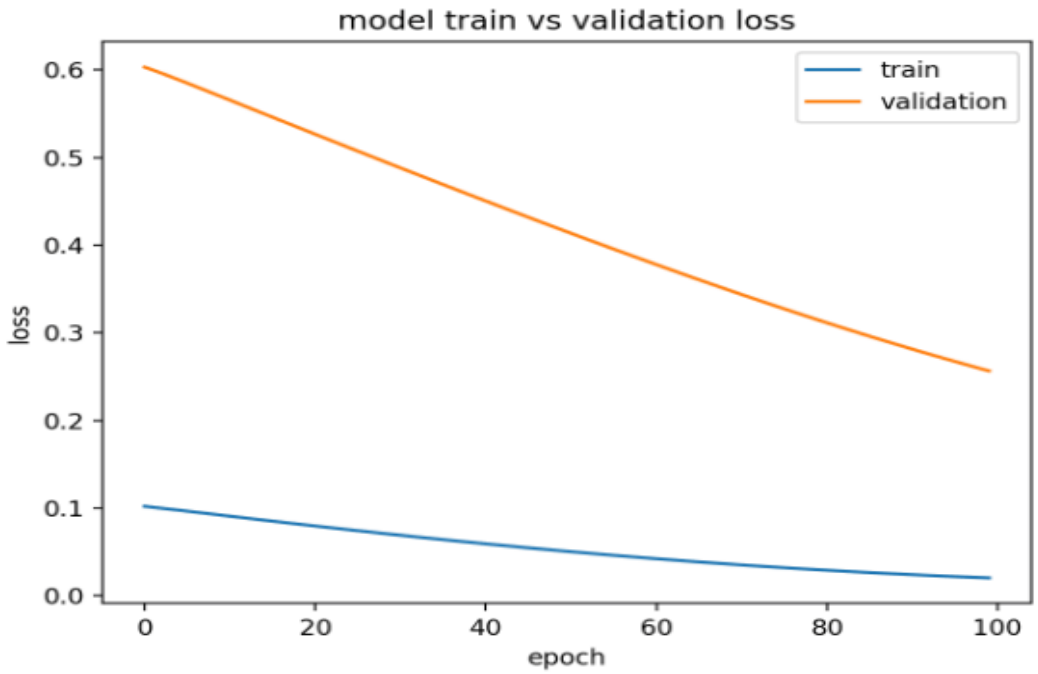


Figure 2.7 : Exemple de la fonction coût en fonction de nombre d'itération pour un modèle sous-appris

Un bon apprentissage est un cas où la performance du modèle est bonne à la fois sur les ensembles de formation et de validation. Ceci peut être diagnostiqué à partir d'un graphique où les pertes de l'ensemble d'entraînement et de l'ensemble de validation diminuent et se stabilisent autour du même point.

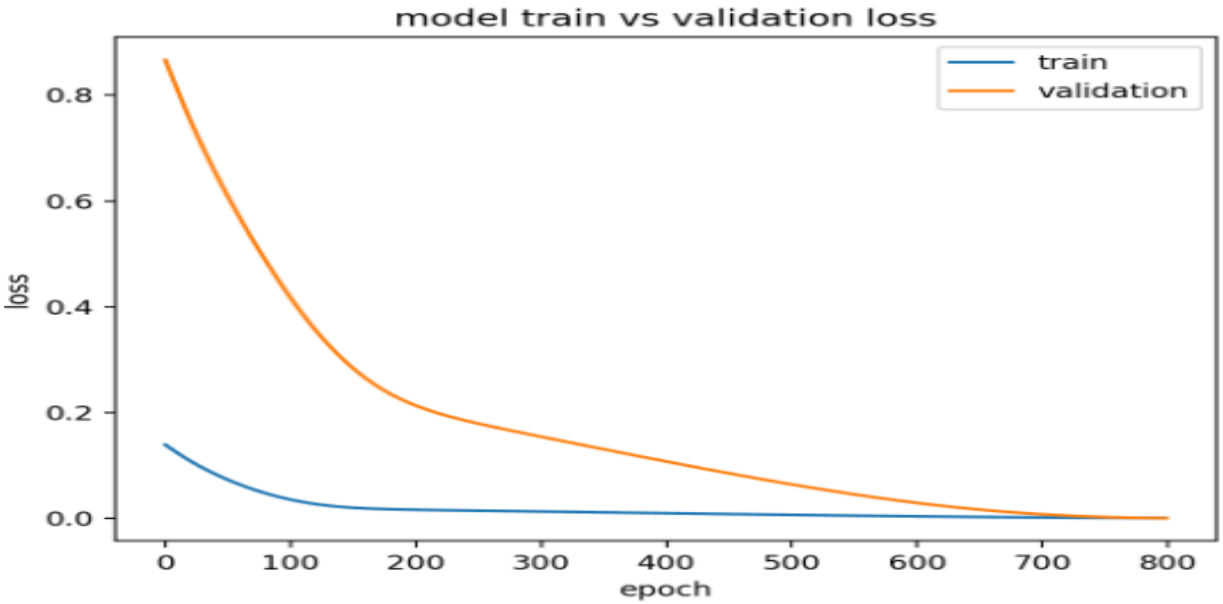


Figure 2.8: Exemple de la fonction coût en fonction de nombre d'itération pour un modèle bien-appris

Un modèle sur - appris est un modèle dont les performances sur l'ensemble de formation sont bonnes et continuent de s'améliorer, tandis que les performances sur l'ensemble de test s'améliorent jusqu'à

un certain point, puis commencent à se dégrader. Ce phénomène peut être diagnostiqué à partir d'un graphique où la fonction coût diminue et la perte de validation diminue, atteint un point d'inflexion et recommence à augmenter.

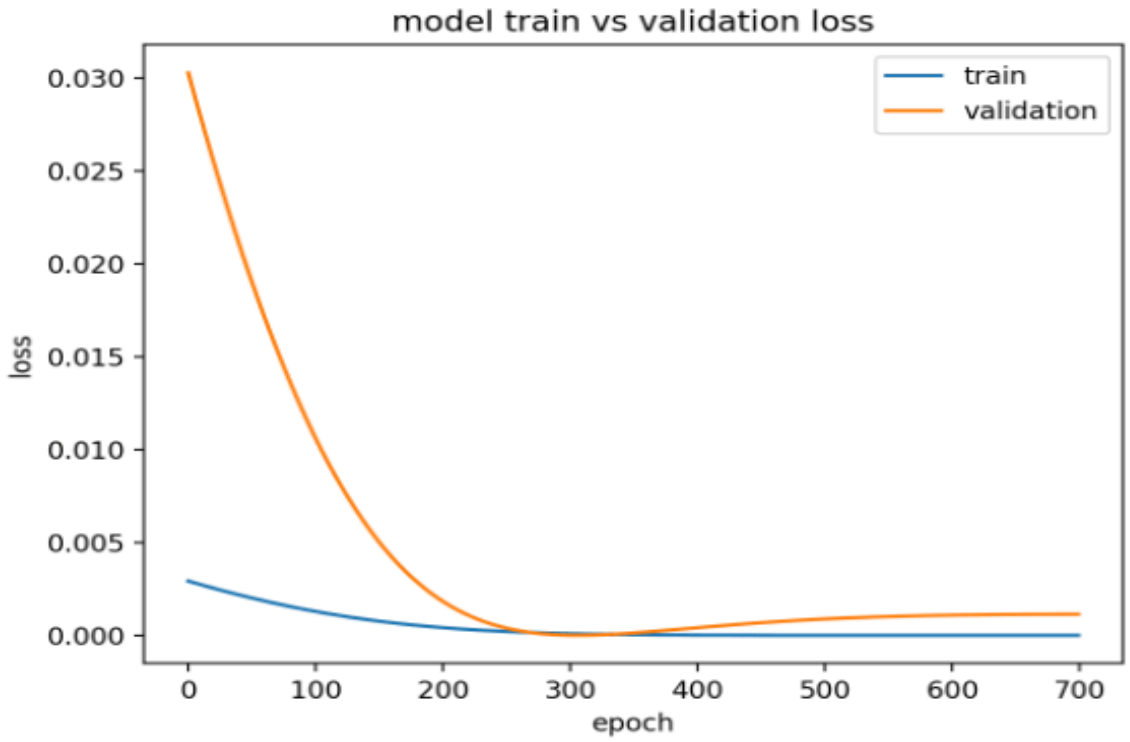


Figure 2.9: Exemple la fonction coût en fonction de nombre d'itération pour un modèle sur-appris

2.3.6 Rétro – propagation par descente du gradient

Dans la littérature, il existe des algorithmes de correction des poids et des biais pour les réseaux de neurones, mais qui sont limités en termes d'apprentissage. Puisque on ne connait pas les sorties désirées de la couche cachée mais seulement la dernière couche, on fait une sorte de propagation de responsabilité des erreurs de la dernière couche au première, et donc dans le sens contraire de l'exécution du réseau de neurones [11], d'où l'appellation de rétro – propagation.

La fonction d'activation utilisée est généralement la tangente hyperbolique car l'importance des intrants se traduit par le gradient de la fonction d'activation (dérivée). On préfère utiliser une fonction de type **tanh** que de type **signe** en raison que la courbe produite par le système se modifie jusqu'à ce que les réponses soient correctes et cessera après [11], contrairement à la fonction **tanh** qui n'a pas cette notion de réponse correcte ou incorrecte, les valeurs de la réponse de la couche cachée sont bornées entre -1 et 1. Cet algorithme travaille également avec une règle d'apprentissage dite la loi de delta, consiste à réduire la fonction de coût par rapport au poids synaptiques en cherchant la dérivée d'où l'utilisation de calcul du gradient par la méthode de descente du gradient.

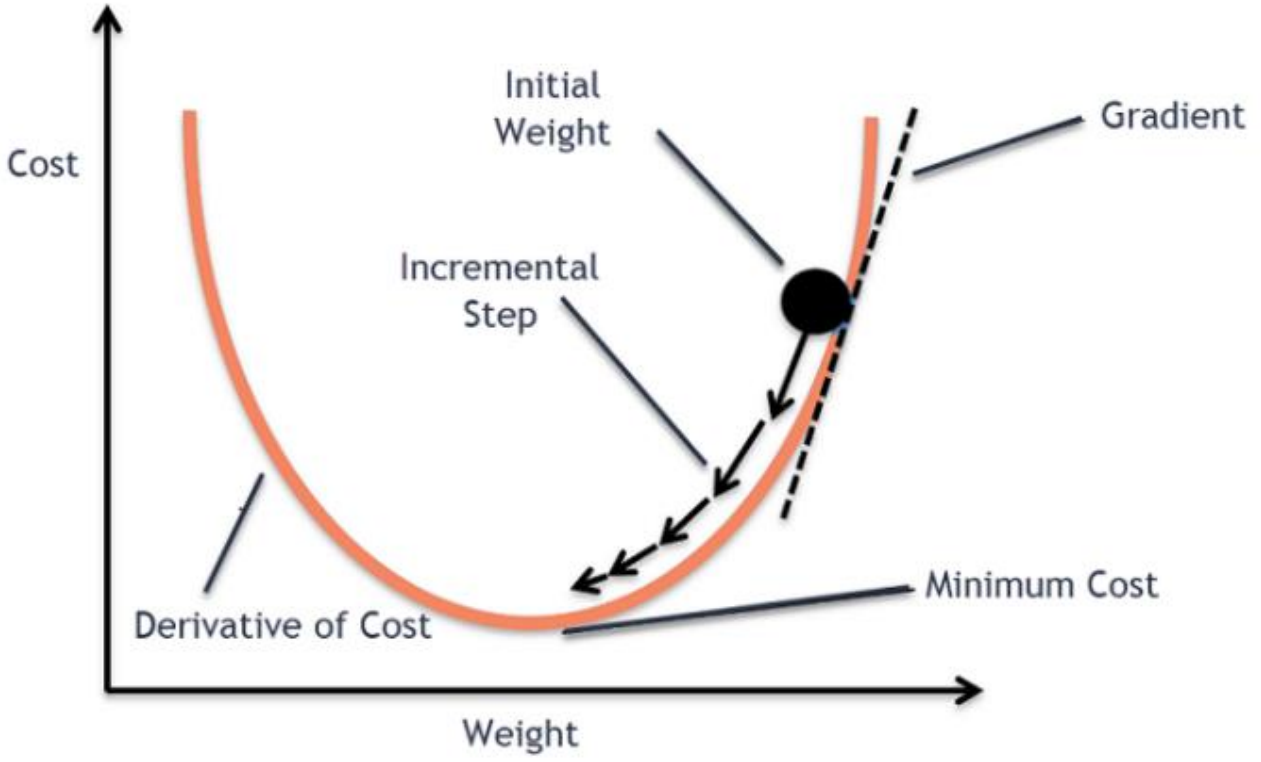


Figure 2.10 : Minimisation de la fonction coût par rapport aux pondérations

La figure (2.10) représente la manière de minimiser la fonction coût J par rapport aux pondérations W, et on voit bien que les pondérations se corrigent depuis la valeur initiale en calculant le gradient de la fonction J par rapport aux pondérations avec un avancement incrémental dit : pas d'apprentissage.

L'objectif de cet algorithme est de calculer dans chaque étape du processus les nouveaux pondérations et biais liés entre les neurones, cette étape est dite phase d'entraînement des poids et donc le réseau en lui-même.

La rétro – propagation est la partie où nous utilisons les sorties et les réseaux de couches avec les fonctions d'activation pour obtenir les gradients que nous utilisons ensuite pour mettre à jour les poids afin de réduire le coût. Examinons rapidement les équations et les étapes du processus de la descente du gradient.

Nous prenons un paquet d'exemple d'apprentissage de taille m depuis (2.6) et (2.7) et en passant par la phase de propagation, calculant l'erreur entre les réponses prédites du PMC et les réponses désirées :

$$e = (Y_{calc}^{out} - Y_{dés}^{out}) \tag{2.22}$$

Une fois la fonction coût est calculée à partir de l'erreur (2.21), on effectue une mise à jour des poids et des biais selon la loi de delta généralisé qui se permet à calculer la variation de la fonction de coût par rapport aux poids et biais. L'algorithme de rétro – propagation se donne donc par cette succession d'instructions.

Cas de neurones de la couche de sortie

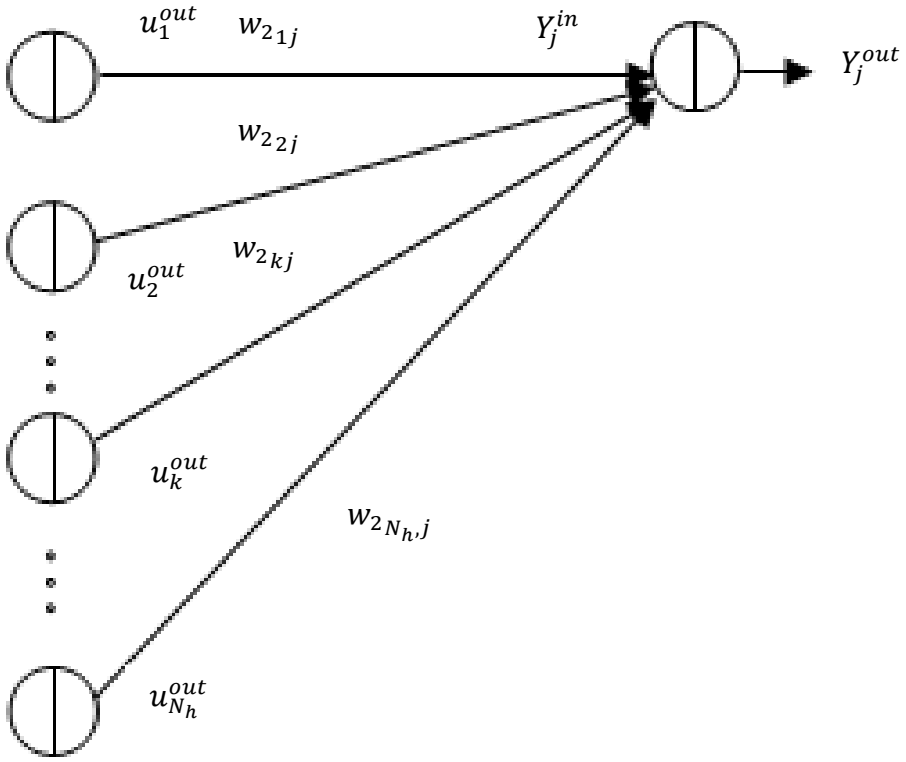


Figure 2.11: Cas de neurone de la couche de sortie [11]

En commençant à partir de la couche de sortie vers la couche cachée (couche précédente), la correction des pondérations va obéir à la règle du gradient :

$$w_{2_{kj}} = w_{2_{kj}} + \Delta w_{2_{kj}} = w_{2_{kj}} - \alpha \frac{\partial E}{\partial w_{2_{kj}}} \tag{2.23}$$

Où α est un paramètre de déplacement dit : pas d'apprentissage.

Il faudra donc déterminer ($\partial E / \partial w_{2_{kj}}$), or E est une fonction ou expression composée, c'est-à-dire que E dépend de Y_j^{out} qui dépend de Y_j^{in} qui dépend à son tour de $w_{2_{kj}}$, ce qui permet d'écrire :

$$\frac{\partial E}{\partial w_{2_{kj}}} = \frac{\partial E}{\partial Y_j^{out}} \frac{\partial Y_j^{out}}{\partial Y_j^{in}} \frac{\partial Y_j^{in}}{\partial w_{2_{kj}}} \tag{2.24}$$

Avec

$$\frac{\partial E}{\partial Y_j^{out}} = \left(Y_j^{out} - Y_{des,j}^{out}\right) = \left(g'\left(Y_j^{in}\right) - Y_{des,j}^{out}\right) \tag{2.25}$$

$$\frac{\partial Y_j^{out}}{\partial Y_j^{in}} = \frac{\partial f(Y_j^{in})}{\partial Y_j^{in}} = g'\left(Y_j^{in}\right) \tag{2.26}$$

$$\frac{\partial Y_j^{in}}{\partial w_{2_{kj}}} = u_k^{out} \tag{2.27}$$

Si par exemple la fonction d'activation de la couche de sortie est une fonction linéaire :

$$Y_j^{out} = g\left(Y_j^{in}\right) = \mathrm{Y}_j^{in} \tag{2.28}$$

De (2.26) :

$$\frac{\partial Y_j^{out}}{\partial Y_j^{in}} = g'\left(Y_j^{in}\right) = 1 \tag{2.29}$$

Depuis (2.25), (2.27) et (2.29) on obtient :

$$\frac{\partial E}{\partial w_{2_{kj}}} = \left(g\left(Y_j^{in}\right) - Y_{des,j}^{out}\right) u_k^{out} \tag{2.30}$$

La définition du gradient local δ_j (la loi delta) traduit l'expression :

$$\frac{\partial E}{\partial w_{2_{kj}}} = \delta_j u_k^{out} \tag{2.31}$$

En posant :

$$\delta_j = e_j . g'\left(Y_j^{in}\right) = \left(g\left(Y_j^{in}\right) - Y_{des,j}^{out}\right) . g'\left(Y_j^{in}\right) \tag{2.32}$$

Ce qui donne

$$w_{2_{kj}} = w_{2_{kj}} - \alpha . \delta_j \, u_k^{out} \tag{2.33}$$

Cas de neurones de la couche cachée

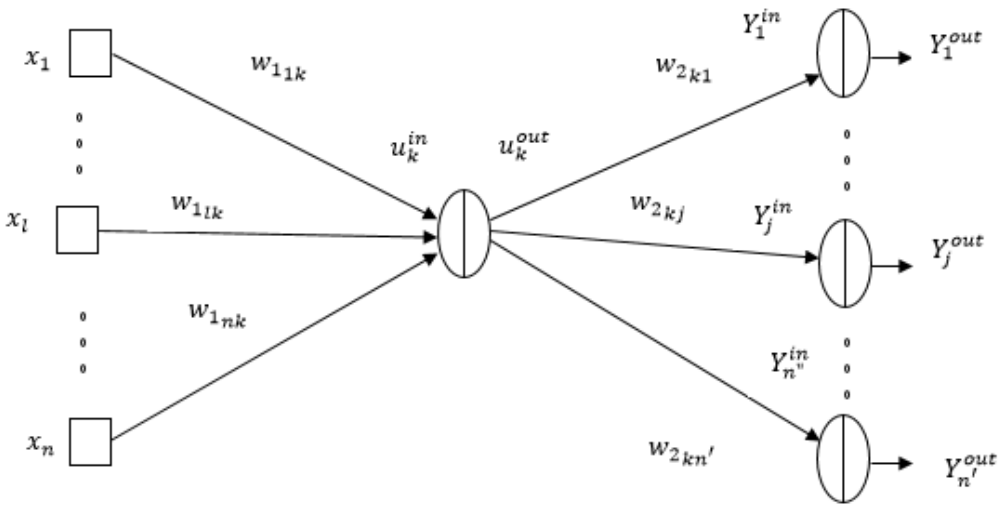


Figure 2.12: Cas de neurones de la couche cachée [11]

Nous devons ensuite corriger les pondérations entre la couche cachée et la couche d'entrée par le biais de la formulation ci – dessous :

$$w_{1_{lk}} = w_{1_{lk}} + \Delta w_{1_{lk}} = w_{1_{lk}} - \alpha \frac{\partial E}{\partial w_{1_{lk}}} \tag{2.34}$$

Il faudra donc déterminer $(\partial E/\partial w_{1_{lk}})$ en utilisant l'erreur quadratique (2.20). Dans ce cas, il faut dériver *E* par rapport aux poids de la couche cachée précédente $w_{1_{lk}}$ à travers une décomposition reliant u_k^{out} à u_k^{in} qui dépend à son tour de$w_{1_{lk}}$, ce qui permet d'écrire :

$$\frac{\partial E}{\partial w_{1_{lk}}} = \frac{\partial E}{\partial u_k^{out}} \frac{\partial u_k^{out}}{\partial u_k^{in}} \frac{\partial u_k^{in}}{\partial w_{1_{lk}}} \tag{2.35}$$

En premier, dérivons l'erreur E par rapport à u_k^{out} :

Pour un neurone de cette couche cachée, le calcul est le même que le cas d'un neurone de sortie, jusqu'à :

$$\frac{\partial E}{\partial u_k^{out}} = \frac{1}{2} \sum_{j=1}^{s} \frac{\partial e_j^2}{\partial u_k^{out}} = \sum_j e_j \frac{\partial e_j}{\partial u_k^{out}} \tag{2.36}$$

La dépendance entre e_j et u_k^{out} est à deux niveaux, ce qui nous amène à faire la décomposition suivante :

$$\frac{\partial e_j}{\partial u_k^{out}} = \frac{\partial e_j}{\partial Y_j^{in}} \frac{\partial Y_j^{in}}{\partial u_k^{out}} \tag{2.37}$$

Où :

$$\frac{\partial e_j}{\partial Y_j^{in}} = \frac{\partial \left(g\left(Y_j^{in}\right) - Y_{des,j}^{out}\right)}{\partial Y_j^{in}} = g'\left(Y_j^{in}\right) \tag{2.38}$$

La dérivée de la fonction g est 1 si la fonction utilisée est une fonction linéaire.

Et on a encore :

$$\frac{\partial Y_j^{in}}{\partial u_k^{out}} = \frac{\partial \left(\sum w_{2_{kj}} u_k^{out}\right)}{\partial u_k^{out}} = w_{2_{kj}} \tag{2.39}$$

D'où :

$$\frac{\partial E}{\partial u_k^{out}} = \sum_j e_j g'\left(Y_j^{in}\right) w_{2_{kj}} \tag{2.40}$$

Calcul de $\frac{\partial u_k^{out}}{\partial u_k^{in}}$:

A partir de l'équation (2.17), nous avons :

$$\frac{\partial u_k^{out}}{\partial u_k^{in}} = f'(u_k^{in}) \tag{2.41}$$

Si la fonction d'activation de la couche cachée est supposée etre une tangente hyperbolique **tanh** :

$$f'\left(u_k^{in}\right) = \tanh'(u_k^{in}) = 1 - \tanh^2\left(u_k^{in}\right) = (1 - u_k^{out})(1 + u_k^{out}) \tag{2.42}$$

Calcul de $\frac{\partial u_k^{in}}{\partial w_{1_{lk}}}$:

Depuis l'équation de propagation (2.16) :

$$\frac{\partial u_k^{in}}{\partial w_{1_{lk}}} = x_l \tag{2.43}$$

On obtient donc :

$$\frac{\partial E}{\partial w_{1_{lk}}} = \left(\sum_j e_j g'\left(Y_j^{in}\right) w_{2_{kj}} \right) . f'\left(u_k^{in}\right) x_l \tag{2.44}$$

Le gradient local δ_j apparaît aussi dans cette expression et l'expression peut s'écrire de cette manière :

$$\frac{\partial E}{\partial w_{1_{lk}}} = \left(\sum_j \delta_j \, w_{2_{kj}} \right) . f'\left(u_k^{in}\right) x_l \tag{2.45}$$

La définition du gradient local pour ce deuxième cas δ_k est donc valable en posant :

$$\delta_k = \left(\sum_j \delta_j \, w_{2_{kj}} \right) . f'\left(u_k^{in}\right) \tag{2.46}$$

Ce qui donne bien :

$$\frac{\partial E}{\partial w_{1_{lk}}} = \delta_k \, x_l \tag{2.47}$$

Finalement la correction des pondérations entre la couche cachée et la couche d'entrée est :

$$w_{1_{lk}} = w_{1_{lk}} - \alpha \, \delta_k \, x_l = w_{1_{lk}} - \alpha \left(\sum_j \delta_j \, w_{2_{kj}} \right) . f'\left(u_k^{in}\right) x_l \tag{2.48}$$

On retrouvera cette récursivité entre les autres couches, et donc la méthode permet d'adapter les poids de n'importe quelle couche du PMC, menant à une diminution progressive de l'erreur de sortie.

Il s'agit maintenant de mettre à jour les pondérations et les biais en fonction de nombre d'échantillons sélectionnés (un échantillon ou plusieurs), et si on pose une architecture d'un PMC avec une activation **tanh** dans la couche cachée et une linéarité (**pureline**) au niveau de la couche de sortie, les équations des pondérations et des biais deviennent :

$$dw_2 = \frac{1}{k}\, e^i\, {u^{out}}^{i^T} \tag{2.49}$$

$$db_2 = \frac{1}{k}\sum_{i=1}^{k} e^i \tag{2.50}$$

$$dw_1 = \frac{1}{k}\left[w_2^T\, e^i \,.\left(1 - u^{out^i}\right).\left(1 + u^{out^i}\right)\right] X^T \tag{2.51}$$

$$db_2 = \frac{1}{k}\sum_{i=1}^{k} w_2^T e^i \,.\, \left(1 - u^{out^i}\right).\left(1 + u^{out^i}\right) \tag{2.52}$$

k : représente le nombre d'exemples injectés dans le PMC.

Notons que la méthode de descente du gradient se formule dans trois formulations, il y'a bien : la méthode stochastique, mini – lot (mini – batch) et le lot (batch).

Si k = 1, la méthode est stochastique, un exemple choisi d'une façon aléatoire qui peut alimenter le PMC.

Si k est compris entre 2 et m-1, l'appellation devient une descente du gradient par mini – lot, le calcul se fait en parallèle pour plusieurs exemples à la fois.

Si k = m, on dit que c'est une méthode de descente du gradient par lot, tous les exemples rentres à la fois, pour une seule itération de propagation et de rétro – propagation.

Les nouveaux poids et biais se trouve par ces équations :

$$W_1^{t+1} = W_1^t - \alpha\, dW_1^t \tag{2.53}$$

$$W_2^{t+1} = W_2^t - \alpha\, dW_2^t \tag{2.54}$$

$$b_1^{t+1} = b_1^t - \alpha\, db_1^t \tag{2.55}$$

$$b_2^{t+1} = b_2^t - \alpha\, db_2^t \tag{2.56}$$

avec t : Itération t.

2.3.7 Les hyper-paramètres

2.3.7.1 La dynamique (Momentum)

La dynamique des réseaux neuronaux est une technique simple qui améliore souvent la vitesse et la précision de la formation. La formation d'un réseau neuronal consiste à trouver des valeurs pour les poids et les biais de sorte que, pour un ensemble donné de valeurs d'entrée, les valeurs de sortie calculées correspondent étroitement aux valeurs cibles connues et correctes. C'est une dynamique newtonienne qui se rajoute à la méthode descende de gradient, ayant des propriétés avantageuses sur l'apprentissage :

- Le Momentum aide lorsqu'il y'a des ravins.
- Les mises à jour des pondérations tendent à s'aligner.
- Atténue les oscillations causées par les grandes courbures [14].

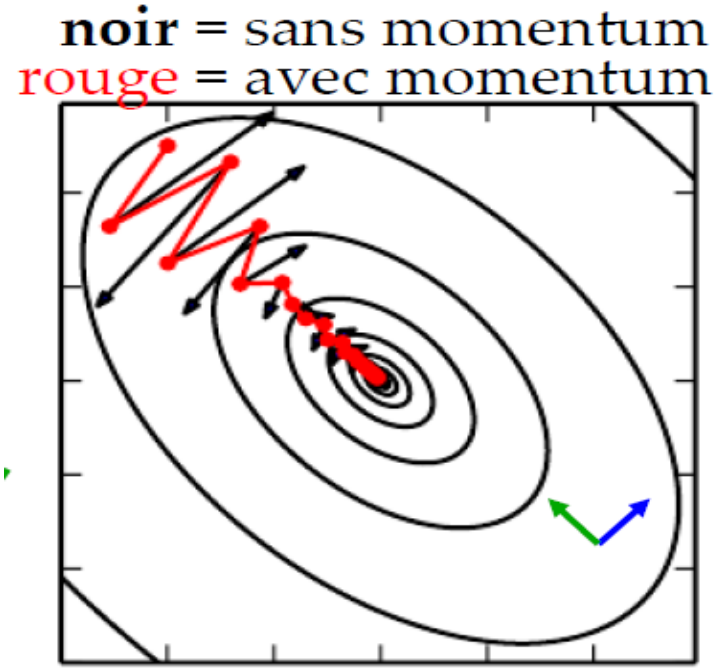


Figure 2.13 : La représentation 2D de la fonction coût *J* par rapport aux pondérations avec et sans Momentum [14]

Remarque :
Si le réseau est initialisé au hasard :

- Peut avoir de très grandes valeurs de gradients au début d'entraînement.
- Judicieux d'utiliser un Momentum de petite valeur, le choix tombe empiriquement sur la valeur 0.5 au début de l'optimisation.
- Augmenter le Momentum par la suite et interprétation des résultats obtenus.

L'introduction de l'hyper – paramètre Momentum se fait par l'insertion de la notion de vecteur **vélocité**, il sert à accélérer le processus de l'apprentissage avec une meilleure approche et précision.

La vélocité des paramètres se traduit par ces équations :

$$V_{dW}^{t+1} = \beta V_{dW}^{t} + (1-\beta) dW^{t} \tag{2.57}$$

$$V_{db}^{t+1} = \beta V_{db}^{t} + (1-\beta)\, db^{t} \tag{2.58}$$

La nouvelle mise à jour des poids et biais depuis (2.53), (2.54), (2.55) et (2.56) se transforme donc :

$$W_{1,2}^{t+1} = W_{1,2}^{t} - \alpha\, V_{dW_{1,2}}^{t+1} \tag{2.59}$$

$$b_{1,2}^{t+1} = b_{1,2}^{t} - \alpha\, V_{db_{1,2}}^{t+1} \tag{2.60}$$

2.3.7.2 Le pas d'apprentissage

Le pas d'apprentissage est un hyper – paramètre configurable utilisé dans la formation des réseaux neuronaux qui a une petite valeur positive, souvent comprise entre 0,0 et 1,0.

Le pas d'apprentissage contrôle la vitesse à laquelle le modèle s'adapte au problème. Les pas d'apprentissage plus faibles nécessitent un plus grand nombre d'époques de formation, étant donné les changements plus faibles apportés aux poids à chaque mise à jour, tandis que les pas d'apprentissage plus importants entraînent des changements rapides et nécessitent moins d'itérations de formation.

Un pas d'apprentissage trop élevé peut faire converger le modèle trop rapidement vers une solution sous-optimale, tandis qu'un pas d'apprentissage trop faible peut bloquer le processus.

Le défi de la formation des réseaux neuronaux d'apprentissage profond consiste à sélectionner soigneusement le pas d'apprentissage. Il peut s'agir de l'hyper – paramètre le plus important du modèle.

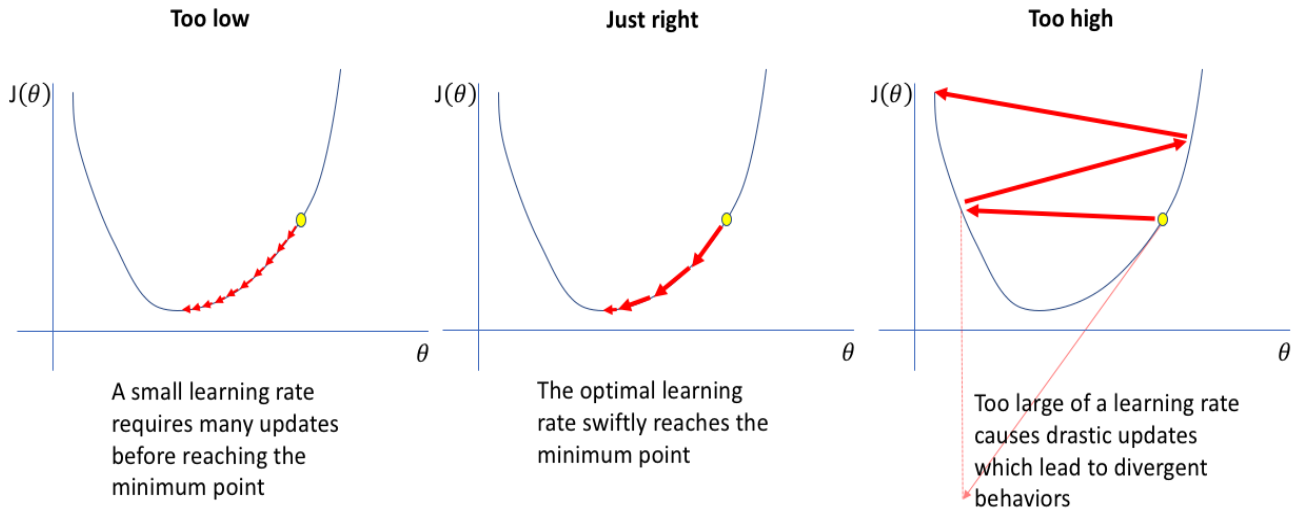


Figure 2.14 : L'influence du pas d'apprentissage sur l'actualisation des pondérations [14]

Il existe des algorithmes qui font optimiser le pas d'apprentissage à chaque itération comme il est possible de le déterminer par plusieurs expériences mais ça reste moins pratique.

2.3.7.3 L'itération (Epoque)

Une époque est un terme utilisé dans l'apprentissage automatique et indique le nombre de passages de l'ensemble des données d'apprentissage que l'algorithme d'apprentissage automatique a effectué. Les ensembles de données sont généralement regroupés en lots (surtout lorsque la quantité de données est très importante). Certaines personnes utilisent le terme "itération" de manière approximative et font référence au passage d'un lot dans le modèle comme une itération.

La détermination du nombre d'époques qu'un modèle doit exécuter pour s'entraîner est basée sur de nombreux paramètres liés à la fois aux données elles-mêmes et à l'objectif du modèle, et bien que des efforts aient été faits pour transformer ce processus en algorithme, une compréhension approfondie des données elles-mêmes est souvent indispensable.

2.3.7.4 Le nombre de neurones dans la couche cachée

Le choix du nombre de neurones dans la couche cachée est une partie très importante du choix de l'architecture globale du réseau neuronal. Bien que cette couche n'interagisse pas directement avec l'environnement externe, elle a une influence considérable sur la sortie finale. Le nombre de neurones de la couche cachée doit être soigneusement étudié.

L'utilisation d'un nombre trop faible de neurones dans la couche cachée entraînera ce que l'on appelle un sous-ajustement. Il y a sous-adaptation lorsqu'il y a trop peu de neurones pour détecter correctement les signaux dans un ensemble de données complexe.

L'utilisation de trop de neurones dans la couche cachée peut entraîner plusieurs problèmes. Premièrement, un trop grand nombre de neurones dans la couche cachée peut engendrer un sur-apprentissage. Ce phénomène se produit lorsque la capacité de traitement de l'information du réseau neuronal est telle que la quantité limitée d'informations contenues dans l'ensemble de formation ne suffit pas à former tous les neurones de la couche cachée. Un deuxième problème peut survenir même lorsque les données de formation sont suffisantes. Un nombre anormalement élevé de neurones dans la couche cachée peut augmenter le temps nécessaire à l'apprentissage du réseau. Le temps de formation peut augmenter au point qu'il est impossible de former le réseau neuronal de manière adéquate. Il est évident qu'un compromis doit être trouvé entre un nombre trop élevé et un nombre trop faible de neurones dans la couche cachée.

Il existe de nombreuses méthodes empiriques pour déterminer le nombre correct de neurones à utiliser dans la couche cachée, telles que les suivantes :

- Le nombre de neurones cachés doit être compris entre le nombre d'entrée et le nombre de neurones de sortie.
- Le nombre de neurones cachés doit être approximativement égal à 2/3 de la taille de la couche d'entrée, plus la taille de la couche de sortie.
- Le nombre de neurones cachés doit être inférieur à deux fois la taille de la couche d'entrée.
- Parfois, on se réfère à utiliser les puissances de 2 (2^n) et tester à chaque fois jusqu'à atteindre les bonnes réponses.

2.3.7.5 La taille du lot (batch size)

La taille du lot est un terme utilisé dans l'apprentissage et fait référence au nombre d'exemples de formation utilisés dans une itération. La taille du lot peut être l'une des trois options suivantes :

- Mode lot : où la taille du lot est égale à l'ensemble des données, toutes ces données rentrent à la fois pour un apprentissage à une seule époque. L'avancement vers le minimum sera un chemin direct et ça omet les oscillations avec une bonne convergence de la méthode de descente du gradient, mais elle demande de stockage et de temps, ça risque en plus de converger vers un minimum local.

- Mode mini – lot : où la taille du lot est supérieure à 1 mais inférieure à la taille totale de l'ensemble de données. Habituellement, il s'agit d'un nombre qui peut être divisé par la taille totale de l'ensemble de données. Il est généralement utilisé pour l'apprentissage en raison de son pouvoir de gérer la mémoire d'une bonne façon, ça peut mener à s'approcher vers un minimum local de la fonction coût, ce qui crée une perturbation pour l'avancement des poids et des biais et qui pourra sortir de ce minimum local. Par expérience, ce mode est stable pour une bonne convergence.

- Mode stochastique : où la taille du lot est égale à 1. Par conséquent, le gradient et les paramètres du réseau neuronal sont mis à jour après chaque échantillon. Cela nécessite moins de mémoire et rapide mais avec avancement long pour atteindre le minimum de la fonction coût. Utilisé généralement pour des bases de données de grande quantité.

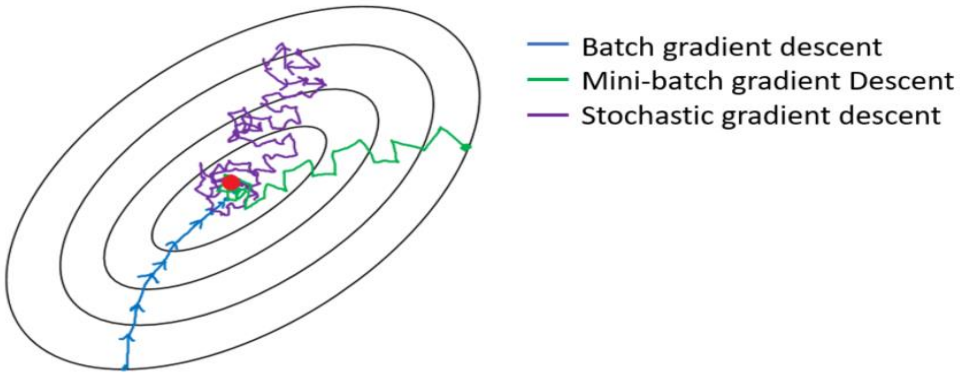


Figure 2.15: Représentation 2D comparative des 3 modes de descente de gradient influant sur l'optimisation de la fonction coût [14]

2.4 Application d'un PMC dans un problème de régression

Essayons maintenant de traiter un exemple de régression pour tracer une fonction donnée dans un intervalle donné dans un plan (x, y) afin de juger si le modèle d'un PMC peut traiter le problème sans connaitre les formulations mathématiques et le comprendre via un apprentissage effectué.

Le choix était pour tracer la fonction de type cubique tel que :

$$f(x) = x^3 + x^2 - x + 1 \tag{2.61}$$

Dans un intervalle $x \in [-1,1]$.

Nous générons 1000 échantillons qui varient entre -1 et 1 d'une façon aléatoire pour faire entrainer le PMC, et 1000 données dans l'intervalle $[-1,1]$ pour valider le modèle construit. L'architecture du PMC utilisé est d'une seule entrée x avec une sortie obtenue y (PMC $1-4-1$).

En fixant les hyper – paramètres suivants :

- Pas d'apprentissage $\alpha = 0.08$.
- Momentum $\beta = 0.5$.
- Nombre de neurones cachés $N_h = 4$.

Nous obtenons les figures (2.16), (2.17), (2.18) et (2.19) pour deux modes : lot, stochastique et pour deux nombres d'itérations appliqués (10000 et 100000 itérations).

a) Itération = 10000, taille du lot = 1

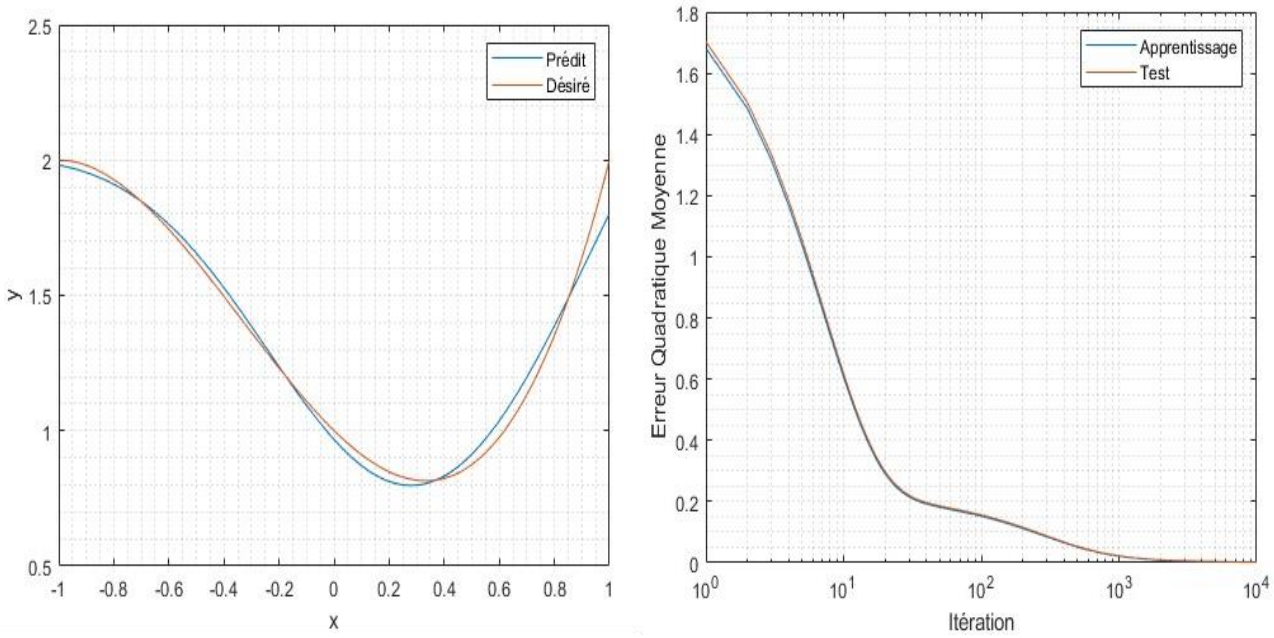


Figure 2.16:Tracés de la fonction par prédiction et par formulation analytique à gauche et représentation de la fonction coût en fonction des itérations à droite pour 10000 itérations et pour une taille de lot de 1

b) Itération = 100000, taille du lot = 1

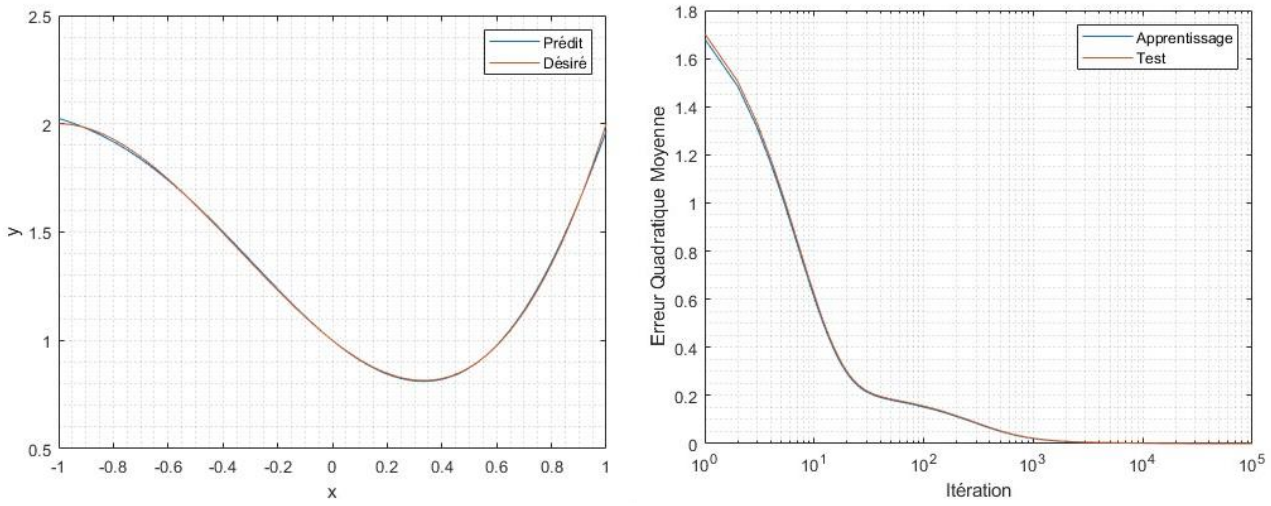


Figure 2.17:Tracés de la fonction par prédiction et par formulation analytique à gauche et représentation de la fonction coût en fonction des itérations à droite pour 100000 itérations et pour une taille de lot de 1

c) Itération =10000, taille du lot = 1000

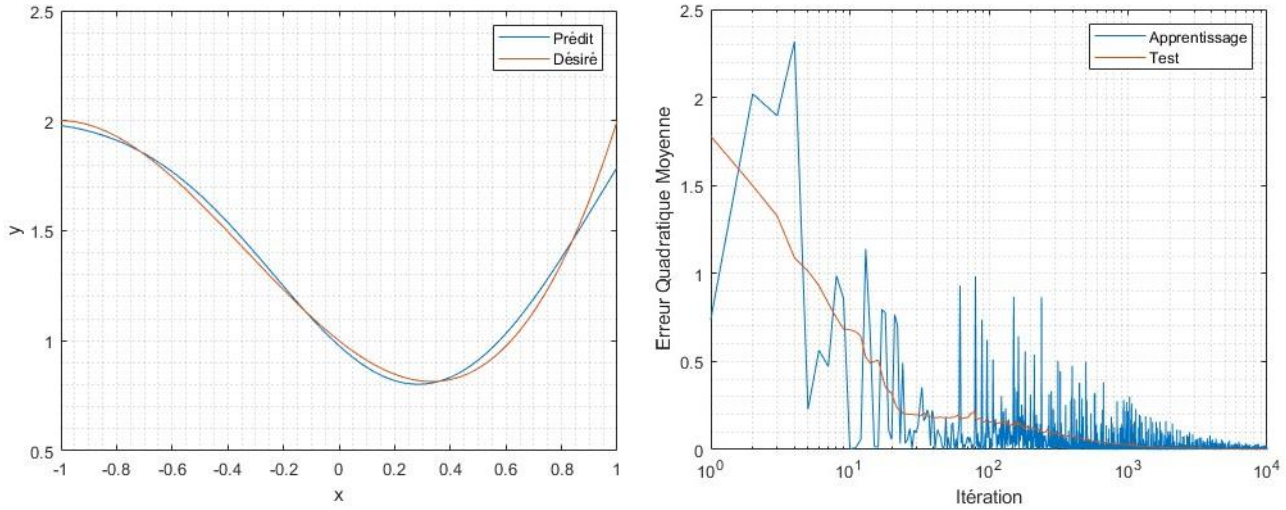


Figure 2.18:Tracés de la fonction par prédiction et par formulation analytique à gauche et représentation de la fonction coût en fonction des itérations à droite pour 10000 itérations et pour une taille de lot de 1000

d) Itération = 100000, taille du lot = 1000

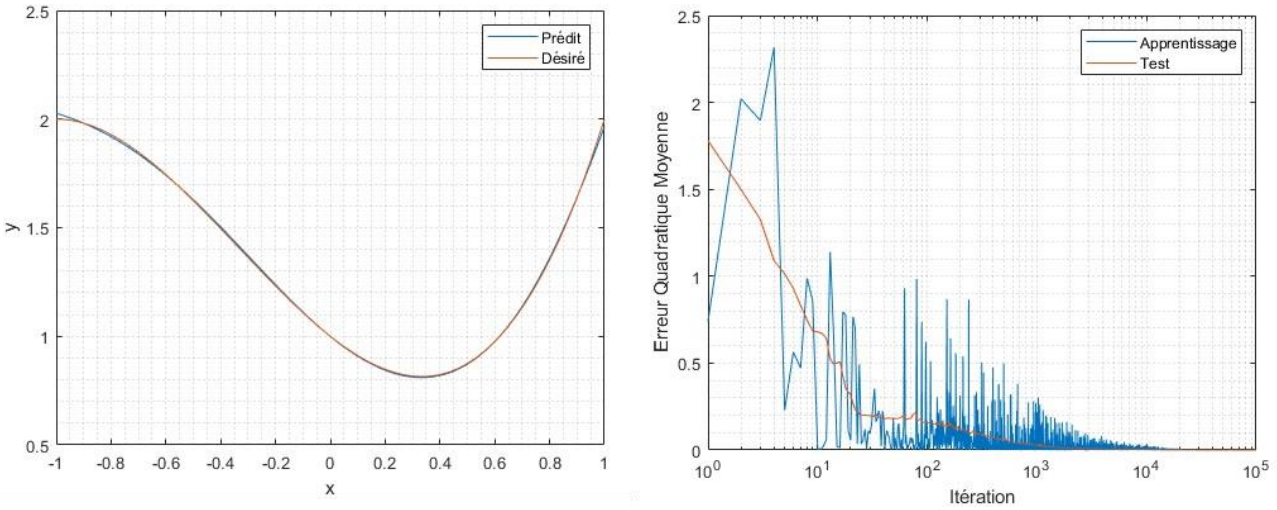


Figure 2.19:Tracés de la fonction par prédiction et par formulation analytique à gauche et représentation de la fonction coût en fonction des itérations à droite pour 100000 itérations et pour une taille de lot de 1000

Interprétation des résultats

Les cas **a)** et **b)** représentent l'apprentissage en mode lot (batch), la taille du batch est de 1 ce qui implique que ce lot contient toutes les données (les 1000 échantillons), l'apprentissage est dit apprentissage parallèle, durant une époque, un cycle de propagation et rétro – propagation se fait avec toutes les données présentes dans le lot, le temps d'apprentissage est lent en raison d'avoir une dimension importante du lot, il nécessite également une taille de mémoire si importante sur une machine à calcul. Le mode lot peut être visualisé via l'allure de l'évolution de l'erreur quadratique moyenne durant l'apprentissage, l'allure est parfaitement lisse, également pour l'allure d'erreur de test, l'évolution des courbes d'erreurs ne fait que se décroître jusqu'à atteindre la convergence où l'erreur tend vers 0 (voir figure (2.16) et (2.17)). Le modèle s'adapte bien puisque les erreurs d'apprentissage et de test converge vers la valeur 0. Les deux courbes à gauche pour le mode lot montrent qu'en augmentant le nombre d'itération de 10000 vers 100000, la sortie prédite se superpose parfaitement avec la sortie désirée, l'erreur dans le cas **b)** est minimale, les courbes se superpose parfaitement. On dit bien que le perceptron multicouche établi a très bien compris le modèle analytique défini.

Dans les cas **c)** et **d)**, nous avons passé au mode d'apprentissage par descente de gradient stochastique, un lot est subdivisé en 1000 mini – lots, c'est-à-dire en 1000 données, l'apprentissage est fait par un échantillon pris d'une façon aléatoire dans une époque. L'apprentissage est rapide puisque la taille du lot est petite (1 seul échantillon au hasard) et il n'y aura pas une taille de mémoire requise importante. Ce mode d'entraînement peut être reconnu en faisant visualiser l'allure de

l'erreur de la phase d'entraînement qui ne fait qu'à osciller jusqu'à atteindre la convergence, en d'autre terme, l'erreur d'apprentissage tend vers 0. Depuis les figures (2.16) et (2.17) à droite, on peut déduire que le modèle apprend d'une manière convenable puisque les erreurs d'apprentissage et de test convergent au même temps vers 0. A gauche des figures citées ci – dessus, et en faisant augmenter le nombre d'itération de 10000 à 100000, les courbes prédite et analytique se superpose parfaitement, nous concluons que le modèle PMC établi pour cet exemple traité est performant et a compris la formulation mathématique sans la connaître, c'est l'objectif de l'implémentation de cette technique intelligente dans notre travail dans l'ingénierie et en particulier dans les réseaux électriques présenté dans le chapitre qui suit.

Chapitre 3 :

Application du Perceptron Multicouche dans le Calcul d'Ecoulement de Puissance.

3. Chapitre 3 : Application du Perceptron Multicouche dans le Calcul d'Ecoulement de Puissance

3.1 Introduction

Ce chapitre propose un modèle et une méthodologie pour l'utilisation de réseaux neuronaux artificiels afin de résoudre le problème de l'écoulement de puissance. Une évaluation des données d'entrée requises par le RNA ainsi que son architecture sont également présentées. Le modèle de réseaux de neurones utilisé dans ce chapitre est le perceptron multicouche, et le processus d'apprentissage est basé sur la rétro – propagation par descente du gradient.

Le perceptron multicouche est l'architecture de réseaux de neurones la plus utilisée dans l'ingénierie, y compris dans les applications de systèmes électriques. Elle sera donc adoptée dans ce chapitre. Un perceptron multicouche est capable de résoudre des problèmes complexes et non linéaires tels que le problème de l'écoulement de puissance dans les réseaux électriques.

3.2 Description du modèle PMC dédié un écoulement de puissance

L'objectif d'introduire le perceptron multicouche afin de pouvoir effectuer un calcul d'écoulement de puissance est de concevoir une boîte noire susceptible de recevoir des données entrantes et obtenir des sorties qui présentent la solution de l'écoulement de puissance pour un réseau choisi. Dans la réalité, c'est une méthode assez efficace et rapide pour un traitement en ligne (on – line load flow) après effectuer un apprentissage sur le réseau électrique en lui-même mais en dehors de l'activité du réseau (off-line) ; une base de données ancienne sera utilisée pour l'apprentissage, le choix de données se fait en fonction de différents facteurs (climat, saison, jour, nuit, …etc.) [2].

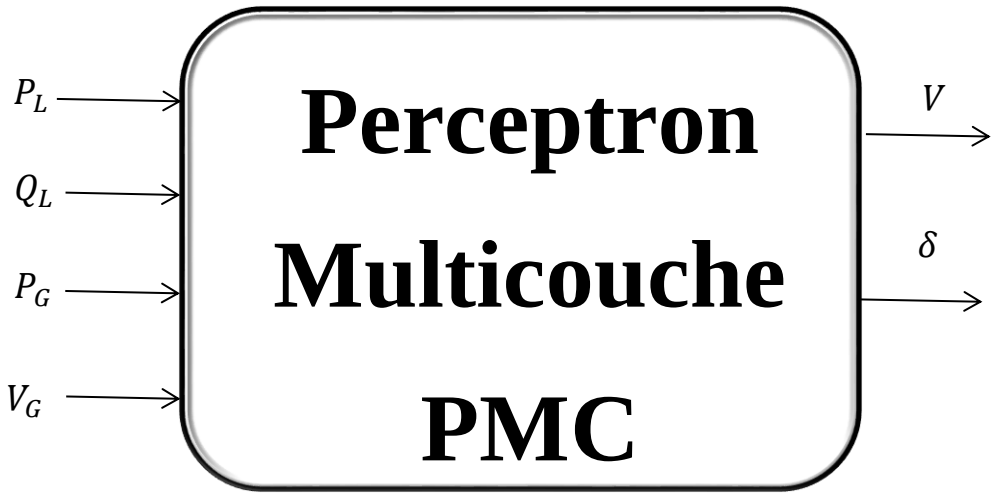


Figure 3.1: Architecture du PMC proposée pour le calcul d'écoulement de puissance [2]

Le modèle établi du PMC à propagation directe à une seule couche cachée est proposé pour fournir le modèle de système électrique souhaité. On suppose que la topologie du réseau neuronal serait suffisante pour reconnaître toute non-linéarité dans le système électrique simulé.

La dimension du vecteur d'entrée du perceptron multicouche serait de $2(npq + npv)$. Les $2(npq + npv)$ entrées représentent la puissance active des npq nœuds de charge et de npv génération, réactive des npq nœuds de charge, ainsi que l'amplitude de la tension des npv nœuds de génération. Si nous remarquons bien, il s'agit de définir les paramètres spécifiés comme des entrées, tandis qu'il faut déduire les tensions en module et phase comme sorties, ce sont des paramètres inconnus [15].

La dimension de la couche de sortie du PMC est de $2npq + npv$ neurones, représentant les amplitudes de tension de charge et les angles des nœuds de charge et génération ", à l'exclusion des valeurs du nœud balancier. La taille de la couche cachée est basée sur une approche par essais et erreurs. Initialement, elle est identique à la taille de la couche de sortie ou couche d'entrée, puis augmentée en fonction du processus d'apprentissage. Les entrées sont représentées par le vecteur X tel que :

$$X = \begin{bmatrix} P_{L_1} \\ \vdots \\ P_{L_{npq}} \\ Q_{L_1} \\ \vdots \\ Q_{L_{npq}} \\ P_{G_1} \\ \vdots \\ P_{G_{npv}} \\ V_{G_1} \\ \vdots \\ V_{G_{npv}} \end{bmatrix} \tag{3.1}$$

Et les sorties sont représentées par le vecteur Y tel que :

$$Y = \begin{bmatrix} V_{L_1} \\ \vdots \\ V_{L_{npq}} \\ \delta_1 \\ \vdots \\ \delta_{(npq+npv)} \end{bmatrix} \tag{3.2}$$

Les poids ajustables de la couche cachée et de la couche de sortie sont représentés par w_1 et w_2 respectivement, ainsi que les biais b_1 et b_2.

Un choix a été imposé sur l'architecture du réseau neuronal, les sorties des neurones de la couche cachée sont activées en utilisant la fonction tangente hyperbolique. Cependant, pour des raisons de commodité, les neurones de la couche de sortie sont activés linéairement en raison d'avoir des

valeurs d'amplitude de tension positives au voisinage de 1 par excès ou par défaut en pu, les valeurs des angles peuvent être positives comme elles peuvent être négative en radian. Le choix de nombre de neurones de la couche cachée s'est fait en effectuant des tests avec les puissances de 2 (2^n) de tel sorte que ce nombre soit supérieur ou égal au nombre d'entrée ou de sortie afin de maîtriser l'avantage de la forte connectivité des neurones entre eux.

Les entrées sont supposées appartenir à des plages spécifiées en pu. Pour les nœuds de charge et de génération, les plages peuvent être déterminées à partir des données relatives aux capacités de charge de la charge/génération et des lignes du système de transmission.

3.3 Analyse du modèle PMC proposé pour le calcul d'écoulement de puissance

3.3.1 Base de données

La sélection de l'algorithme de l'écoulement de puissance utilisé dans le processus de formation du PMC est optionnelle. L'objectif du modèle de réseau neuronal est de fournir une solution au problème du flux de charge avec l'erreur la plus faible possible. Les méthodes Gauss-Seidel, Newton-Raphson (coordonnées rectangulaires et polaires) et le FDLF convergent vers la solution selon des approches totalement différentes. Les quatre méthodes peuvent être utilisées pour la formation du réseau neuronal, chacune compensant les inconvénients de l'autre. Cela améliorera l'approche de la solution de flux de charge du réseau neuronal. Pour notre cas, nous avons opté pour la méthode de Newton-Raphson en coordonnées rectangulaires puisqu'elle converge dans la plupart des cas de figures. Celui-ci nous ramène à la construction de notre base de données, chaque échantillon depuis cette base représente un écoulement de puissance effectué sur un point de fonctionnement à un instant donné. Pour avoir un bon apprentissage du PMC pour l'application dans le flux de puissance, les points de fonctionnement doivent être extraits depuis une prévision de charge pour chaque instant dans un intervalle donné de temps ; 24h à voir quelques semaines.

Vu l'indisponibilité d'une base de données déjà existante que ça soit en théorie depuis les réseaux standards ou en pratique depuis les fournisseurs d'énergie électrique, nous avons établis certains critères afin de pouvoir générer cette base de données d'une façon aléatoire et la diviser en deux section, une grande partie (généralement 70% à 80% des données globales) sera utilisée pour la phase d'apprentissage et le reste de données est dédié pour faire des tests. Le choix a été effectué pour un réseau de 5 nœuds et d'un réseau de 14 nœuds (IEEE 14 bus) afin de valider le modèle établi du perceptron multicouche, leurs données sont mentionnées dans l'annexe.

3.3.2 Procédure d'apprentissage

Le perceptron multicouche de la figure 3.1 est formé pour résoudre le problème de l'écoulement de puissance au moyen de l'algorithme de rétro-propagation. Au cours du processus de formation, un ensemble de valeurs d'entrée est fourni à la fois au modèle de réseau neuronal et à un algorithme normal de l'écoulement de puissance. Les valeurs d'entrée sont sélectionnées aléatoirement dans les plages données. Ensuite, les sorties des deux systèmes sont comparées pour produire des signaux d'erreur. Les signaux sont rétro-propagés vers les couches de sortie et cachées pour l'ajustement du poids [15]. Ce processus est illustré dans le schéma fonctionnel de la figure qui suit :

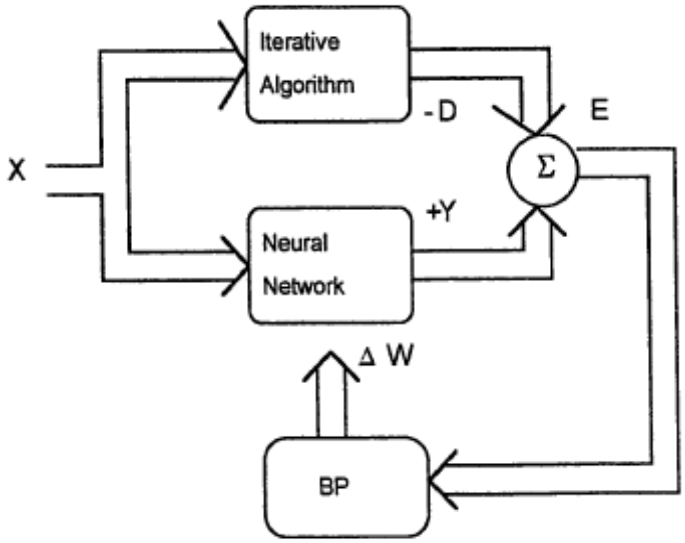


Figure 3.2 : Procédure d'apprentissage du réseau neuronal du système électrique [15]

Où :

X : Données d'entrée.

Y : Sortie prédite.

D : Sortie désirée.

E : Erreur entre la sortie prédite et la sortie désirée.

ΔW : Correction des poids.

BP : Algorithme de rétro-propagation.

Le choix est opté pour un apprentissage via l'algorithme par descente du gradient stochastique en raison de sa convergence rapide et son efficacité ainsi pour le nombre global des échantillons existants ; la base de données n'est pas aussi volumineuse en nombre. La visualisation de la fonction coût peut nous donner des informations sur le bon apprentissage de son échec. Une fois l'entraînement est assuré, l'écoulement de puissance peut se faire sur les données de test en effectuant une propagation directe en introduisant les poids et les biais finaux, pour un point de fonctionnement, la détermination des paramètres inconnus se fait en une seule itération comparant aux méthodes conventionnelles qui nécessitent plus de nombre d'itérations. Ceci est un avantage de la méthode intelligente implémentée lorsqu'il s'agit du traitement en ligne et le concept d'une boîte noire.

Remarque :

- Il est noté que le point de fonctionnement basique présente un point de test puisque la donnée

n'était pas utilisée pour faire entraîner le PMC.

- Lorsqu'on trouve les profils de tensions en amplitude et en phase, les autres paramètres inconnus peuvent être déduits en utilisant la formulation du calcul de la puissance active du nœud balancier et les puissances réactives des nœuds de génération ainsi le nœud balancier. Pour le nœud balancier, la puissance active et réactive se donne par les formules (1.18) et (1.19) en fonction des tensions et de la matrice admittance. Le calcul de puissance réactive des nœuds PV utilise l'équation (1.22).

3.3.3 Réseau 5 nœuds

La résolution de problème d'écoulement de puissance par le PMC peut se faire en choisissant une architecture conforme avec les caractéristiques de ce réseau électrique. Ce réseau comporte un nœud balancier, deux nœuds de génération PV et deux nœuds de charge PQ, d'où le nombre de neurones de la couche d'entrée est deux fois le nombre de nœuds de génération plus ceux de charge, ce qui veut dire qu'on aura 8 entrées, le nombre de neurones de sortie est avec 6 sorties. La construction d'une base de données d'entrées et sorties a été faite par le biais d'un programme écrit avec le logiciel calculateur **MATLAB**, en faisant varier les puissances actives et réactives des nœuds de charge d'une façon aléatoire dans un intervalle de $\pm 30\%$ des valeurs de charge de l'état basique présenté dans l'annexe, varier la puissance active générée des nœuds PV au allant tour de $\pm 10\%$ de la production l'état basique, faire varier aléatoirement les tensions des nœuds de génération dans une marge acceptable entre 0.9 et 1.1 pu, ainsi de modifier également à chaque fois le rapport de transformation des transformateurs d'une façon stochastique dans la plage de 0.9 et 1.1 pu. Ceci peut simuler aux différents points de fonctionnement en un instant bien défini. Il faut bien noter qu'il est recommandé de faire apprendre le PMC par des données soit – disant réaliste. Lors la conception des données, nous nous sommes basés sur le critère de garder le facteur de puissance des charges constant dans une marge de 0.8 à 1 dans le but d'assurer un bon fonctionnement des charges et de ne pas avoir des anomalies. Nous avons pu créer 200 points de fonctionnement du réseau de 5 nœuds avec leurs solutions par la méthode NR, 80% des échantillons sont utilisés pour le processus d'entraînement et 20% pour le test et validation.

Après plusieurs essais des hyper –paramètres pour obtenir les pondérations et le biais finaux, nous avons trouvé des paramètres qui sont considérés optimaux pour avoir un calcul de flux de charge précis :

Tableau 3.1 : Hyper - paramètres optimaux pour le réseau 5 nœuds

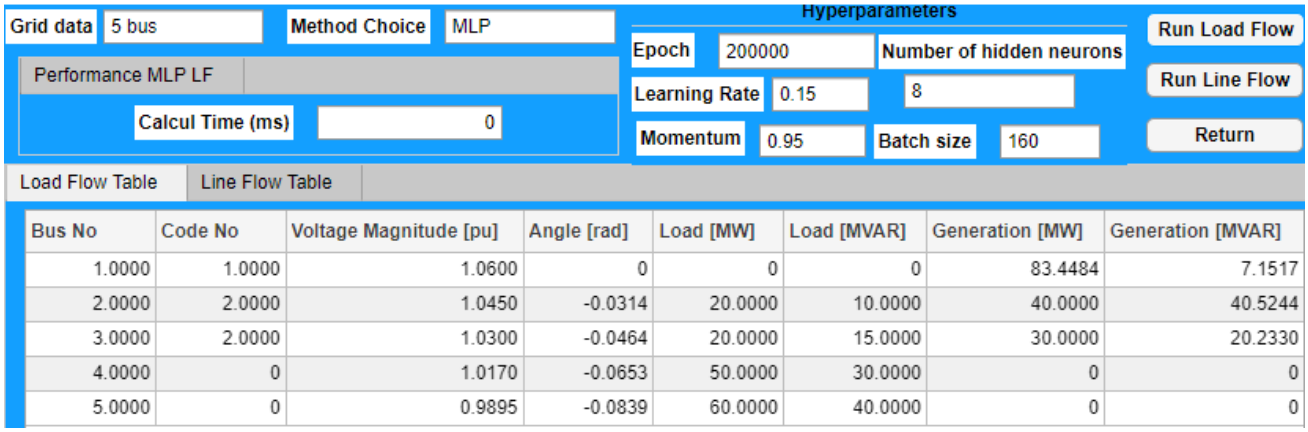
Pas d'apprentissage $\boldsymbol{\alpha}$	Momentum $\boldsymbol{\beta}$	Neurones cachées $\boldsymbol{N_h}$	Epoques	Taille du lot
0.15	**0.95**	**8**	**200000**	**160**

La structure du perceptron **(6 – 8 – 6)** lié au problème du réseau 5 nœuds est illustrée dans le tableau suivant :

Tableau 3.2 : Structure du réseau neuronal dédié au réseau 5 nœuds

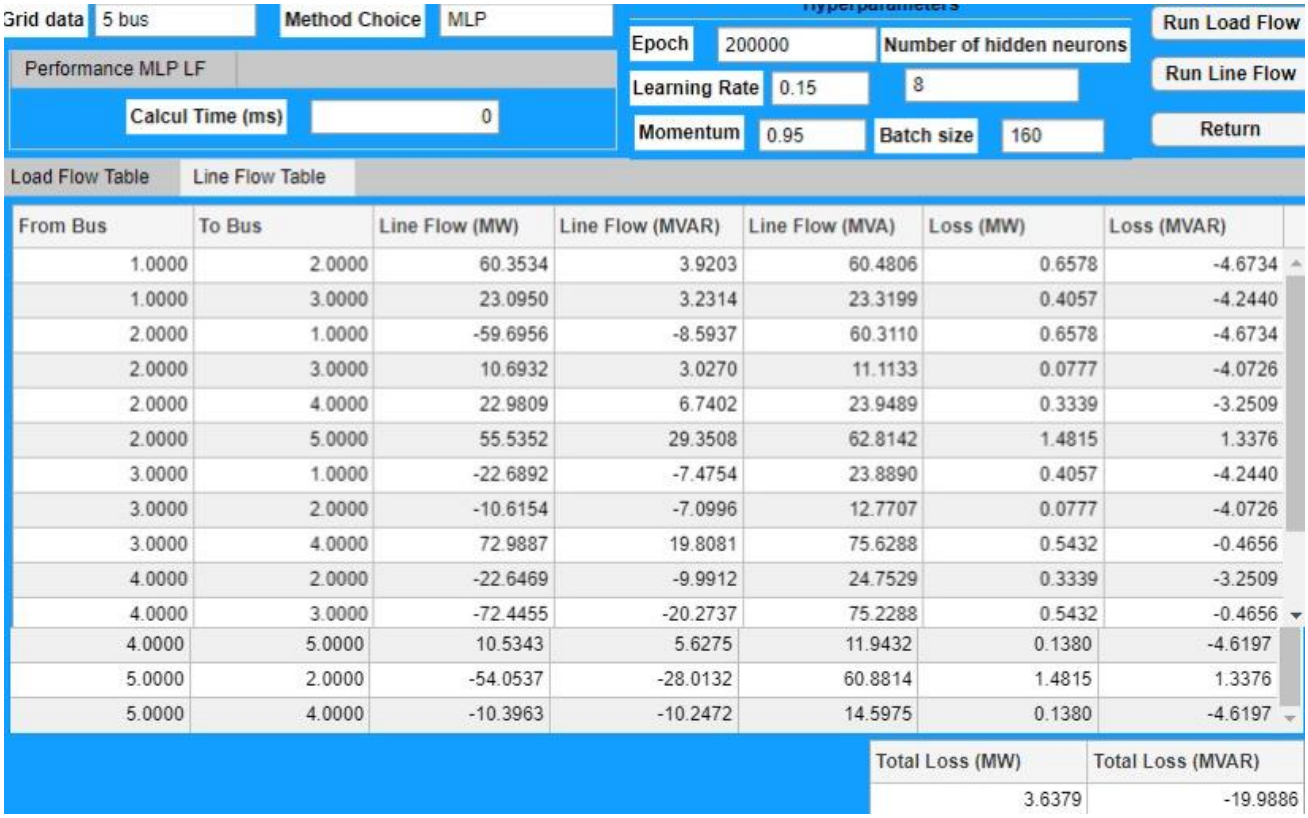
Nombre d'entrées	Neurones de sortie	Neurones cachés	Nombre de pondérations	Nombre de biais
8	**6**	**8**	**112**	**14**

Afin de valider le modèle construit, voici les résultats obtenus du calcul d'écoulement de puissance du point de fonctionnement basique, en utilisant les hyper – paramètres mentionnés en haut :



Bus No	Code No	Voltage Magnitude [pu]	Angle [rad]	Load [MW]	Load [MVAR]	Generation [MW]	Generation [MVAR]
1.0000	1.0000	1.0600	0	0	0	83.4484	7.1517
2.0000	2.0000	1.0450	-0.0314	20.0000	10.0000	40.0000	40.5244
3.0000	2.0000	1.0300	-0.0464	20.0000	15.0000	30.0000	20.2330
4.0000	0	1.0170	-0.0653	50.0000	30.0000	0	0
5.0000	0	0.9895	-0.0839	60.0000	40.0000	0	0

Figure 3.3 : Ecoulement de puissance par PMC pour le réseau 5 nœuds



From Bus	To Bus	Line Flow (MW)	Line Flow (MVAR)	Line Flow (MVA)	Loss (MW)	Loss (MVAR)
1.0000	2.0000	60.3534	3.9203	60.4806	0.6578	-4.6734
1.0000	3.0000	23.0950	3.2314	23.3199	0.4057	-4.2440
2.0000	1.0000	-59.6956	-8.5937	60.3110	0.6578	-4.6734
2.0000	3.0000	10.6932	3.0270	11.1133	0.0777	-4.0726
2.0000	4.0000	22.9809	6.7402	23.9489	0.3339	-3.2509
2.0000	5.0000	55.5352	29.3508	62.8142	1.4815	1.3376
3.0000	1.0000	-22.6892	-7.4754	23.8890	0.4057	-4.2440
3.0000	2.0000	-10.6154	-7.0996	12.7707	0.0777	-4.0726
3.0000	4.0000	72.9887	19.8081	75.6288	0.5432	-0.4656
4.0000	2.0000	-22.6469	-9.9912	24.7529	0.3339	-3.2509
4.0000	3.0000	-72.4455	-20.2737	75.2288	0.5432	-0.4656
4.0000	5.0000	10.5343	5.6275	11.9432	0.1380	-4.6197
5.0000	2.0000	-54.0537	-28.0132	60.8814	1.4815	1.3376
5.0000	4.0000	-10.3963	-10.2472	14.5975	0.1380	-4.6197

Total Loss (MW)	Total Loss (MVAR)
3.6379	-19.9886

Figure 3.4 : Calcul de puissance de transit par PMC

La fonction coût dans ce cas, est représentée en fonction du nombre d'itérations, elle montre l'évolution des erreurs d'apprentissage et de test, illustrée dans la figure qui suit :

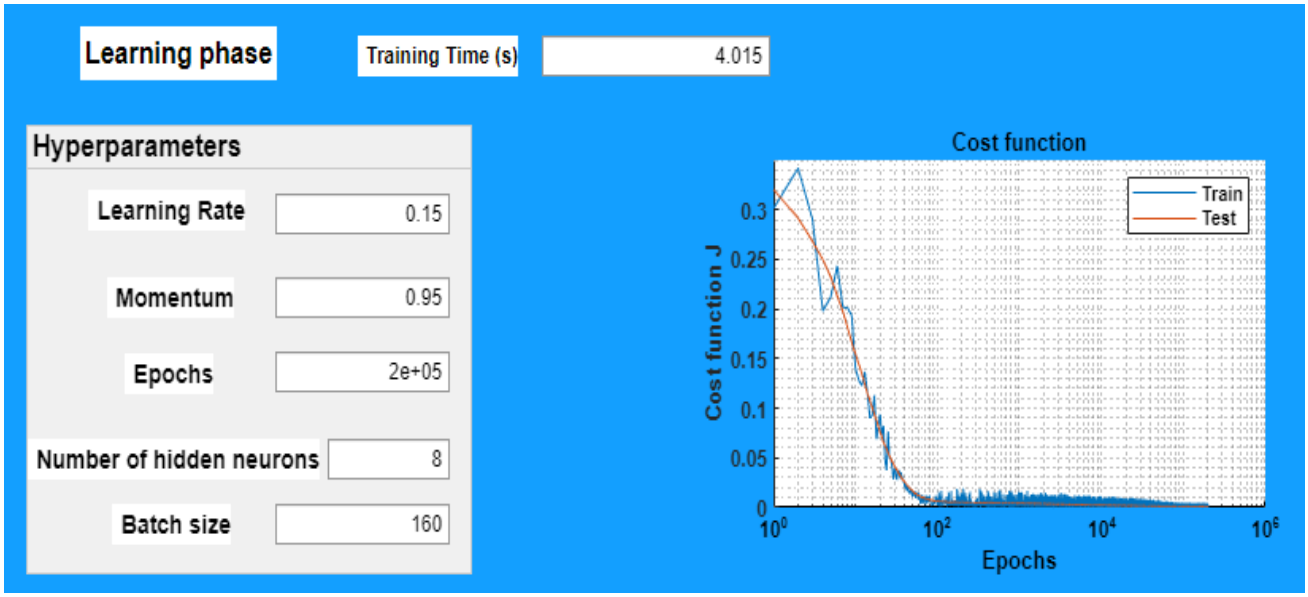


Figure 3.5 : Représentation de la fonction coût en fonction des époques pour le réseau 5 nœuds

Voici également les résultats de l'écoulement de puissance du réseau 5 nœuds par la méthode NR ainsi pour le calcul de puissance de transit :

Grid data: 5 bus | Method Choice: NR - CR | Run Load Flow | Run Line Flow | Return

Performance Iterative LF — Iteration: 4 | Calcul Time (ms): 0

Load Flow Table | Line Flow Table

Bus No	Code No	Voltage Magnitude [pu]	Angle [rad]	Load [MW]	Load [MVAR]	Generation [MW]	Generation [MVAR]
1.0000	1.0000	1.0600	0	0	0	83.0526	7.2710
2.0000	2.0000	1.0450	-0.0311	20.0000	10.0000	40.0000	41.8123
3.0000	2.0000	1.0300	-0.0465	20.0000	15.0000	30.0000	24.1494
4.0000	0	1.0186	-0.0566	50.0000	30.0000	0	0
5.0000	0	0.9901	-0.0769	60.0000	40.0000	0	0

Figure 3.6 : Ecoulement de puissance du réseau 5 nœuds par NR

Grid data 5 bus | Method Choice NR - CR | Run Load Flow | Run Line Flow | Return

Performance Iterative LF

Iteration 4 | Calcul Time (ms) 0

Load Flow Table | Line Flow Table

From Bus	To Bus	Line Flow (MW)	Line Flow (MVAR)	Line Flow (MVA)	Loss (MW)	Loss (MVAR)
1.0000	2.0000	59.9004	4.0557	60.0376	0.6485	-4.7014
1.0000	3.0000	23.1522	3.2153	23.3744	0.4075	-4.2388
2.0000	1.0000	-59.2519	-8.7571	59.8956	0.6485	-4.7014
2.0000	3.0000	10.9137	2.9572	11.3072	0.0800	-4.0660
2.0000	4.0000	18.2174	7.2446	19.6050	0.2312	-3.5657
2.0000	5.0000	50.1209	30.3677	58.6029	1.2954	0.7777
3.0000	1.0000	-22.7447	-7.4540	23.9350	0.4075	-4.2388
3.0000	2.0000	-10.8337	-7.0231	12.9110	0.0800	-4.0660
3.0000	4.0000	43.5784	23.6266	49.5711	0.2365	-1.3891
4.0000	2.0000	-17.9862	-10.8103	20.9849	0.2312	-3.5657
4.0000	3.0000	-43.3419	-25.0157	50.0431	0.2365	-1.3891
4.0000	5.0000	11.3281	5.8260	12.7385	0.1536	-4.5840
5.0000	2.0000	-48.8255	-29.5900	57.0920	1.2954	0.7777
5.0000	4.0000	-11.1745	-10.4100	15.2721	0.1536	-4.5840

Total Loss (MW)	Total Loss (MVAR)
3.0526	-21.7673

Figure 3.7 : Puissance de transit calculée par NR

Comparaison entre la méthode intelligente PMC et la méthode NR :

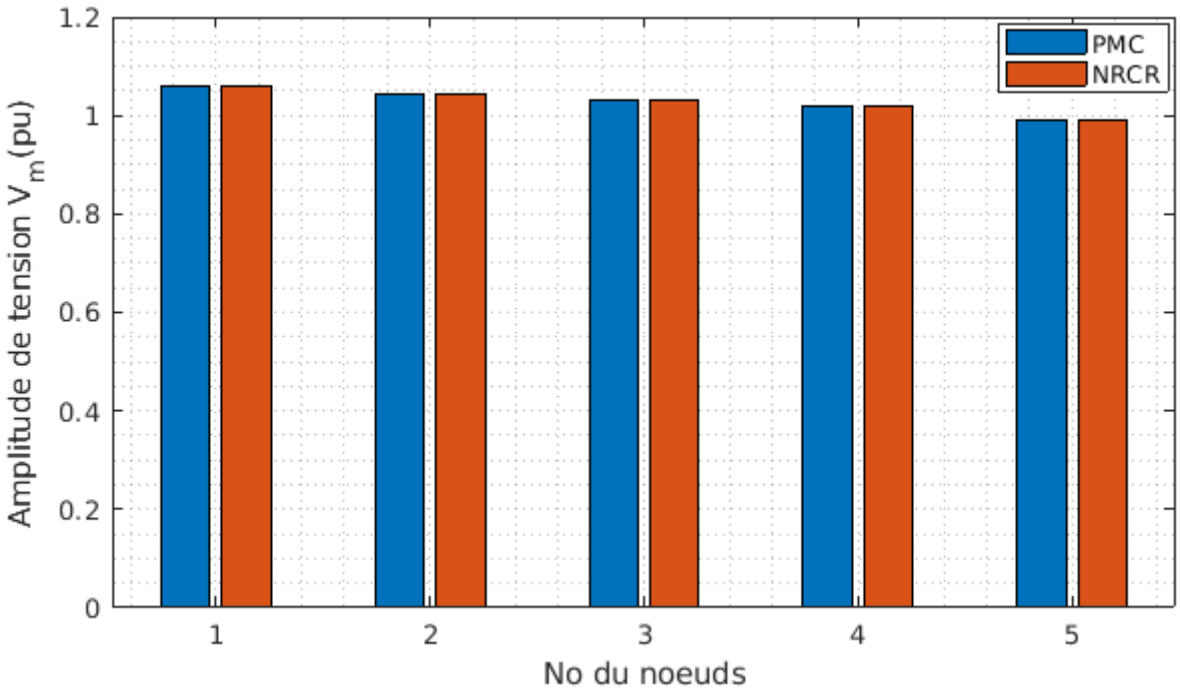


Figure 3.8 : Comparaison des modules de tension pour les deux méthodes

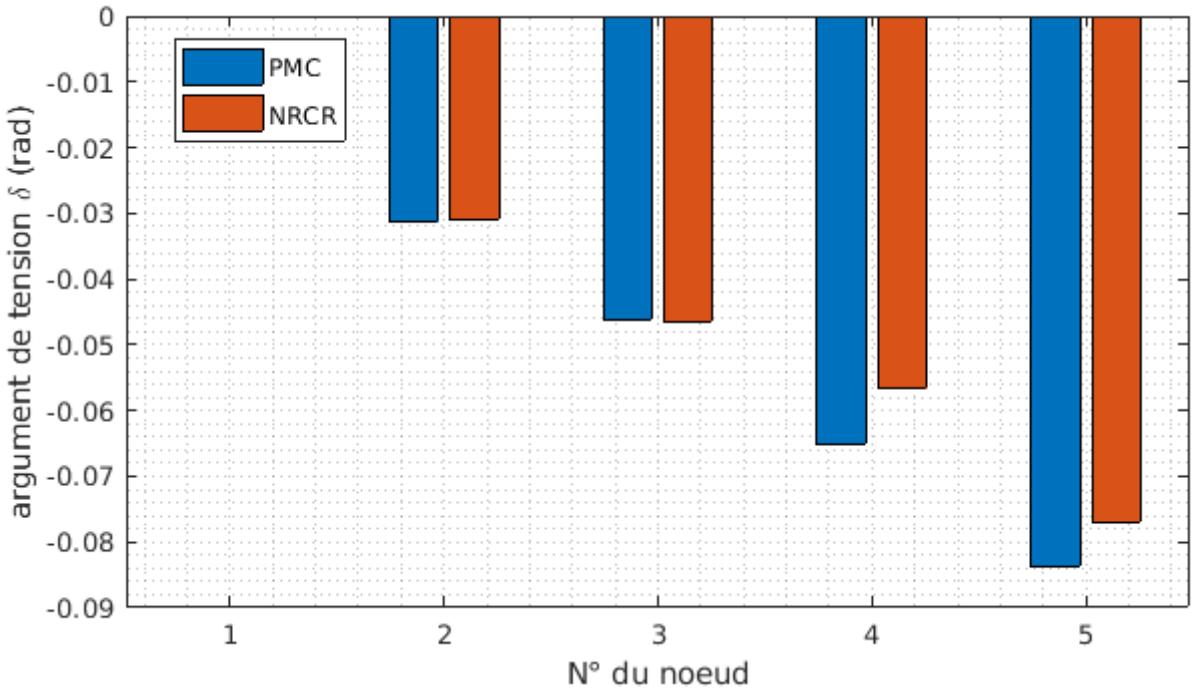


Figure 3.9 : Comparaison des angles de tension pour deux méthodes

3.3.4 Réseau IEEE 14 (14 nœuds)

Pour le réseau standard de 14 nœuds, le concept est le même par rapport au réseau de 5 nœuds, Ce réseau dispose de 4 nœuds de génération PV et 9 nœuds de charge PQ avec un nœud balancier, Ceci nous donne des informations sur le nombre d'entrées et ceux de sortie, l'architecture du PMC lié à ce réseau électrique devient avec 26 entrées et 22 neurones de sortie. La création de la base de données suit la même méthodologie appliquée auparavant dans le cas d'un réseau de 5 nœuds, les marges de valeurs ont été gardées avec les mêmes pourcentages et les valeurs en per unit par rapport au point de fonctionnement basique du réseau électrique 14 nœuds. 75% des données conçues (450 échantillons) sont faits pour l'apprentissage et 25% (150 échantillons), en total 600 données.

En faisant des essais par tâtonnement et par observation de l'évolution de la fonction coût, nous nous sommes tombés sur un bon choix des hyper – paramètres pour le IEEE 14 nœuds :

Tableau 3.3 : Hyper - paramètres optimaux pour le réseau 14 nœuds

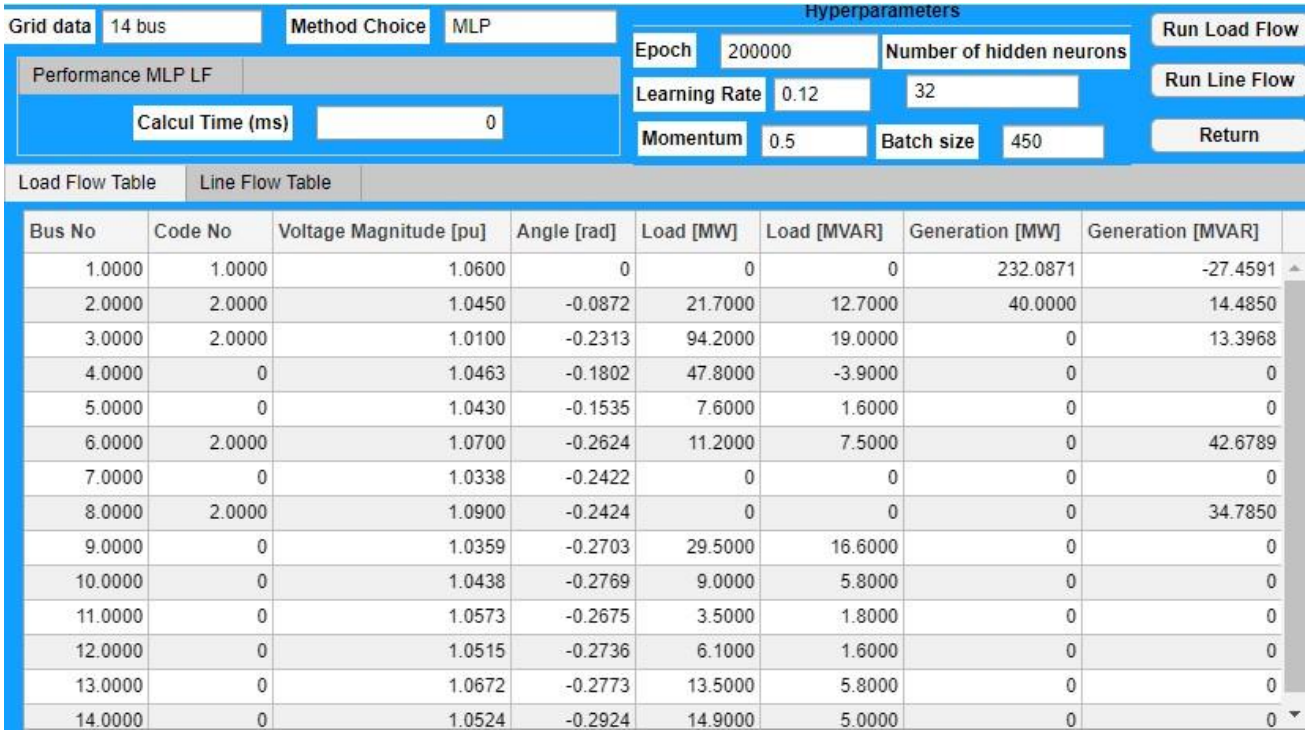
Pas d'apprentissage $\boldsymbol{\alpha}$	Momentum $\boldsymbol{\beta}$	Neurones cachées $\boldsymbol{N_h}$	Epoques	Taille du lot
0.12	**0.5**	**32**	**200000**	**450**

L'architecture du perceptron **(26 – 32 – 22)** dans ce cas est présentée dans le tableau ci – dessous :

Tableau 3.4: Architecture du perceptron lié au problème du réseau IEEE 14 nœuds

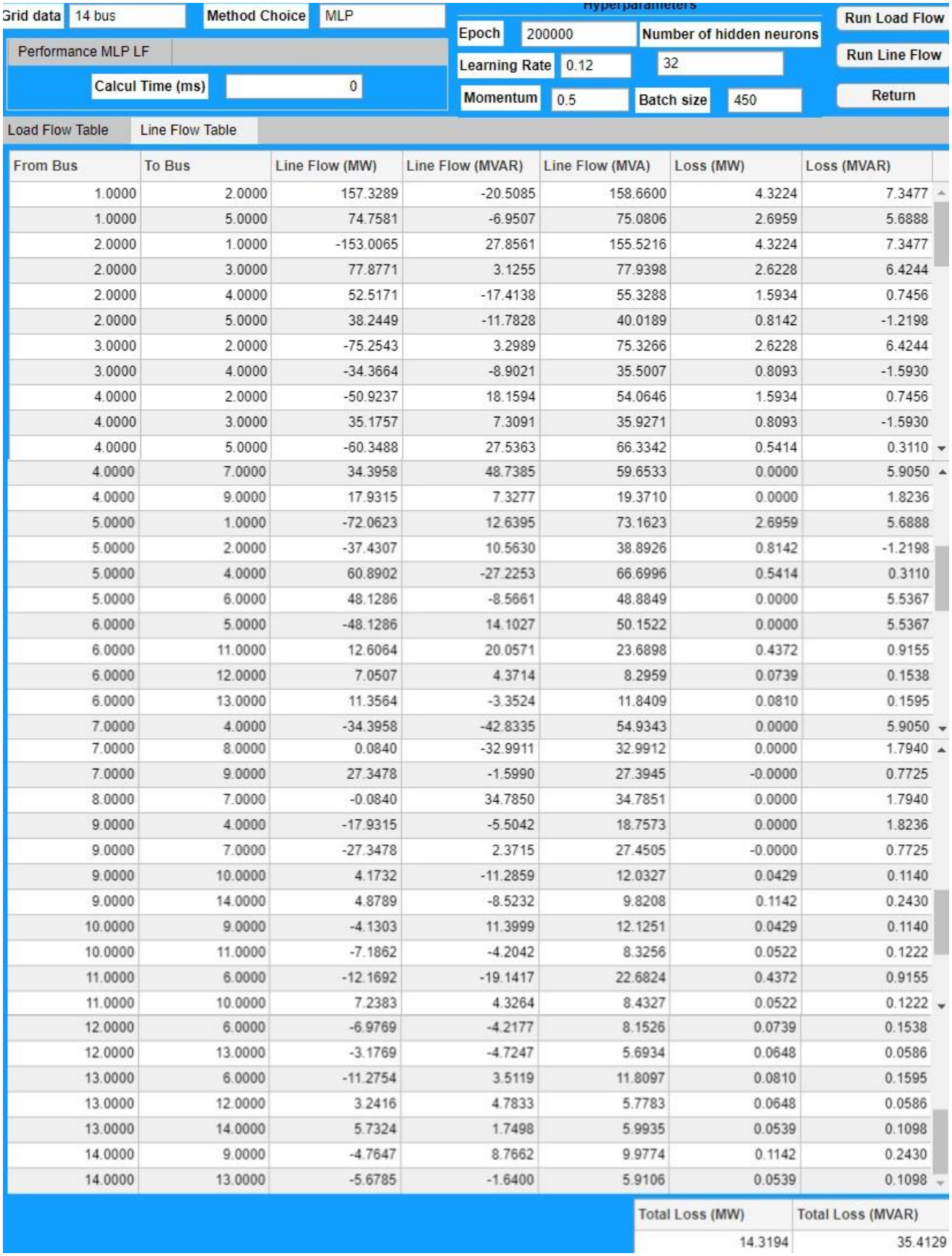
Nombre d'entrées	Neurones de sortie	Neurones cachés	Nombre de pondérations	Nombre de biais
26	**22**	**32**	**1536**	**54**

Prenons le point de fonctionnement basique du réseau 14 nœuds comme un point de test et appliquons l'écoulement de puissance par le PMC et comparant avec le NR :



Bus No	Code No	Voltage Magnitude [pu]	Angle [rad]	Load [MW]	Load [MVAR]	Generation [MW]	Generation [MVAR]
1.0000	1.0000	1.0600	0	0	0	232.0871	-27.4591
2.0000	2.0000	1.0450	-0.0872	21.7000	12.7000	40.0000	14.4850
3.0000	2.0000	1.0100	-0.2313	94.2000	19.0000	0	13.3968
4.0000	0	1.0463	-0.1802	47.8000	-3.9000	0	0
5.0000	0	1.0430	-0.1535	7.6000	1.6000	0	0
6.0000	2.0000	1.0700	-0.2624	11.2000	7.5000	0	42.6789
7.0000	0	1.0338	-0.2422	0	0	0	0
8.0000	2.0000	1.0900	-0.2424	0	0	0	34.7850
9.0000	0	1.0359	-0.2703	29.5000	16.6000	0	0
10.0000	0	1.0438	-0.2769	9.0000	5.8000	0	0
11.0000	0	1.0573	-0.2675	3.5000	1.8000	0	0
12.0000	0	1.0515	-0.2736	6.1000	1.6000	0	0
13.0000	0	1.0672	-0.2773	13.5000	5.8000	0	0
14.0000	0	1.0524	-0.2924	14.9000	5.0000	0	0

Figure 3.10 : Ecoulement de puissance du réseau IEEE 14 par PMC



From Bus	To Bus	Line Flow (MW)	Line Flow (MVAR)	Line Flow (MVA)	Loss (MW)	Loss (MVAR)
1.0000	2.0000	157.3289	-20.5085	158.6600	4.3224	7.3477
1.0000	5.0000	74.7581	-6.9507	75.0806	2.6959	5.6888
2.0000	1.0000	-153.0065	27.8561	155.5216	4.3224	7.3477
2.0000	3.0000	77.8771	3.1255	77.9398	2.6228	6.4244
2.0000	4.0000	52.5171	-17.4138	55.3288	1.5934	0.7456
2.0000	5.0000	38.2449	-11.7828	40.0189	0.8142	-1.2198
3.0000	2.0000	-75.2543	3.2989	75.3266	2.6228	6.4244
3.0000	4.0000	-34.3664	-8.9021	35.5007	0.8093	-1.5930
4.0000	2.0000	-50.9237	18.1594	54.0646	1.5934	0.7456
4.0000	3.0000	35.1757	7.3091	35.9271	0.8093	-1.5930
4.0000	5.0000	-60.3488	27.5363	66.3342	0.5414	0.3110
4.0000	7.0000	34.3958	48.7385	59.6533	0.0000	5.9050
4.0000	9.0000	17.9315	7.3277	19.3710	0.0000	1.8236
5.0000	1.0000	-72.0623	12.6395	73.1623	2.6959	5.6888
5.0000	2.0000	-37.4307	10.5630	38.8926	0.8142	-1.2198
5.0000	4.0000	60.8902	-27.2253	66.6996	0.5414	0.3110
5.0000	6.0000	48.1286	-8.5661	48.8849	0.0000	5.5367
6.0000	5.0000	-48.1286	14.1027	50.1522	0.0000	5.5367
6.0000	11.0000	12.6064	20.0571	23.6898	0.4372	0.9155
6.0000	12.0000	7.0507	4.3714	8.2959	0.0739	0.1538
6.0000	13.0000	11.3564	-3.3524	11.8409	0.0810	0.1595
7.0000	4.0000	-34.3958	-42.8335	54.9343	0.0000	5.9050
7.0000	8.0000	0.0840	-32.9911	32.9912	0.0000	1.7940
7.0000	9.0000	27.3478	-1.5990	27.3945	-0.0000	0.7725
8.0000	7.0000	-0.0840	34.7850	34.7851	0.0000	1.7940
9.0000	4.0000	-17.9315	-5.5042	18.7573	0.0000	1.8236
9.0000	7.0000	-27.3478	2.3715	27.4505	-0.0000	0.7725
9.0000	10.0000	4.1732	-11.2859	12.0327	0.0429	0.1140
9.0000	14.0000	4.8789	-8.5232	9.8208	0.1142	0.2430
10.0000	9.0000	-4.1303	11.3999	12.1251	0.0429	0.1140
10.0000	11.0000	-7.1862	-4.2042	8.3256	0.0522	0.1222
11.0000	6.0000	-12.1692	-19.1417	22.6824	0.4372	0.9155
11.0000	10.0000	7.2383	4.3264	8.4327	0.0522	0.1222
12.0000	6.0000	-6.9769	-4.2177	8.1526	0.0739	0.1538
12.0000	13.0000	-3.1769	-4.7247	5.6934	0.0648	0.0586
13.0000	6.0000	-11.2754	3.5119	11.8097	0.0810	0.1595
13.0000	12.0000	3.2416	4.7833	5.7783	0.0648	0.0586
13.0000	14.0000	5.7324	1.7498	5.9935	0.0539	0.1098
14.0000	9.0000	-4.7647	8.7662	9.9774	0.1142	0.2430
14.0000	13.0000	-5.6785	-1.6400	5.9106	0.0539	0.1098

Total Loss (MW)	Total Loss (MVAR)
14.3194	35.4129

Figure 3.11 : Calcul de puissance de transit par PMC

Visualisons maintenant la courbe de la fonction de coût en fonction de nombre d'époques choisi :

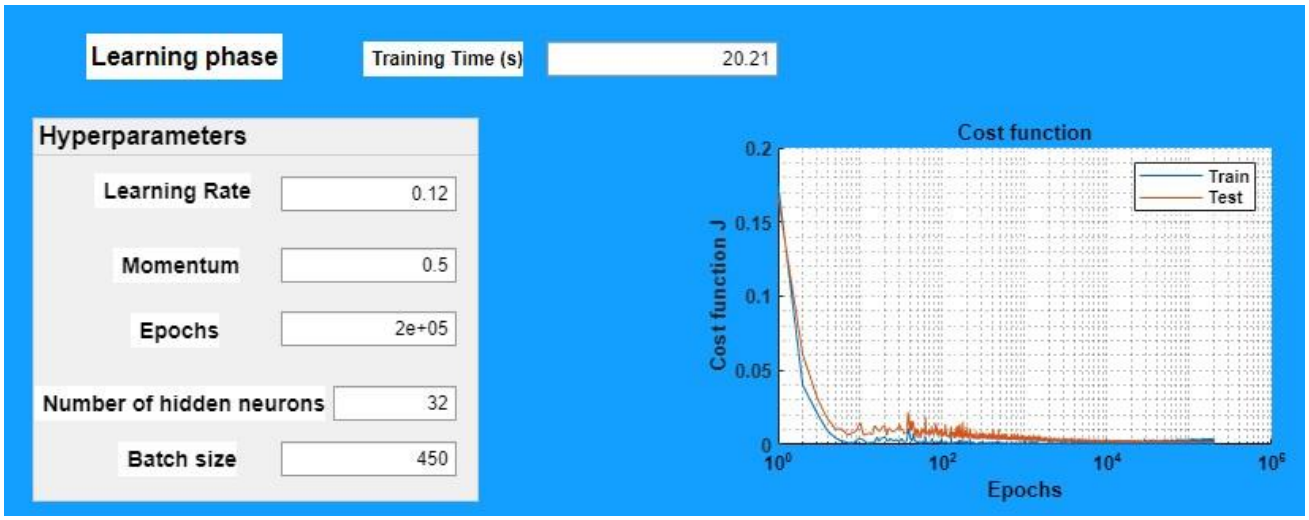


Figure 3.12 : Représentation de la fonction coût en fonction des époques pour le réseau IEEE 14

Comparaison entre le PMC et le NR

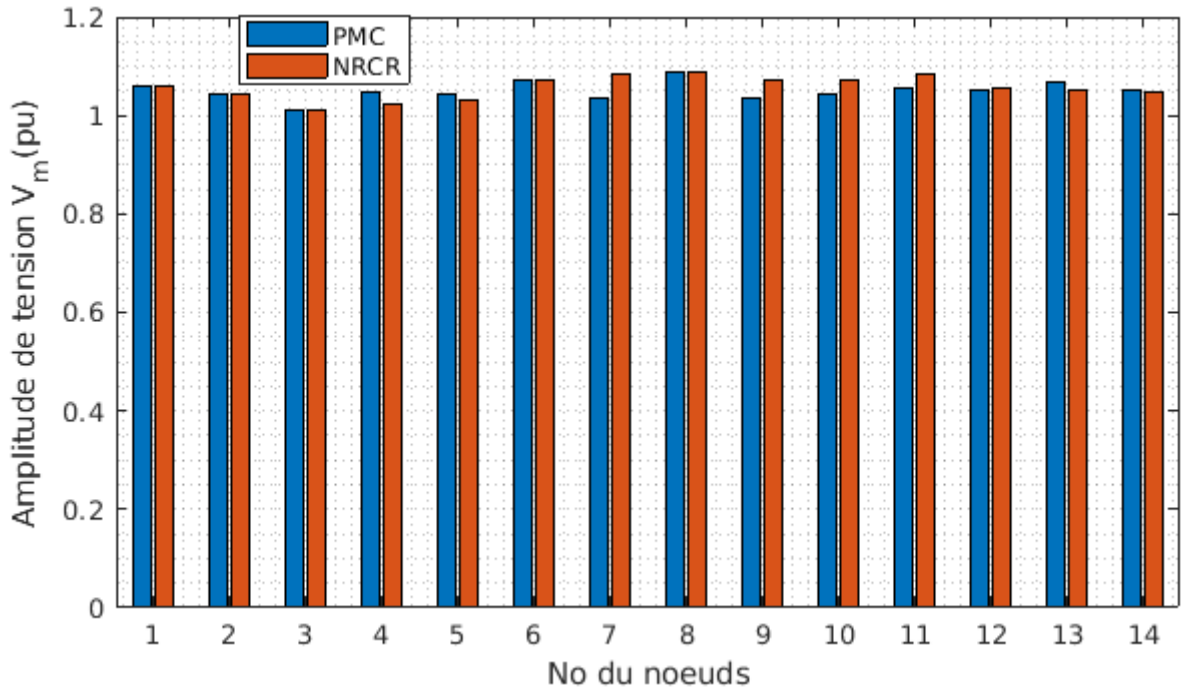


Figure 3.13 : Comparaison des modules de tension pour les deux méthodes

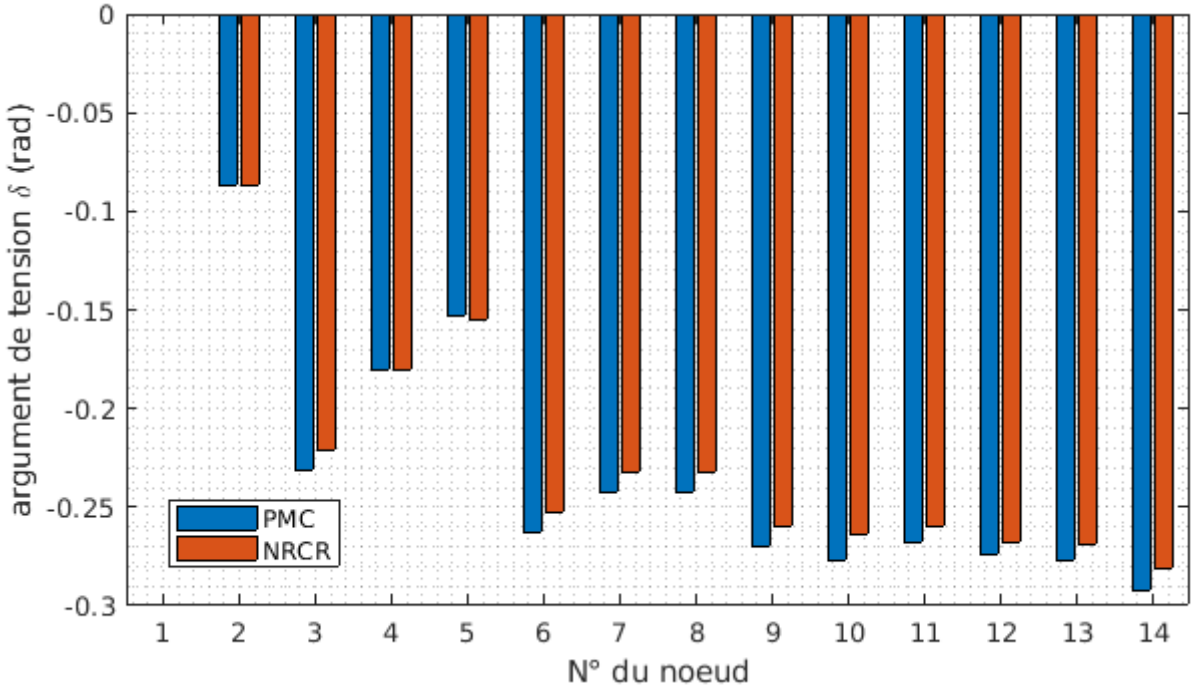


Figure 3.14 : Comparaison des angles de tension pour les deux méthodes

3.4 Discussion des résultats

Réseau 5 nœuds

Une série d'analyse des résultats obtenus par le NR et la PMC utilisés sur les réseaux de 5 nœuds et celui de 14 nœuds génère toute une série de constatation et observation.

Pour le réseau de 5 nœuds, les résultats d'écoulement de puissance obtenus à partir des deux méthodes NR d'une part, et PMC d'autre part, et après comparaison (fig : 3.3- 3.9) des amplitudes et angles de tension, ces derniers sont très proches, ce qui valide le modèle intelligent élaboré pour le réseau électrique étudié. Les puissances de transit déduites par le PMC se rapprochent de celles trouvées par la méthode NR, les pertes actives et réactives totales pour le cas d'emploi du perceptron multicouche sont respectivement 3.63 MW et -19.98 MVAR, tandis que pour le NR, elles sont de 3.05 MW et de -21.76 MVAR, ce qui représente 4% de pertes actives par rapport à la puissance active générée totale, et que les pertes réactives peuvent être compensées. Les amplitudes des tensions trouvées par le perceptron sont comprises entre 0.9 et 1.1, assurant la stabilité de la tension dans chaque nœud. Pour les angles de tension, les valeurs ne dépassent pas les 5° en absolu, ce qui assure une stabilité d'angle au niveau des nœuds, spécialement des nœuds de génération PV.

En ce qui concerne les performances des deux algorithmes abordés dans notre cas (réseau 5 nœuds), le PMC semble être plus performant que la méthode NR en termes de temps de calcul et de nombre d'itérations. En effet, le traitement en ligne pour le PMC s'effectue en un temps très court de l'ordre de quelques fractions de seconde, avec une seule itération par propagation directe, tandis que la

méthode NR nécessite 4 itérations pour atteindre sa convergence. La rapidité de calcul de la méthode NR est due à la faible taille du réseau de 5 nœuds.

Pour rappel, le PMC a nécessité une phase d'apprentissage, pour laquelle, les valeurs optimales des hyper – paramètres testés et trouvés par tâtonnement. L'apprentissage par descente de gradient stochastique a nécessité 4.015 s avec des erreurs d'apprentissage pour converger. L'allure de l'erreur d'apprentissage est un peu bruitée en raison d'utilisation de mode stochastique dans la rétro – propagation, la minimisation de la fonction de coût se fait par des pas allant dans des directions aléatoires mais en assurant la convergence du perceptron. L'allure de l'erreur peut devenir lisse si on bascule vers le mode de lot mais les résultats seront moins précis par expérience.

Réseau 14 nœuds

Au cours de l'apprentissage du perceptron multicouche dédié au réseau électrique du 14 nœuds avec ses hyper – paramètres optimaux, nous avons remarqué dans la figure (3.12) que la fonction coût, pour les cas d'apprentissage et de test, évolue au sens d'avoir une convergence telle que l'erreur tend vers 0, ce qui fait que le PMC a très bien appris le problème de l'écoulement de puissance de ce réseau électrique, les pondérations obtenues et les biais également sont des grandeurs optimale qui peuvent être utilisées pour la prédiction de la solution de l'écoulement de puissance du réseau 14 nœuds en appliquant le processus de la propagation. Le temps d'apprentissage est un peu long comparativement au cas du réseau de 5 nœuds (environs de 20 s). Évidemment, la taille du réseau devient importante et l'architecture du PMC devient plus dense en termes de nombre de neurones et de connexions reliant ses neurones. Le choix de nombre de neurones dans la couche cachée a été fixé à 32 neurones, pour 26 entrées et 22 neurones de sortie, la valeur 32 étant la puissance de 2 la plus proche du nombre d'entrées et de sorties, ce qui donne un avantage sur la bonne convergence du modèle intelligent proposé.

Les résultats obtenus de l'écoulement de puissance par le PMC illustré dans la figure (3.10), en tension (amplitudes et angles), puissances actives et réactives, se rapprochent des résultats obtenus dans le chapitre 1 présentés dans la figure (1.9) par la méthode NR, les erreurs étant minimales. Les résultats montrent que le système de transport d'énergie à 14 nœuds est stable, les valeurs de tensions en module de chaque nœud sont admissibles dans la plage de 0.9 et 1.1 pu. Les valeurs des angles ne dépassent pas en absolu les 17°, ce qui implique que la stabilité en angle est assurée. Le calcul de puissance de transit par PMC présenté dans la figure (3.11), donne des résultats proches des valeurs calculées par la méthode NR présentées dans le chapitre 1, figure (1.10). Les pertes actives et réactives calculées par le PMC sont bien égales à 14.32 MW et 35.41 MVAR, tandis que pour la méthode NR, on trouve bien des valeurs de pertes actives et réactives qui sont de 13.36 MW et de 26.12 MVAR respectivement. Le pourcentage de pertes actives présentées dans ce réseau est de l'environ de 5%. Cela nous montre que le modèle du PMC établi pour le réseau IEEE 14 nœuds fonctionne bien.

En terme de performance, le temps de calcul d'écoulement de puissance par PMC est très court ; en une seule itération, comparant à la méthode NR qui nécessite 5 itérations pour atteindre la convergence en 1 ms au moyenne. Eventuellement, le PMC est rapide dans ce cas (réseau IEEE 14

nœuds), le point de fonctionnement basique figuré dans l'annexe est considéré comme une donnée de test, une phase de propagation est suffisante pour prédire la solution de problème d'écoulement de puissance. Pour le réseau 14 nœuds, la méthode NR nécessite une correction de l'erreur dans chaque itération pour atteindre la convergence.

Chapitre 4 :

Présentation de
l'Application

«Intelligent Load Flow
Calculator ILFC»

4. Chapitre 4 : Présentation de l'Application " Intelligent Load Flow Calculator - ILFC "

4.1 Introduction

Dans ce chapitre, nous avons élaboré un programme utilisant les différentes techniques classiques et intelligentes présentées dans ce mémoire, pour la résolution du problème de l'écoulement de puissance dans les réseaux électriques, ainsi qu'une interface graphique regroupant les applications de ce programme.

L'interface graphique que nous avons conçue, est d'utilisation simple, acceptant des modifications ou amélioration possible, puisque nous pouvons y ajouter d'autres programmes et y insérer de nouvelles méthodes d'intelligence artificielle de calcul d'écoulement de puissance ou autres méthodes.

4.2 Interface graphique

La figure (4.1) représente la page d'accueil de cet interface graphique lors du lancement du programme **ILFC** (Mainapp.mlapp). L'utilisation de l'interface comporte des fenêtres faciles à utiliser, que nous présentons dans ce qui suit.

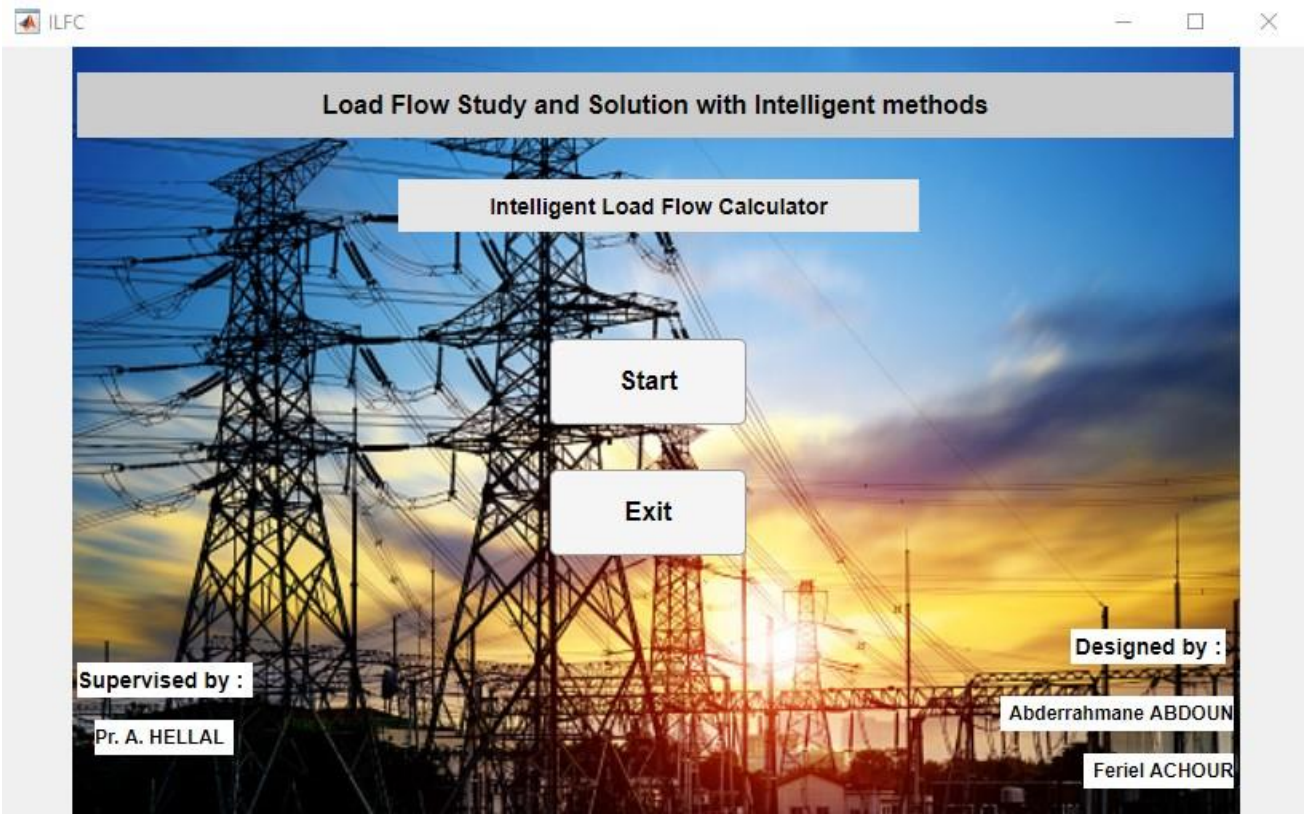


Figure 4.1 : Fenêtre d'accueil lors du lancement du programme

En appuyant sur le bouton **Start** de la fenêtre d'accueil, une autre fenêtre apparaît qui affiche le choix du réseau et le choix des méthodes existantes pour le calcul d'écoulement de puissance. Le bouton **Exit** permet de sortir de l'application.

-Choisir le réseau électrique étudié

L'utilisateur peut choisir un réseau électrique parmi les réseaux préexistants dans le programme élaboré.

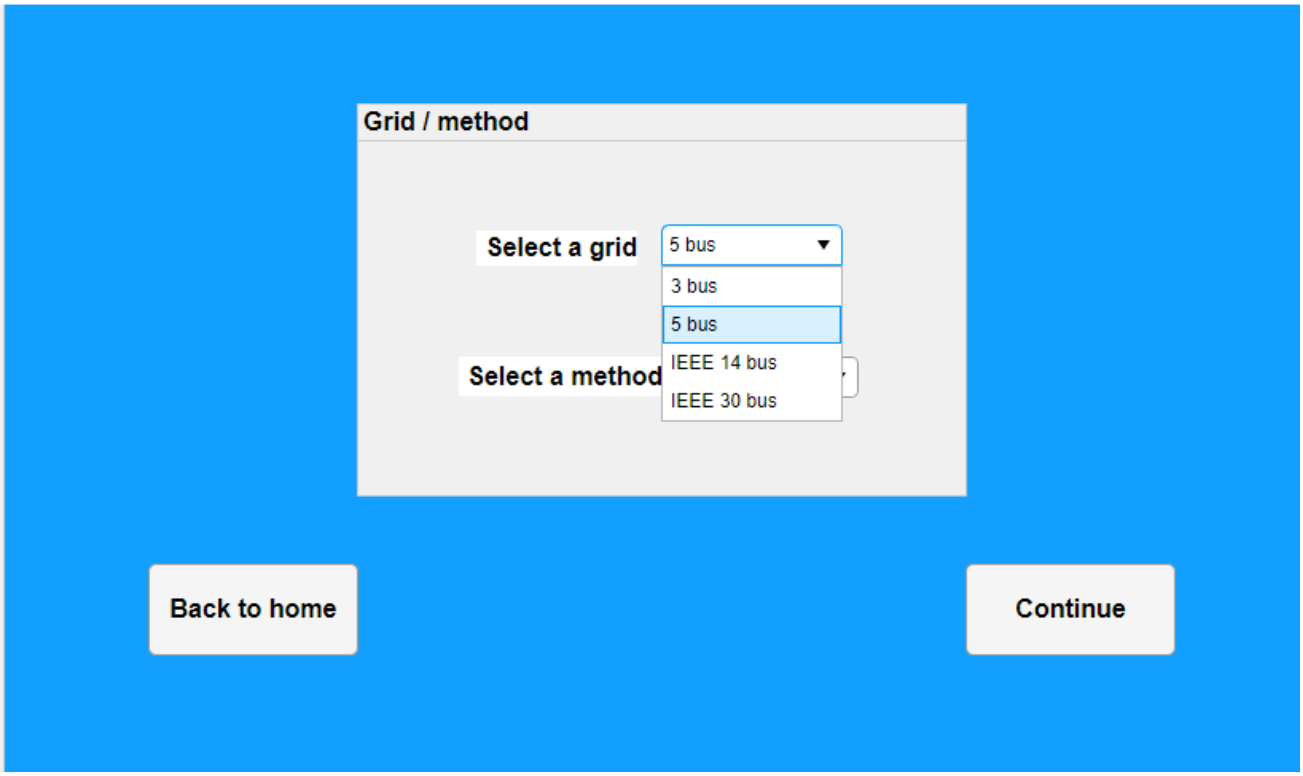


Figure 4.2 : Choix du réseau électrique

- Choisir la méthode de calcul d'écoulement de puissance

Le choix de la méthode est à faire parmi la liste qui s'affiche : Gauss – Seidel (GS), Newton – Raphson avec ses deux formes rectangulaires et polaires (NRCR et NRCP), la méthode Découplée Rapide (FDLF) et notre méthode intelligente : le perceptron multicouche (MLP).

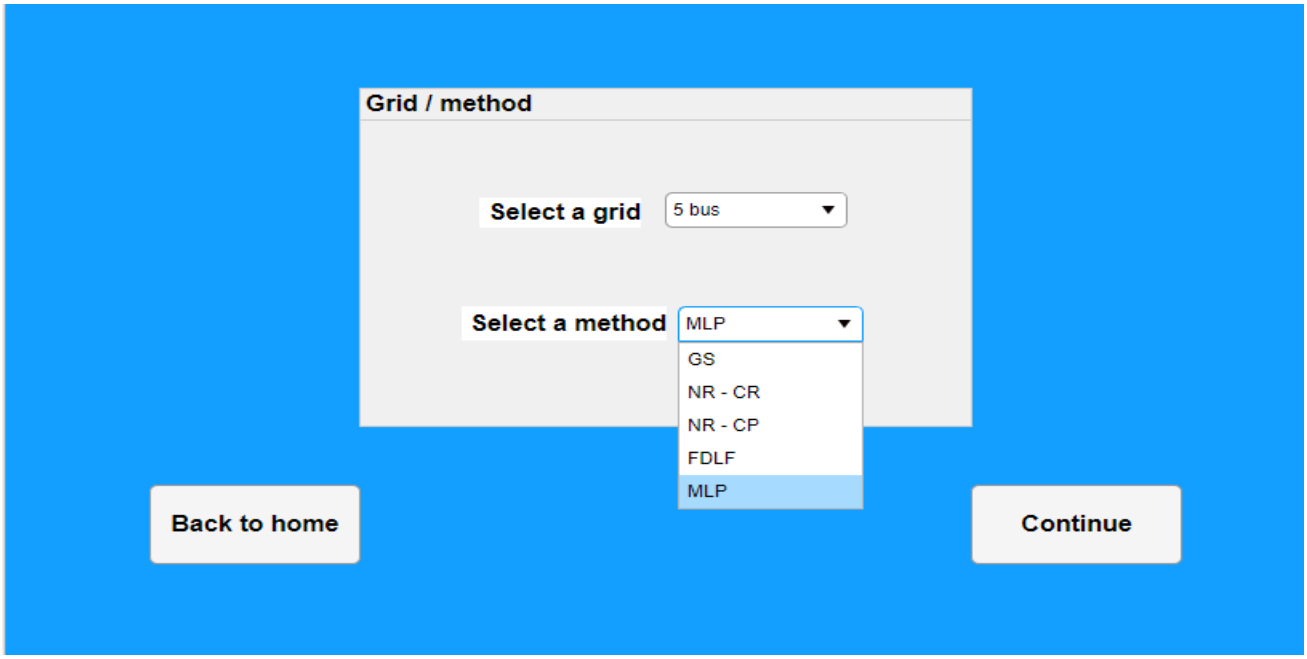


Figure 4.3 : Choix de la méthode d'application

Si on choisit, par exemple, les données du réseau 5 nœuds et que nous voulons effectuer un calcul par la méthode du perceptron multicouche, nous cliquons sur le bouton **Continue.** Une nouvelle fenêtre (figure 4.4) s'affiche et nous permet d'introduire les hyper – paramètres ainsi que visualiser la courbe d'évolution de la fonction coût durant les phases d'apprentissage ou de test. Dans le cas où nous cliquons sur le bouton **Back to home**, on revient vers l'affichage de départ de l'application.

4.3 Choix d'un réseau 5 nœuds

- **Méthode de Perceptron Multicouche**

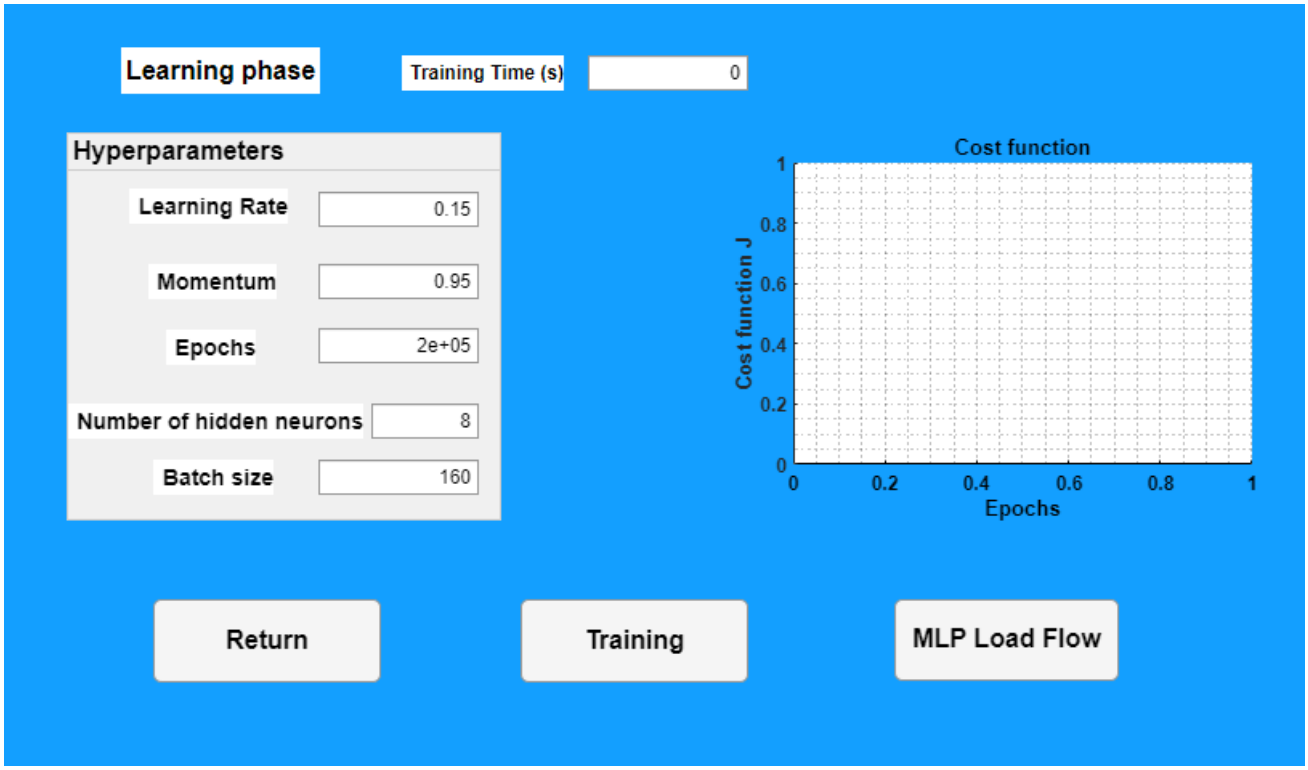


Figure 4.4 : Fenêtre indiquant la phase d'apprentissage ainsi que l'évolution de la fonction coût

Dans cette fenêtre, nous observons que les valeurs des hyper-paramètres existent déjà par défaut (valeurs optimales), mais nous pouvons les modifier à notre aise. Cliquer sur le bouton **Training** lance la phase d'apprentissage et de test du réseau de neurones selon les paramètres introduits. Une fois l'apprentissage et le test terminés le temps de calcul s'affiche et un tracé de la fonction coût apparait (figure 4.5), donnant des informations sur la qualité d'apprentissage afin d'estimer les performances du PMC construit.

Le bouton **Return** retourne vers la fenêtre des choix pour recommencer à nouveau.

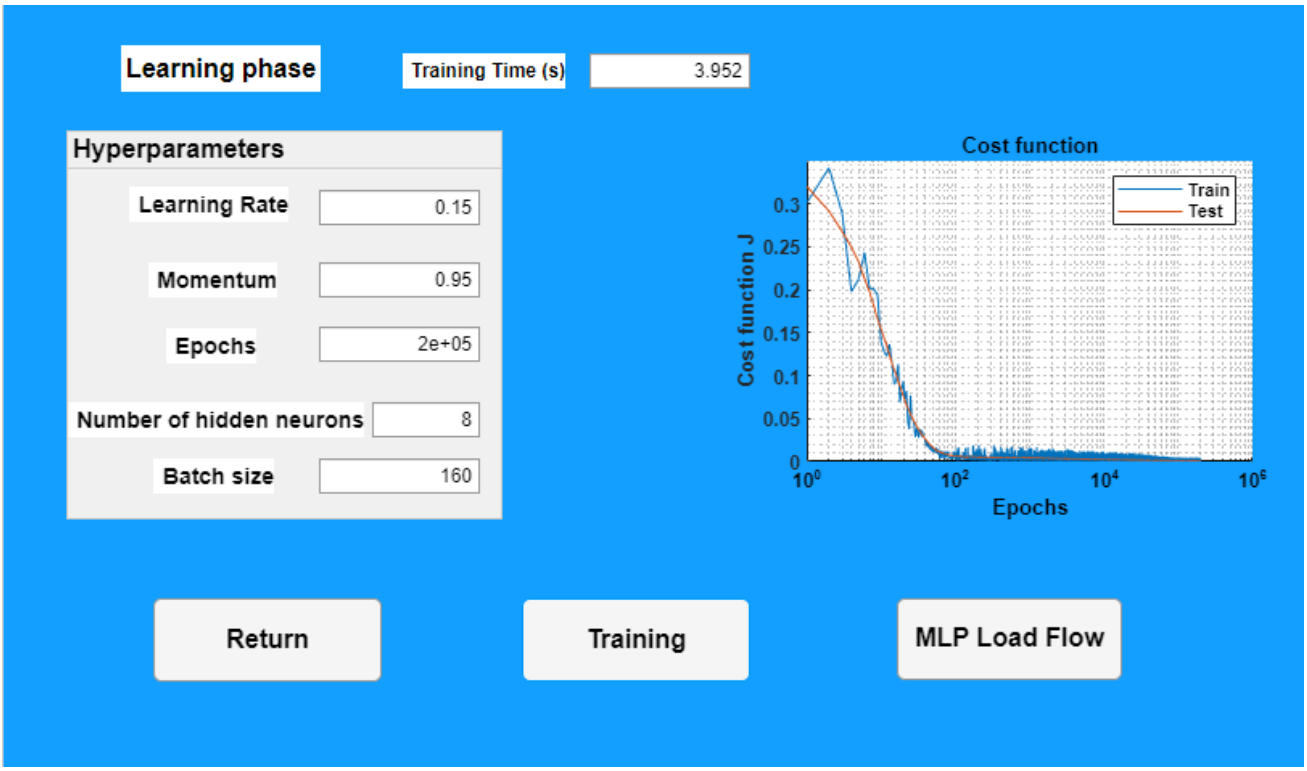
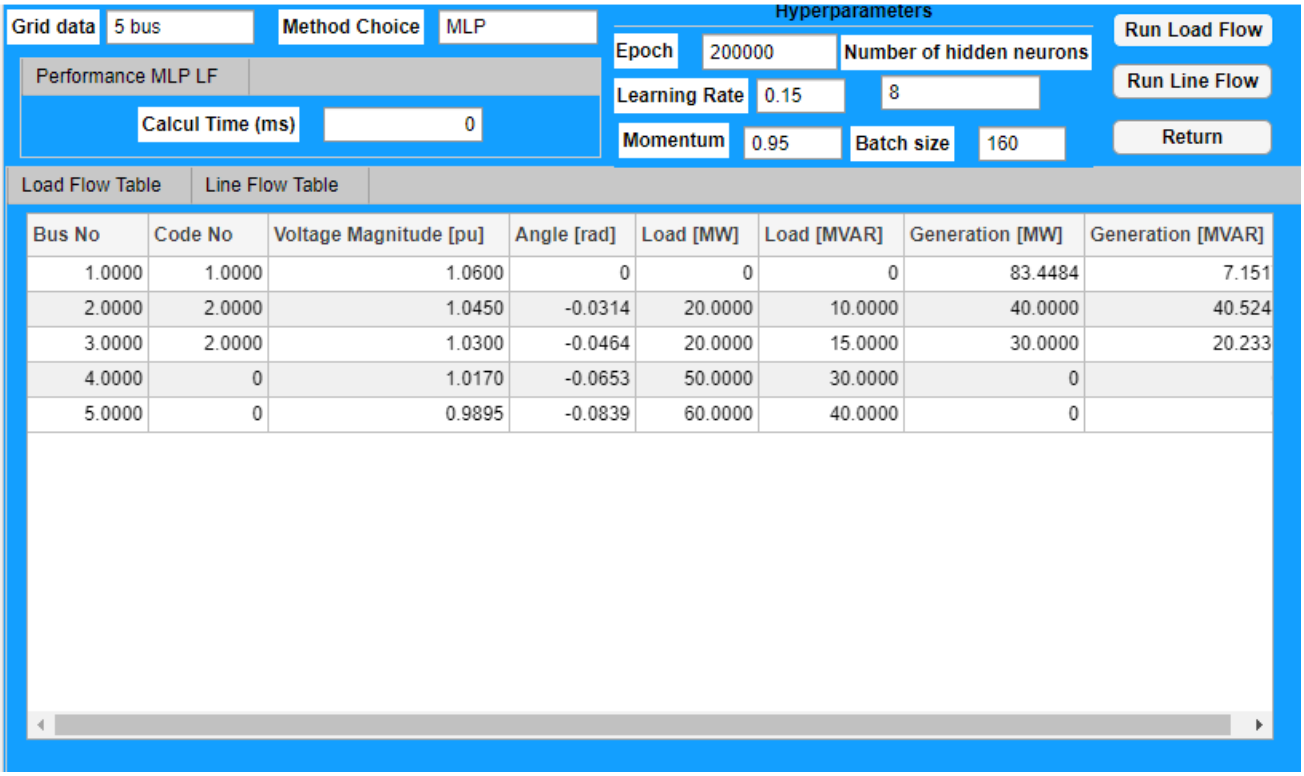


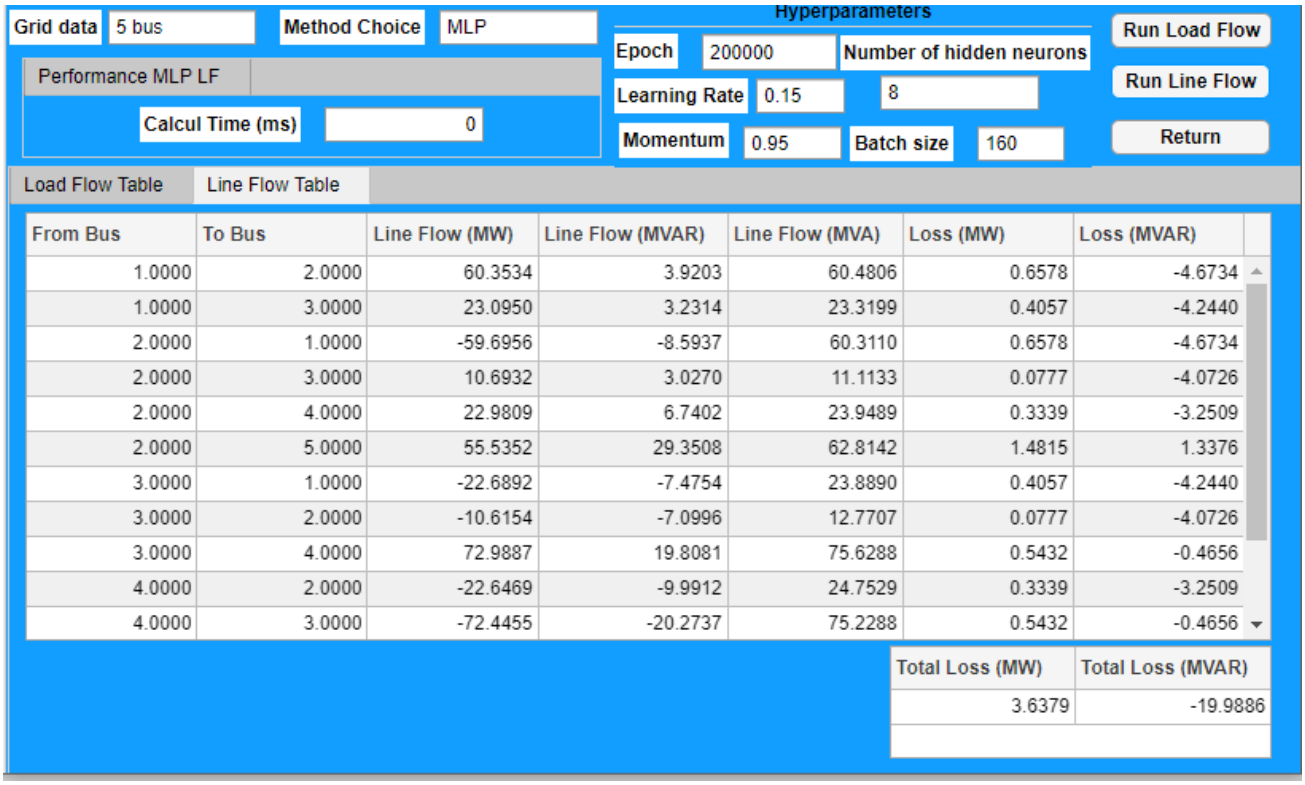
Figure 4.5 : Informations sur le modèle PMC après apprentissage

Une fois le modèle bien entrainé, et une fois les valeurs finales des pondérations et des biais obtenus, nous cliquons sur le bouton **MLP Load Flow** qui nous envoie vers une nouvelle fenêtre. Pour calculer l'écoulement de puissance par propagation au sens de prédiction des nouvelles tensions en modules et angles et les puissances actives et réactives de chaque nœud du réseau électrique, nous cliquons sur le bouton **Run Load Flow**. Le calcul des puissances de transit devient très simple en appuyant enfin sur **Run Line Flow**.



Bus No	Code No	Voltage Magnitude [pu]	Angle [rad]	Load [MW]	Load [MVAR]	Generation [MW]	Generation [MVAR]
1.0000	1.0000	1.0600	0	0	0	83.4484	7.151
2.0000	2.0000	1.0450	-0.0314	20.0000	10.0000	40.0000	40.524
3.0000	2.0000	1.0300	-0.0464	20.0000	15.0000	30.0000	20.233
4.0000	0	1.0170	-0.0653	50.0000	30.0000	0	
5.0000	0	0.9895	-0.0839	60.0000	40.0000	0	

Figure 4.6 : Ecoulement de puissance par PMC



From Bus	To Bus	Line Flow (MW)	Line Flow (MVAR)	Line Flow (MVA)	Loss (MW)	Loss (MVAR)
1.0000	2.0000	60.3534	3.9203	60.4806	0.6578	-4.6734
1.0000	3.0000	23.0950	3.2314	23.3199	0.4057	-4.2440
2.0000	1.0000	-59.6956	-8.5937	60.3110	0.6578	-4.6734
2.0000	3.0000	10.6932	3.0270	11.1133	0.0777	-4.0726
2.0000	4.0000	22.9809	6.7402	23.9489	0.3339	-3.2509
2.0000	5.0000	55.5352	29.3508	62.8142	1.4815	1.3376
3.0000	1.0000	-22.6892	-7.4754	23.8890	0.4057	-4.2440
3.0000	2.0000	-10.6154	-7.0996	12.7707	0.0777	-4.0726
3.0000	4.0000	72.9887	19.8081	75.6288	0.5432	-0.4656
4.0000	2.0000	-22.6469	-9.9912	24.7529	0.3339	-3.2509
4.0000	3.0000	-72.4455	-20.2737	75.2288	0.5432	-0.4656

Total Loss (MW)	Total Loss (MVAR)
3.6379	-19.9886

Figure 4.7 : Calcul de puissance de transit par PMC

- **Méthode de Gauss-Seidel**

Des informations supplémentaires sont ajoutées comme un affichage pour assurer une bonne lisibilité d'exécution de programme à partir de ses caractéristiques ; les paramètres, le système électrique et la méthode utilisée, ainsi que le temps de calcul pour l'exécution réalisée.

Le bouton **Return** (figure 4.7), donne à l'utilisateur la main pour changer de méthode de calcul ainsi que le choix du système à étudier.

Prenons l'exemple d'un réseau de 5 nœuds mais cette fois ci par la méthode GS.

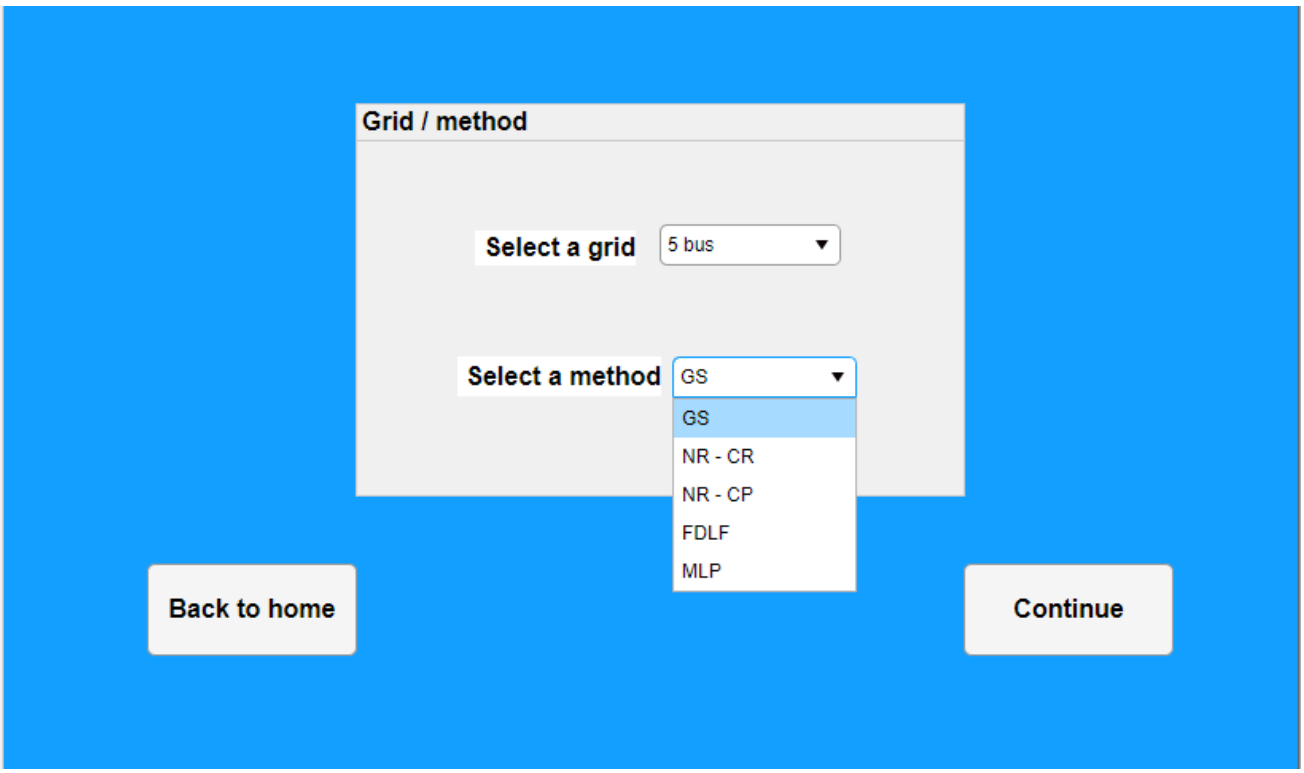
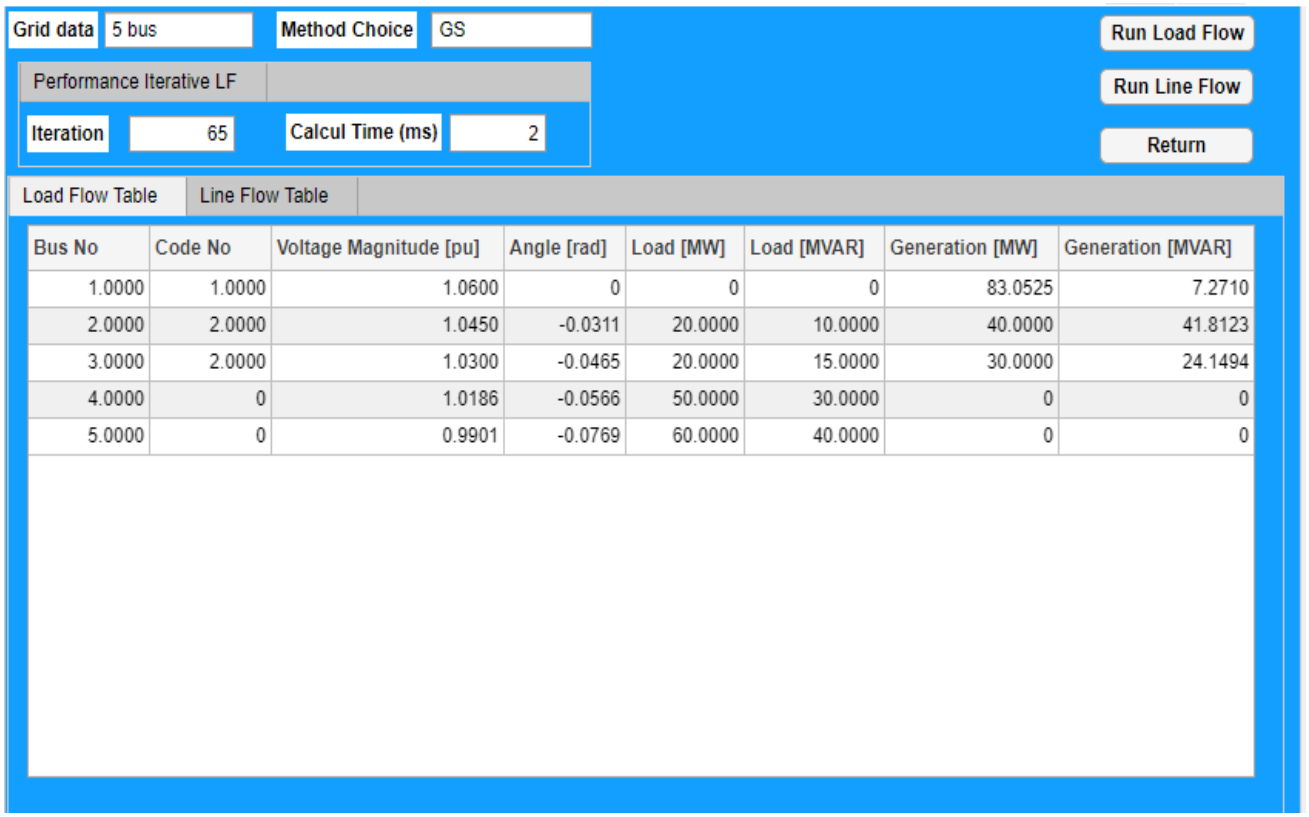


Figure 4.8 : Choix de la méthode GS

L'affichage des tableaux d'écoulement de puissance ainsi que de la puissance de transit sont illustrés comme suit :

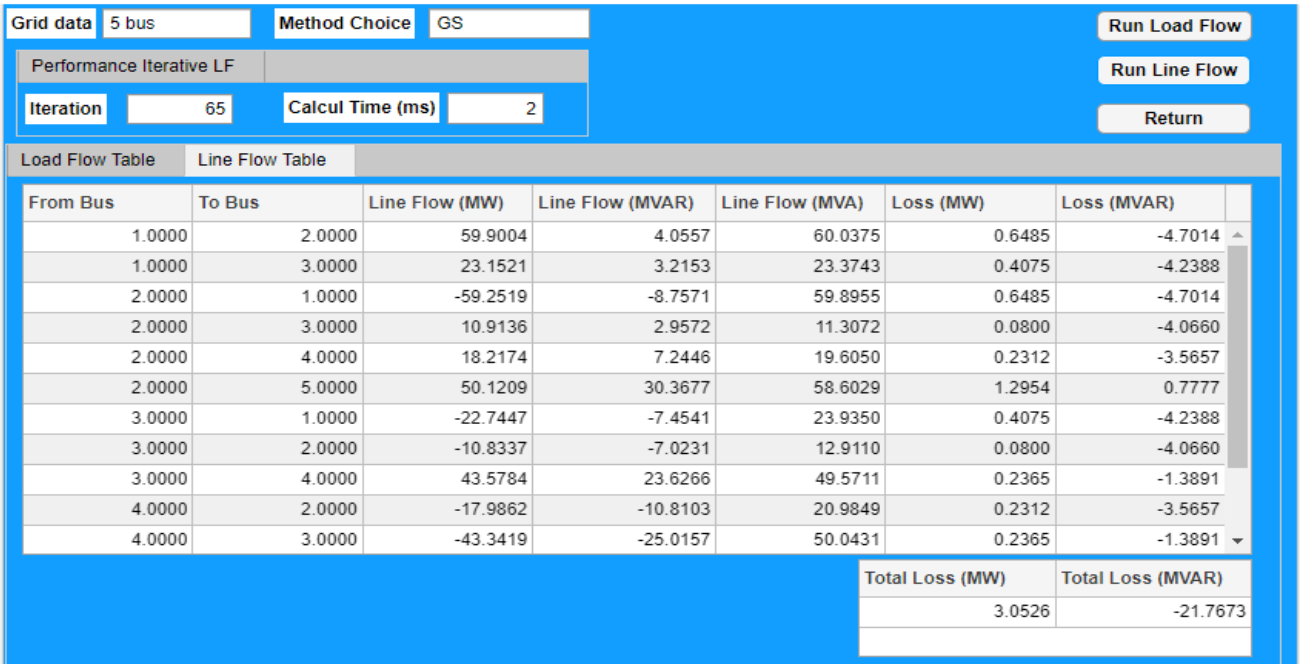
Run Load Flow



Bus No	Code No	Voltage Magnitude [pu]	Angle [rad]	Load [MW]	Load [MVAR]	Generation [MW]	Generation [MVAR]
1.0000	1.0000	1.0600	0	0	0	83.0525	7.2710
2.0000	2.0000	1.0450	-0.0311	20.0000	10.0000	40.0000	41.8123
3.0000	2.0000	1.0300	-0.0465	20.0000	15.0000	30.0000	24.1494
4.0000	0	1.0186	-0.0566	50.0000	30.0000	0	0
5.0000	0	0.9901	-0.0769	60.0000	40.0000	0	0

Figure 4.9 : Ecoulement de puissance par GS

Run Line Flow



From Bus	To Bus	Line Flow (MW)	Line Flow (MVAR)	Line Flow (MVA)	Loss (MW)	Loss (MVAR)
1.0000	2.0000	59.9004	4.0557	60.0375	0.6485	-4.7014
1.0000	3.0000	23.1521	3.2153	23.3743	0.4075	-4.2388
2.0000	1.0000	-59.2519	-8.7571	59.8955	0.6485	-4.7014
2.0000	3.0000	10.9136	2.9572	11.3072	0.0800	-4.0660
2.0000	4.0000	18.2174	7.2446	19.6050	0.2312	-3.5657
2.0000	5.0000	50.1209	30.3677	58.6029	1.2954	0.7777
3.0000	1.0000	-22.7447	-7.4541	23.9350	0.4075	-4.2388
3.0000	2.0000	-10.8337	-7.0231	12.9110	0.0800	-4.0660
3.0000	4.0000	43.5784	23.6266	49.5711	0.2365	-1.3891
4.0000	2.0000	-17.9862	-10.8103	20.9849	0.2312	-3.5657
4.0000	3.0000	-43.3419	-25.0157	50.0431	0.2365	-1.3891

Total Loss (MW)	Total Loss (MVAR)
3.0526	-21.7673

Figure 4.10 : Puissance de transit par GS

La case **Iteration** nous donner le nombre d'itérations nécessaire pour atteindre la convergence. Nous avons aussi (Calcul Time (ms)) le temps de calcul, qui varie d'une machine à une autre en fonction du Hardware. Le principe est le même pour les autres moyens de calcul en conventionnel.

Conclusion générale

Dans ce mémoire, nous avons présenté l'application des méthodes conventionnelles itératives par Gauss–Seidel, Newton-Raphson (en coordonnées rectangulaires et en coordonnées polaires), méthode découplée rapide (FDLF), ainsi que par les réseaux de neurones artificiels en utilisant le perceptron multicouche à une couche cachée, au calcul d'écoulement de puissance des réseaux électriques. Cette étude nous a permis aussi de faire une étude comparative des méthodes mentionnées en terme de convergence et de temps de calcul.

Dans une première partie, notre travail a consisté à la compréhension de la démarche algorithmique des méthodes conventionnelles d'écoulement de puissance et de les programmer sur l'environnement MATLAB. Les programmes conçus ont été testés et validés sur une multitude de réseaux standards (IEEE 14 bus, IEEE 30 bus, IEEE 57 bus, etc.). Ce pendant dans ce mémoire nous nous sommes limités à ne présenter que le cas des réseaux IEEE 05 et 14 nœuds afin de les comparer avec la méthode ultérieurement proposée. Les résultats obtenus par les différentes méthodes conventionnelles nous ont montré que les équations non linéaires de ces méthodes donnent la même solution de l'écoulement de puissance. Les informations calculées pour chaque nœud en amplitude de tension, angle, puissance active et réactive coïncident parfaitement pour chacune des algorithmes itératifs. Il en est de même pour le calcul des puissances de transit des lignes et les pertes totales des mêmes réseaux électriques étudiés. Les performances de chacune de ces méthodes se présentent en nombre d'itérations et en temps de calcul, chaque méthode ayant ses propriétés de convergence, et nous mènent à conclure que la méthode de Newton-Raphson est la plus performante, rapidement convergente, d'ailleurs, d'autant plus que c'est la méthode employée en pratique dans les centres de conduite des compagnies d'électricité.

Nous avons présenté ensuite une technique d'intelligence artificielle basée sur les réseaux de neurones artificiels, soit le perceptron multicouche avec une seule couche cachée, avec application pour le cas de traitement des problèmes de régression. La bonne initialisation des grandeurs d'apprentissage, le choix d'un bon algorithme d'entraînement et l'emploi des hyper-paramètres optimaux influencent fortement sur la convergence du modèle construit. Afin de prouver cette approche, un exemple d'application (régression) a été traité par le réseau neuronal où nous avons abouti à conclure que les résultats de prédiction trouvés par le PMC se superposent avec le modèle analytique imposé sans connaitre la formulation mathématique, ce qui signifie simplement que l'architecture proposée est bonne pour modéliser une telle fonction analytique étudiée quel que soit sa complexité et son ordre de non linéarité.

La troisième partie du mémoire est la partie la plus importante puisqu'elle se focalise sur l'application du PMC au calcul d'écoulement de puissance pour un réseau électrique. Une architecture du réseau neuronal a été choisie selon les propriétés du réseau électrique et un algorithme d'entraînement est établi par descente du gradient avec rétro – propagation des erreurs depuis une base de données de différents points de fonctionnement existants déjà. Le PMC a été testé et validé sur les réseaux de 5 nœuds et 14 nœuds (IEEE 14 bus). Les résultats obtenus pour chacun de ces réseaux sont très proches en comparaison à ceux trouvés avec les méthodes conventionnelles, particulièrement la méthode de Newton – Raphson. Le temps de calcul par la méthode intelligente

est très court avec une seule itération comparativement aux méthodes itératives qui nécessitent quelques itérations pour atteindre la convergence. Les mêmes remarques sont valables pour le calcul des puissances de transit des lignes, ainsi que les pertes actives et réactives. Le PMC semble être plus rapide et plus efficace, avec la capacité d'apprendre et traiter le problème de l'écoulement de puissance en évitant tous les problèmes de la complexité des méthodes conventionnelles et le risque de se bloquer dans des minimums locaux ou carrément de diverger. Cette méthode intelligente pourrait devenir une autre alternative à utiliser en pratique, en la développant pour des réseaux de plus grande dimension.

En dernière partie, une présentation guidée a été faite pour notre programme que nous avons dénommé '**ILFC**', développé sous le gestionnaire d'interface graphique **APPDESIGNER de Matlab.** Notre programme est un outil de calcul, avec une interface conviviale de communication avec l'utilisateur. Il comprend une petite base de réseaux électriques, qui peut être agrandie en insérant d'autres modèles de réseaux électriques, avec la possibilité de choisir une des différentes techniques de résolution abordées dans ce mémoire, et visualise tous les résultats obtenus sous forme de tableaux faciles à lire afin de pouvoir les interpréter d'une manière plus simple, flexible et dynamique.

Bibliographie

[1] H. H. Müller, M. J. Rider “Power Flow Model Based on Artificial Neural Networks”, School of Electrical and Computer Engineering, University of Campinas (UNICAMP), Campinas, SP, Brazil.

[2] H. Kubba, « Assessment and Comparative Study of Different Enhanced Artificial Neural Networks Based Power Flow Solutions », *International Journal of Engineering Research*, vol. 3, n° 3, p. 17, 2014.

[3] M. Benhasna, M. Fahem, “Etude de l’écoulement de puissance sécuritaire d’un réseau de transport électrique”, Mémoire en fin d’étude, Master académique en électrotechnique industrielle, Université Abdelhamid Ibn Badis, Mostaganem Algérie, juin 2018.

[4] B. Stott, “Review of Load – Flow Calculation Methods”, Proceedings of the IEEE, Vol. 62, NO. 7, July 1974, pp. 916-929.

[5] B. S. Hota, A. K. Mallick, “Load Flow Study In Power System”, Thesis for the degree of bachelor of technology, Departement of Electrical Engineering, National Institute of Technology, Rourkela, India, 2011.

[6] A. Bouzidi, “ Cours de Modélisation et simulation des réseaux électriques”, Département de génie électrique, Université A/Mira, Béjaia, Algérie.

[7] URL : https://www.elia.be/fr/donnees-de-reseau/charge-et-prevision-de-charge.

[8] A.Hellal, “Cours d’Analyse des Réseaux Electrique”, Département d’Electrotechnique, Ecole Nationale Polytechnique, Alger, Algérie, 2020.

[9] N. R. Bokka, “Comparaison of Power Flow Algorithms for inclusion in On – line Power System Operation Tools”, Thesis for the degree of master of science in Engineering, University of New Orleans, U.S, 2010.

[10] Y. Wallach, R. K. Even, “Application of Newton’s method to load – flow calculations”, PROC. IEE, Vol. 114, No. 3, March 1967, pp. 372-374.

[11] A. Hellal, “ Cours de Programmation Mathématique ”, Département d’Electrotechnique, Ecole Nationale Polytechnique, Alger, Algérie, 2016.

[12] S. Laqrichi, “Approche pour la construction de modèles d’estimation réaliste de l’effort/coût de projet dans un environnement incertain : application au domaine du développement logiciel ”, Thèse de doctorat en génie industrielle, Université Fédérale de Toulouse Midi – Pyrénées, France, 17 décembre 2015.

[13] URL: https://machinelearningmastery.com/weight-initialization-for-deep-learning-neural-networks/ .

[14] Philippe Giguère cours apprentissage par réseaux de neurones profonds « optimisation pour l’apprentissage profond », Université Laval, Québec, Canada, 2020.

[15] S.A Zainaddin, « A Neural Network Approach for Load Flow Estimation and VAR Allocation », Faculty of the College of Graduate Studies king Fahd University of Petroleum & Minerals Dhahran, Saudi Arabia.

Annexe

Données du réseau 5 nœuds (point de fonctionnement basique) :

- **Données des nœuds :**

```
%                              ----  Bus data  ----
%        Bus Bus  Voltage Angle   ---Load---- -------Generator----- Static Mvar
%        No  code Mag.    Degree  MW    Mvar  MW  Mvar Qmin Qmax    +Qc/-Ql
busdata=[1   1    1.06    0.0      0     0     0   0    10   50       0
         2   2    1.045   0.0     20    10    40  30    10   50       0
         3   2    1.03    0.0     20    15    30  10    10   40       0
         4   0    1.00    0.0     50    30     0   0     0    0       0
         5   0    1.00    0.0     60    40     0   0     0    0       0];
```

- **Données de lignes :**

```
%                                        Line data
%          Bus bus   R      X     1/2 B   = 1 for lines
%          nl  nr  p.u.   p.u.   p.u.     > 1 or < 1 tr. tap at bus nl
linedata=[1   2   0.02   0.06   0.030      1
          1   3   0.08   0.24   0.025      1
          2   3   0.06   0.18   0.020      1
          2   4   0.06   0.18   0.020      1
          2   5   0.04   0.12   0.015      1
          3   4   0.01   0.03   0.010      1
          4   5   0.08   0.24   0.025      1];
```

Données du réseau 14 nœuds (IEEE 14 ; point de fonctionnement basique) :

- **Données des nœuds :**

```
%                                Bus data or Node data
%                                ---------------------
%                                   -- Load --    -- Gen --                  Injected
%       Bus Bus Voltage  Angle      Pl     Ql     Pg     Qg     Qmin   Qmax   +Qc/-Ql
%       No code  Mag.    Degree     MW     Mvar   MW     Mvar   Mvar   Mvar    Mvar
busdata=[1    1    1.06     0.0      0.0    0       0      0      0      0       0
         2    2    1.045    0.0     21.7   12.7    40      0    -40     50       0
         3    2    1.010    0.0     94.2   19.0     0      0     40      0       0
         4    0    1.0      0.0     47.8   -3.9     0      0      0      0       0
         5    0    1.0      0.0      7.6    1.6     0      0      0      0       0
         6    2    1.070    0.0     11.2    7.5     0      0     -6     24       0
         7    0    1.0      0.0      0      0       0      0      0      0       0
         8    2    1.090    0.0      0      0       0      0     -6     24       0
         9    0    1.0      0.0     29.5   16.6     0      0      0      0      19
        10    0    1.0      0.0      9.0    5.8     0      0      0      0       0
        11    0    1.0      0.0      3.5    1.8     0      0      0      0       0
        12    0    1.0      0.0      6.1    1.6     0      0      0      0       0
        13    0    1.0      0.0     13.5    5.8     0      0      0      0       0
        14    0    1.0      0.0     14.9    5.0     0      0      0      0       0];
```

- **Données des lignes :**

```
%                    Line data or Branch data
%                                                Line code
%        Bus bus     R         X       B/2     = 1 for lines
%        nl  nr    p.u.      p.u.      p.u.   > 1 or < 1 tr. tap at bus nl
linedata=[1   2   0.01938   0.05917   0.0264       1
          1   5   0.05403   0.22304   0.0246       1
          2   3   0.04699   0.19797   0.0219       1
          2   4   0.05811   0.17632   0.0187       1
          2   5   0.05695   0.17388   0.0170       1
          3   4   0.06701   0.17103   0.0173       1
          4   5   0.01335   0.04211   0.0064       1
          4   7   0.0       0.20912   0.0          0.932
          4   9   0.0       0.55618   0.0          0.978
          5   6   0.0       0.25202   0.0          1
          6  11   0.09498   0.19890   0.0          0.969
          6  12   0.12291   0.25581   0.0          1
          6  13   0.06615   0.13027   0.0          1
          7   8   0.0       0.17615   0.0          1
          7   9   0.0       0.11001   0.0          1
          9  10   0.03181   0.08450   0.0          1
          9  14   0.12711   0.27038   0.0          1
         10  11   0.08205   0.19207   0.0          1
         12  13   0.22092   0.19988   0.0          1
         13  14   0.17093   0.34802   0.0          1];
```