République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique
Département d'Electrotechnique
Laboratoire de Recherche en Electrotechnique

Mémoire de

Master en Electrotechnique

Présenté par
ZEKKOUR Fateh

Intitulé

Méthode d'Essaim de Particules pour le calcul d'écoulement de puissance optimale

Soutenu le 17 juin 2015 à l'ENP

Membres du Jury d'examen

Rapporteur	A.Hellal	Professeur à l'ENP
Président	T. Zebbadji	Professeur à l'ENP
Examinateurs	R.Ibtiouen	Professeur à l'ENP
	S.Mekhtoub	Professeur à l'ENP

ENP 2015

ملخص

يهدف هذا العمل الى دراسة السريان الأمثل للطاقة الكهربائية بطرق مختلفة حيث نسعى لتخفيض تكلفة انتاج الكهرباء و أيضا للحد من الخسائر أثناء توزيع الكهرباء على الشبكة. نحسب هذا السريان الأمثل بواسطة طريقة ذكاء اصطناعي حيث نقارن ونناقش النتائج المتحصل عليها وكذا العوامل المؤثرة سلبا وإيجابا على السير الأمثل للطاقة الكهربائية .

الكلمات المفتاحية : السريان الأمثل للطاقة الكهربائية ، طريقة ذكاء اصطناعي .

Résumé

Ce travail traite le problème d'écoulement de puissance optimal en général et particulièrement le cas de la répartition économique de puissance. Pour cela, une méthodes d'intelligence artificiel (PSO) est utilisée pour resoudre le problème. Des simulations sur un réseau standard ont été exécutées et les résultats comparés et analysés.

Mots clés : écoulement de puissance optimal, répartitions économique de puissance , méthodes d'intelligence artificiel

Abstract

This work deals with the problem of opf in general and discuss the case of distributions of economical dispatch. We solve this problem with an artificial inteligence method wich is partical swarm optimization . Finally, we represent the different results obtained.

Key words : opf, economical dispatch, artificial inteligence methods.

Remerciements

En premier lieu, je remercie le bon Dieu miséricordieux et clément, qui m'a guidés dans la bonne voie des sciences et de la connaissance.

Je tiens à remercier mon promoteur **Pr : A.Hellal** qui m'a fait profiter de son expérience et ses précieux conseils dans ce travail et dans le domaine de la recherche scientifique. Puisse ce travail vous satisfaire et témoigner ma grande reconnaissance et ma profonde estime.

Je suis très sensibles à l'honneur que me fait **Pr : T.Zebbadji** en présidant ce jury. Je tiens à lui exprimer mes remerciements les plus sincères.

j'exprime ma profonde reconnaissance au **Pr :R.Ibtiouen** et au **Pr : S.Mekhtoub** d'avoir accepté d'être membres du jury et de juger ce modeste travail.

Je tiens aussi à remercier mon frère et ami **Islam Ziouani** pour l'aide qu'il m'a donner et pour sa patience et sa honnêteté et pour ces qualité humaines.

Je tiens également à remercier tous les enseignants ayant contribué à ma formation depuis le tronc commun jusqu'à la dernière année de graduation.

Table des matières

Abréviations

EPO **E**coulement de **P**uissance **O**ptimal

OEP **O**ptimisation par **E**ssaim de **P**articules

REP **R**épartition **E**conomique de **P**uissance.

Dédicaces

A mes cher parents

A mon cher frère Ahmed et mes cher sœurs Amina et Rania

A toute ma famille

A tout mes amis et collègues

Fateh

Introduction

La gestion d'un réseau de production, distribution ou stockage d'énergie est devenue un enjeu tant économique que technique, devant respecter des contraintes climatiques et économiques. Il apparaît donc la nécessité d'optimiser cette gestion afin de profiter du meilleur fonctionnement du réseau tout en respectant les contraintes imposées.

Le problème de répartition économique, qui est un cas des problème d'écoulement de puissance, dans son traitement à l'état statique et dynamique d'un réseau électrique, occupe dans nos jours une place déterminante dans la stratégie concurrentielle de l'entreprise, face à la libéralisation du secteur d'électricité, donc face à une concurrence acharnée, mais aussi par rapport aux nouvelles restrictions liées à l'environnement qu'il faut respecter.

Dans cette logique, un faible coût de production représente un défi pour les sociétés productrices, vu notamment l e prix des combustibles, et toutes les charges supplémentaires liées à la production , aux quelles on peut ajouter des frais comme celles liées au traitement des déchets nucléaires dans le cas de la production nucléaire.

D'un autre coté, la complexité grandissante des réseaux électriques d'aujourd'hui, vu des tailles avec des centaines de jeux de barres et des milliers de kilomètres de lignes de transmission, ainsi que des structures interconnectées, exige qu'une optimisation de la répartition optimale de puissance active générée constitue une nécessité impérative et un coût minimisé représente un objectif primordial.

Notons qu'une optimisation de cette répartition ne doit pas seulement garantir un faible coût de production mais aussi s'accompagner d'une minimisation des pertes de transport (répartition économique avec pertes), ce qui engendre un problème d'optimisation non linéaire.

Plusieurs méthodes sont proposées pour la résolution du problème d'écoulement de puissance optimal dont des méthodes conventionnelles (méthode de Newton, méthode du gradient, méthode des itérations lambda, programmation linéaire ...etc) et d'autres non conventionnelles (méthodes d'intelligence artificielle) comme les algorithmes génétiques (AG), les algorithmes évolutionnaires, optimisation par essaim de particules (OEP ou en anglais PSO) ..etc

Dans ce mémoire, nous allons appliquer un algorithme d'optimisation par essaim de particules avec apprentissage adaptatif (ALPSO) pour la résolution du problème de répartition économique.

Le mémoire est organisé comme suit :

— le premier chapitre intitulé *Formulation et méthodes de résolution du problème de l'EPO* donne des définitions de base sur la notion de l'écoulement de puissance optimale, la formulation du problème d'EPO, le dispatching économique et ces différents aspects.

— le deuxième chapitre intitulé *Méthode d'Optimisation par Essaim de Particules à la REP* décrit l'algorithme d'ALPSO que nous avons utiliser pour la résolution du problème et une analyse des résultats obtenus.

Chapitre 1

Formulation et méthodes de résolution du problème de l'écoulement de puissance optimal

1.1 Formulation du problème d'écoulement de puissance optimal

En général, l'OPF comprend tout problème d'optimisation qui cherche à optimiser le fonctionnement d'un système de réseau électrique (en particulier, la génération et la transmission de l'électricité), sous réserve de contraintes imposées par les lois électriques et les limites techniques sur les variables de contrôle. Ce cadre général englobe divers problèmes d'optimisation pour la planification et le fonctionnement des réseaux électriques. Il agit sur les variables de contrôle disponibles afin d'optimiser un objectif tout en satisfaisant les équations d'écoulement de puissance de réseau, les contraintes physiques et opérationnelles. L'écoulement de puissance optimal a été présenté au début des années 60 [1], et même avant par d'autres auteurs. Depuis ce temps, l'EPO est devenu un outil indispensable pour le développement, et la commande en temps réel des réseaux électriques. L'EPO est en général un problème non linéaire, avec des variables continues et discrètes. La formulation du problème d'EPO peut s'écrire comme suit :

$$minf(x,u) \tag{1.1}$$

Lié à :

$$g(x,u) = 0 \tag{1.2}$$

$$h(x) \leq 0 \tag{1.3}$$

$$U_{min} \leq U \leq U_{max} \tag{1.4}$$

Avec :

- u : variable de contrôle
- x : variable d'état
- L'équation 1.1 représente la fonction objective.
- L'équation 1.2 représente les contraintes d'égalités.
- L'équation 1.3 représente les contraintes d'inégalités.
- L'équation 1.4 représente les limites des variables de contrôle.

1.2 Signification des éléments de problème

1.2.1 Variables d'état

L'état du système est défini par la matrice admittance du réseau et les tensions aux nœuds.

1.2.2 Variables de contrôle

Les variables de contrôle dans l'EPO en générale sont :

- P_g : puissance active générée
- ϕ : transformateur de phase.
- Q_g : puissance réactive générée
- V : amplitude de tension

1.2.3 Contraintes égalités

Les contraintes d'égalité de l'EPO reflètent l'état du réseau électrique ainsi que la tension souhaitée à chaque nœud du système. L'état du réseau électrique est exprimé par les équations de l'écoulement de puissance qu'ils assurent l'équilibre du système.

1.2.4 Contraintes inégalités

On peut rencontrer deux types de contraintes inégalités :

— **Limites supérieures et inférieures**
Concernent les variables de contrôle car elles sont liées au fonctionnement des générateurs et à la stabilité du réseau.
$Pg_{min} < Pg < Pg_{max}$
$Qg_{min} < Qg < Qg_{max}$

— **Contraintes inégalité fonctionnelle**
On prend par exemple la puissance maximale transitée entre deux nœuds.

1.2.5 Fonction objectif

La fonction objectif de l'EPO peut prendre diverses formes selon l'objectif souhaité. Les objectifs les plus utilisés dans l'EPO sont

— la minimisation du coût de production de la puissance.
— la minimisation des pertes actives.
— la minimisation des pertes réactives
— minimisation de délestage
— maximisation de la limite de capacité de charge
— maximisation du profit
— minimisation des déviations causées par les violations des contraintes ... etc

Parmi les objectifs cités, la minimisation du coût de production de la puissance est l'une des fonctions objectif les plus utilisées pour l'EPO, ce qui a fait l'objet de notre travail dans les sections suivantes.

1.3 Étude de problème de la répartition économique de puissance

Le problème de la répartition économique a eu son début à partir du moment que deux ou plusieurs unités se sont engagés à prendre en charge sur un système de puissance dont la demande total dépassait la génération maximal d'une seule unité. Le problème qui se posait à l'opérateur était exactement comment diviser la génération entre les unités.

La répartition économique alors est fait en temps réel, à cause de la variation du coût et de la puissance demandé pour assigner la génération totale parmi les unités disponibles pour satisfaite la demande. La répartition économique est un processus de calcul selon laquelle la production totale nécessaire est répartie entre les unités de production disponibles de sorte que les contraintes imposées sont remplies et que les besoins en énergie en termes de BTU/h ou $\$/h$ sont minimisés.

1.3.1 Formulation du problème de la REP

Le problème de la répartition économique de puissance peut être formulé de la façon suivante :

$$minf(U = P_g) = min \sum_{i=1}^{N_{gen}} a_i + b_i P_{g_i} + c_i P_{g_i}^2 \tag{1.5}$$

Lié à :

$$\sum_{j \in i}^{N} V_i V_j [G_{ij} cos(\delta_i - \delta j) + B_{ij} sin(\delta_i - \delta j)] - Pg_i + Pd_i = 0 \tag{1.6}$$

$$\sum_{j \in i}^{N} V_i V_j [G_{ij} sin(\delta_i - \delta j) - B_{ij} cos(\delta_i - \delta j)] - Qg_i + Qd_i = 0 \tag{1.7}$$

$$P_{g_{min}} \leq P_g \leq P_{g_{max}} \tag{1.8}$$

tel que :

- N : nombre de nœuds.
- N_{gen} : nombre de nœuds de générations.
- G_{ij} :conductance des lignes.
- B_{ij} : susceptance des lignes.
- $\delta_{i,j}$: déphasage de la tension V_{ij} du nœud i(ou j)
- Pg_i : puissance active générée au nœud i.
- Pd_i : puissance active demandée au nœud i.
- Qg_i : puissance réactive générée au nœud i.
- Qd_i : puissance réactive demandée au nœud i.
- P_g est le variable de contrôle.
- V et θ sont les variables d'état.
- Pg_i est la quantité de la production en MW au générateur i.
- 1.5 représente le coût de production total.
- 1.6 et 1.7 représentent les equations de l'écoulement de puissance.
- 1.8 représente les limites des puissances générées

Note

Généralement, la fonction coût de chaque générateur est de forme quadratique de 2^{eme} ordre.

Le problème de la REP peut être résolu par plusieurs méthodes conventionnelles et d'autres non conventionnelles dites d'intelligence artificielle.

1.3.2 Les méthodes conventionnelles

Les techniques classiques dites aussi mathématiques ou encore conventionnelles appliquées au problème de la répartition économique de puissance, peuvent être classifiées en deux groupes. Le premier représente la famille des méthodes d'optimisation non linéaire (ou programmation non linéaire) [1] qui sont basées sur la théorie du calcul différentiel où le gradient et/ou le Hessien qui sont utilisés pour guider la procédure de recherche afin de localiser la solution optimale. Le deuxième groupe inclut les méthodes de programmation linéaire, qui sont fondées sur les techniques du simplexe et du point intérieur [2]. Le dispatching économique est un problème d'optimisation qui consiste à répartir la production de la

puissance active générée entre les différentes centrales du réseau, de sorte à exploiter ce dernier de la manière la plus économique possible. Cette distribution doit évidemment respecter les limites de production des centrales.

1.3.2.1 Méthode du Gradient

La méthode du Gradient cherche à déterminer la direction de descente qui fait décroitre $f(x + dx)$ le plus vite possible. $\nabla f(X_k)$ indique la direction du taux de décroissement de f au point x_k . Plusieurs façons d'utiliser cette direction de descente existent, dont on peut citer :

— Gradient simple
— Gradient à pas optimal
— Gradient conjugué
— Gradient projeté

1.3.2.2 Méthode de newton

La répartition économique de puissance peut être résolue avec pour but de résoudre le gradient du lagrangien $\nabla L_x = 0$. Puisque c'est une fonction vectorielle, le problème peut être formulé pour trouver le chemin qui conduit le gradient vers zéro. La méthode de Newton est utilisée dans ce sens. La méthode de Newton pour une fonction à plusieurs variables est développée comme suit . Soit une fonction g(x) à minimiser tel que :
Soit une fonction $g(x)$ à minimiser tel que
$g(x + \Delta x) = g(x) + [g'(x)]$ et $\Delta x = 0$.
La construction de la matrice Jacobienne $[G'(x)]$.
l'ajustement en chaque itération se fait suivant :
$\Delta x = -[g'(x)]^{-1} g(x)$. Maintenant si la fonction $g(x)$ est le vecteur de gradient ∇L_x, donc $\nabla x = -\left[\frac{\partial \nabla L_x}{\partial x}\right]^{-1} \nabla L$.
Les éléments de la matrice jacobienne sont les dérivées partielles de second ordre, et la matrice $[G(x)]$ est appelée matrice hessienne.

1.3.2.3 La méthode d'itération de Lambda

La méthode d'itération de Lambda est l'une des méthodes utilisées pour trouver la valeur de Lambda du système et trouver le dispatching économique optimal des générateurs. Contrairement aux autres méthodes d'itération, comme : Gauss-Seidel et Newton-Raphson, la méthode d'itération de Lambda n'utilise pas la valeur précédente de l'inconnue pour trouver la valeur suivante, c'est-à-dire qu'il n'y a pas d'équation qui calcule la valeur suivante en fonction de la valeur précédente. La valeur suivante est prédéfinie par intuition, elle est projetée avec interpolation de la bonne valeur possible jusqu'à ce que le décalage spécifié soit obtenu. On va maintenant discuter comment trouver le dispatching économique optimal utilisant cette dernière.

— La méthode exige qu'il y ait une correspondance entre une valeur *lambda* et l'output (en MW) de chaque générateur.

— — La méthode commence avec des valeurs de lambda en dessous et en-dessus de la valeur optimale (qui est inconnue), puis par itération limite la valeur optimale.

1.3.2.4 Méthode du point intérieur

Les méthodes de point intérieur fonctionnent comme suit : à partir d'une valeur initiale du paramètre de perturbation de la condition de complémentarité du système μ, strictement positif, et d'un point initial x_0 strictement réalisable, on résout de manière approchée le problème barrière. On calcule ensuite un nouveau paramètre de perturbation μ strictement positif et inférieur au précédent. On obtient alors un nouveau problème barrière à résoudre. On le résout de manière approchée à partir de la solution approchée du problème barrière précédent et ainsi de suite. On va donc résoudre, de manière approchée, une suite de problèmes barrières à μ fixé, la suite des paramètres de perturbation tendant vers 0, jusqu'à l'obtention d'une solution du problème initial.

1.3.3 Les méthodes d'intelligence artificielle

1.3.3.1 Algorithmes génétiques

Les algorithmes génétiques (AG) développés par J. Holland présentent des qualités intéressantes pour la résolution de problèmes d'optimisation complexes. Ils tentent de simuler le processus d'évolution des espèces dans leur milieu naturel : soit une transposition artificielle de concepts basiques de la génétique et des lois de survie énoncés par Darwin. Et aussi de la théorie de décision du mathématicien Hadamard du siècle passé.

Rappelons que la génétique représente un individu par un code, c'est-à-dire un ensemble de données (appelées chromosomes), identifiant complètement l'individu. La reproduction est dans ce domaine un mixage aléatoire de chromosomes de deux individus, donnant naissance à des individus enfants ayant une empreinte génétique nouvelle, héritée des parents. La mutation génétique est caractérisée dans le code génétique de l'enfant par l'apparition d'un chromosome nouveau, inexistant chez les individus parents.

Ce phénomène génétique d'apparition de " mutants" est rare mais permet d'expliquer les changements dans la morphologie des espèces, toujours dans le sens d'une meilleure adaptation au milieu naturel.

La disparition de certaines espèces est expliquée par " les lois de survie " selon lesquelles seuls les individus les mieux adaptés auront une longévité suffisante pour générer une descendance. Les individus peu adaptés auront une tendance à disparaître.

C'est une sélection naturelle qui conduit de génération en génération à une population composée d'individus de plus en plus adaptés. Un algorithme génétique est construit de manière tout à fait analogue.

Dans l'ensemble des solutions d'un problème d'optimisation, une population est constituée de solutions convenablement marquées par un codage qui les identifie complètement. Une procédure d'évaluation est nécessaire à la détermination de la force de chaque individu de la population. Viennent ensuite une phase de sélection (en sélectionnant les individus suivant de leur force) et une phase de recombinaison (opérateurs artificiels de croisement et de mutation) qui génèrent une nouvelle population d'individus, qui ont de bonnes chances d'être plus forts que ceux de

la génération précédente. De génération en génération, la force des individus de la population augmente et après un certain nombre d'itérations, la population est entièrement constituée d'individus tous forts, soit de solutions quasi-optimales du problème posé [3].

1.3.3.2 Algorithme à évolution différentielle

Classée parmi les méthodes méta-heuristiques stochastiques d'optimisation, l'algorithme à évolution différentielle (DEA) [4] est une technique relativement récente, conçue pour optimiser des problèmes sur les domaines continus. Cet algorithme est inspiré des algorithmes génétiques et les stratégies évolutionnistes combinées avec une technique géométrique de recherche. Les algorithmes génétiques changent la structure des individus en utilisant la mutation et le croisement, alors que les stratégies évolutionnistes réalisent l'auto adaptation par une manipulation géométrique des individus.

Dans cette approche, chaque variable de décision est représentée dans le chromosome (ou individu) par un nombre réel. Comme dans tout algorithme évolutionnaire, la population initiale du DEA est générée aléatoirement, puis évaluée. Après cela, le processus de sélection prend place. Au cours de la phase de sélection, trois parents sont choisis et ils génèrent un seul enfant (ou descendant) qui est en concurrence avec un parent pour déterminer lequel subsistera à la génération suivante. Le DEA génère un seul enfant (au lieu de deux comme dans les algorithmes génétiques) en ajoutant le vecteur de différence pondérée entre deux parents à un troisième parent. Si le vecteur résultant donne une plus faible valeur de la fonction objectif que celle donnée par un membre prédéterminé de la population, le vecteur nouvellement généré remplace le vecteur auquel il a été comparé.

1.3.3.3 Optimisation par essaim de particules

L'optimisation par essaim de particules est une technique évolutionnaire qui utilise une population de solutions candidates pour développer une solution optimale au problème. Le degré d'optimalité est mesuré par une fonction fitness [5]. Il est inspiré du comportement collectif et de l'intelligence émergente qui existent dans les sociétés à population organisée. Les membres de la population, particules, sont dispersés dans l'espace du problème, ainsi que le comportement de l'essaim peut

être décrit en se plaçant du point de vue d'une particule. Au départ de l'algorithme, un essaim est réparti au hasard dans l'espace de recherche, chaque particule ayant également une vitesse aléatoire. Chaque particule est représentée par une position et une vitesse, qui sont mis à jour comme suit :

$$X_i^{k+1} = V_i^{k+1} + X_i^k \tag{1.9}$$

$$V_i^{k+1} = \omega V_i^k + c_1 rand_1(Pbest_i^k - X_i^k) + c_2 rand_2(Gbest^k - X_i^k) \tag{1.10}$$

Remarque

Notons que les générateurs à combustibles distincts possèdent différents coûts pour fournir le même montant d'énergie électrique. C'est important de se rendre compte que le générateur le plus rentable du système ne peut pas produire de l'électricité au coût le plus bas et qu'un générateur économique ne peut pas être le plus rentable, puisqu'un générateur qui se trouve trop loin du lieu de consommation entraine des pertes de transmission énormes. Cependant, ces pertes varient en fonction de la répartition des puissances entre les centrales et la charge.

Dans ce chapitre, nous avons brièvement discuté de l'écoulement de puissance optimal et la répartition économique de puissance, ensuite, nous avons cité quelques méthodes conventionnelles et intelligentes d'une façon générale. Dans le chapitre suivant, nous présentons en en détails trois méthodes conventionnelles pour la répartition économique de puissance.

Chapitre 2

Méthode d'Optimisation par Essaim de Particules à la REP

2.1 Définition

L'optimisation par essaim de particules (Particle Swarm Optimization) est une méthode d'optimisation stochastique, pour les fonctions non-linéaires, basée sur la reproduction d'un comportement social et développée par Eberhart et Kennedy [6] en 1995.

L'origine de cette méthode vient des observations faites lors des simulations informatiques de vols groupés d'oiseaux et de bancs de poissons. Ces simulations ont mis en valeur la capacité des individus d'un groupe en mouvement à conserver une distance optimale entre eux et à suivre un mouvement global par rapport aux mouvements locaux de leur voisinage.

D'autre part, ces simulations ont également révélé l'importance du mimétisme dans la compétition qui oppose les individus à la recherche de la nourriture. En effet, les individus sont à la recherche de sources de nourriture qui sont dispersés de façon aléatoire dans un espace de recherche, et dès lors qu'un individu localise une source de nourriture, les autres individus vont alors chercher à le reproduire.

Ce comportement social basé sur l'analyse de l'environnement et du voisinage constitue alors une méthode de recherche d'optimum par l'observation des tendances des individus voisins. Chaque individu cherche à optimiser ses chances en suivant une tendance qu'il modère par ses propres vécus.

Dans notre travail, nous utiliserons un algorithme d'optimisation par essaim de particules appelé ALPSO-II (Adaptive Learning Particle Swarm Optimizer-II) ou bien optimisation par essaim de particule avec apprentissage adaptatif II [7]

Dans la méthode d'OEP, un essaim de particules vole à travers l'espace de recherche. Chaque particule suit la meilleure position précédente trouvé par ses particules voisines et la précédente meilleure position trouvée par lui-même. Chaque particule est représentée par une position et une vitesse, qui sont mis à jour comme suit :

$$X_i^{k+1} = V_i^{k+1} + X_i^k \tag{2.1}$$

$$V_i^{k+1} = \omega V_i^k + c_1 rand_1(Pbest_i^k - X_i^k) + c_2 rand_2(Gbest^k - X_i^k) \tag{2.2}$$

Où :

- X_i^k : position de l'individu i à l'itération k.
- V_i^k : vitesse de l'individu i à l'itération k.
- ω : est un facteur d'inertie qui détermine de combien la vitesse précédente est conservée.
- $rand_1, rand_2$: des nombres aléatoires entre 0 et 1.
- c_1, c_2 : sont des constantes d'accélération.
- $Pbest_i$: meilleur position de l'individu i à l'iteration k.
- $Gbest^k$: meilleure position du groupe d'individus jusqu'à l'itération k.

Deux principaux modèles d'algorithmes d'OEP existent : *gbest* (meilleure position globale) et *lbest* (meilleure locale), leur différence étant dans la manière de définir le voisinage de chaque particule. Dans le modèle de *gbest*, le voisinage d'une particule se compose des particules de tout l'essaim, qui partagent des informations entre eux. Par contre dans le modèle *lbest*, le voisinage d'une particule est défini par plusieurs particules fixes. Le modèle de *gbest* a une vitesse de convergence plus rapide avec plus de chances de rester coincé en un optimum local que *lbest* [6].

Afin d'améliorer la performance de l'OEP, nous présentons une optimisation par essaim de particule avec apprentissage adaptatif (ALPSO), qui utilise une nouvelle stratégie d'apprentissage où chaque particule peut ajuster sa stratégie de recherche selon les rapports de sélection des quatre opérateurs d'apprentissage dans différents milieux. Pour la meilleure particule, nous introduisons une méthode d'apprentissage qui peut soustraire l'information de toutes les particules améliorés.

2.2 Stratégie d'apprentissage dans ALPSO

Dans la procédure d'optimisation avec ALPSO, l'information apprise par chaque particule provient de quatre sources :

— de *gbest*
— du propre *pbest*
— du *pbest* de la particule la plus proche
— et et une position aléatoire sur lui-même (l'individu).

Les equations d'apprentissage sont les suivantes :

$$exploitation : v_k^d = \omega v_k^d + \eta.r_k^d.(pbest_k^d - x_k^d) \tag{2.3}$$

$$exploration : v_k^d = \omega v_k^d + \eta.r_k^d.(pbest_{k-nearest}^d - x_k^d) \tag{2.4}$$

$$Le\ saut\ :\ x_k^d = x_k^d + v_{avg}^d.N(0,1) \tag{2.5}$$

$$convergence : V_k^d = \omega V_k^d + \eta.r_k^d.(gbest^d - X_k^d) \tag{2.6}$$

Tel que :

— $pbest_{k-nearest}$: est la *pbest* de la particule la plus proche de la particule k.
— V_{avg} : est la vitesse moyenne de tout les particules.
— $N(0,1)$: est un nombre aléatoire d'une distribution normal avec une moyenne 0 et une variance 1.

L'apprentissage à partir du voisin le plus proche permet à une particule d'explorer la région des optimum locaux autour d'elle. Les particules qui sont près d'un optimum local se rapprocheront de plus en plus de cette région parce que le *pbest* est remplacé uniquement lorsque une meilleure position est trouvée. Cette stratégie peut aider l'essaim à trouver plusieurs optimums locaux plutôt qu'un seul comme le PSO basique, et surtout pour les problèmes complexes.

Une fois que les particules convergent vers un optimum local ou bien, s'il y a une région plus prometteuse à proximité sans particules recouvrantes, les particules doivent avoir une probabilité pour accéder à cette région prometteuse. Ainsi, l'apprentissage d'une position aléatoire autour de lui même est nécessaire. Dans l'ALPSO, chaque particule a quatre choix différents pour ajuster son comportement. Les quatre choix permettent à chaque particule de se déplacer vers une position prometteuse avec une probabilité plus élevée que l'EOP de base.

2.2.1 Mécanisme d'apprentissage adaptatif

Dans notre travail, nous introduisons une structure adaptative utilisant les quatre opérateurs d'apprentissage précités, chacun affecté d'un rapport de sélection.

Ces opérateurs d'apprentissage jouent les rôles de convergence, d'exploitation, d'exploration et de saut sur les bassins d'attraction des optimums locaux.

Afin de permettre aux particules de choisir automatiquement un opérateur d'apprentissage approprié au cours du processus de la recherche, un mécanisme de sélection adaptative, basé sur l'hypothèse de l'opérateur le plus utilisé avec succès dans les récentes dernières itérations, peut être aussi efficaces dans les itérations suivantes. Pour chaque particule, l'un des quatre opérateurs d'apprentissage est sélectionné en fonction de leur rapports de sélection. L'opérateur qui résulte à la meilleure performance relative aura son rapport de sélection augmenté. L'opérateur le plus approprié sera choisi automatiquement pour une particule et va contrôler le comportement de recherche de la particule en fonction de son paysage de fitness locale au stade évolutif correspondant.

Pour toutes les particules, le rapport de sélection de chaque opérateur est également initialisé à 1/4 (à l'exception de la particule *gbest*, dans lequel le rapport de sélection initiale est 1/3) et est mis à jour en fonction de sa performance relative.

Les rapports de sélection des opérateurs d'une particule sont mis à jour s'ils ne s'améliorent pas pour une fréquence de mise à jour U_f pendant des itérations successives. Au cours de la période de mise à jour pour chaque particule, la valeur de progression et la valeur de la récompense de l'opérateur i sont calculés comme suit.
La valeur de progression $p_i^k(t)$ de l'opérateur i pour la particules k à l'itération t est défini comme :

$$p_i^k(t) = \begin{cases} |f(\overrightarrow{x}_k(t)) - f(\overrightarrow{x}_k(t-1))| & si \;\; l'operateur\ i\ est\ choisit\ par\ \overrightarrow{x}_k(t) \\ & et\ ce\ dernier\ est\ mieux\ que\ \overrightarrow{x}_k(t-1) \\ 0 & sinon. \end{cases} \tag{2.7}$$

et l'expression de la valeur de récompense $r_i^k(t)$ est :

$$r_i^k(t) = \frac{p_i^k(t)}{\sum_{j=1}^{M} p_j^k(t)}\alpha + \frac{g_i^k}{G_i^k}(1-\alpha) + c_i^k s_i^k(t). \tag{2.8}$$

La valeur de récompense $r_i^k(t)$comporte trois éléments, qui sont la valeur de progression normalisée, le taux de réussite, et le rapport de sélection précédent avec :

- g_i^k : compteur qui enregistre le nombre de fois d'apprentissage réussis de la particule k, où son enfant est plus en forme que particule k en appliquant l'opérateur i depuis la dernière mise à jour du rapport de sélection
- G_i^k : nombre total d'itérations où l'opérateur i est sélectionné par la particule k depuis la dernier mise à jour du rapport de sélection
- $\frac{g_i^k}{G_i^k}$: taux de réussite de l'opérateur i pour la particule k
- α : nombre aléatoire entre 0 est 1
- M : nombre d'opérateurs
- c_i^k : facteur de pénalité de l'opérateur i pour la particule k
- et s_i^k : rapport de sélection de l'opérateur i pour la particule k à l'itération courante.

Le rapport de sélection de l'opérateur i de la particule k à l'itération suivante $t+1$ est écrit comme suit :

$$s_i^k(t+1) = \frac{r_i^k(t)}{\sum_{j=1}^{M} r_j^k(t)}(1 - M * \gamma) + \gamma. \tag{2.9}$$

avec γ rapport de sélection minimum pour chaque opérateur affecté à la valeur 0.01. Pour la particule *gbest* dans ALPSO, elle sera mise à jour tant qu'une particule va mieux au fil du temps par l'extraction d'informations de cette particule améliorée.

Nous observons deux paramètres clés dans l'ALPSO : la fréquence de mise à jour (U_f) et la probabilité d'apprentissage (P_l). Les valeurs de U_f et P_l affectent de manière significative la performance de l'ALPSO. Cette dernière (ALPSO) introduit quelques méthodes pour choisir les valeurs optimales des deux paramètres. Pour le paramètre de la fréquence de mise à jour (U_f), chaque particule est affectée à une valeur de U_f au lieu d'utiliser une même valeur toute les particules. La valeur de U_f est définie pour chaque particule k comme suit :

$$U_f^k = max(10 * exp(-(1.6.k/N)^4, 1) \tag{2.10}$$

avec N : la taille de population et U_f^k est la fréquence de mise à jour de la particule K. Même chose pour le paramètre P_l, chaque particule K a son propre paramètre. P_l est écrit comme suit :

$$P_l^k = max(1 - exp(-(1.6.k/N)^4, 0.05) \tag{2.11}$$

Pour ajuster la valeur de P_l d'une façon adaptative, ALPSO a besoin de calculer le rapport d'amélioration de particule définit par :

$$IMPR_k(t) = max(\frac{f(\overrightarrow{x_k}(t-1)) - f(\overrightarrow{x_k}(t))}{f(\overrightarrow{x_k}(t-1))}, 0) \tag{2.12}$$

avec $IMPR_k$ est le rapport d'amélioration de particule k entre les deux itérations $t-1$, et t.

2.3 ALPSO-II

Dans cette partie, nous nous penchons en détails sur la technique ALPSO-II. D'abord, deux opérateurs d'apprentissage de l'ALPSO sont remplacés par deux nouveaux opérateurs d'apprentissage. Deuxièmement, un mécanisme de contrôle est introduit pour sur- veiller l'état des particules. Enfin, une approche visant à contrôler le nombre de particules qui apprennent à partir de la meilleure position globale (nommée position de abest) est ajouté dans ALPSO-II. Le but de toutes ces améliorations est d'augmenter la diversité de sorte que ALPSO- II peut chercher beaucoup plus de meilleures solutions dans le paysage complexe de remise en forme.

2.3.1 Opérateurs d'apprentissage dans ALPSO-II

Dans ALPSO-II, nous utilisons également quatre opérateurs d'apprentissage, mais l'opérateur "d'apprentissage de la *pbest* de la particule la plus proche" (opérateur d'exploration) est remplacer par "un opérateur d'apprentissage de la *pbest* d'une particule aléatoire ", et chaque particules apprend à partir d'une position archive de *gbest*, (*abest*) à la place de l'apprentissage à partir de la particule *gbest*. Les deux opérateurs d'apprentissage mises à jour sont définis comme suit :
Opérateur c' : apprentissage à partir de *pbest* d'une particule aléatoire.

$$exploration : v_k^d = \omega v_k^d + \eta.r_k^d.(pbest_{rand}^d - x_k^d) \tag{2.13}$$

Opérateur d' : apprentissage à partir de la position *abest*

$$convergence : v_k^d = \omega v_k^d + \eta.r_k^d.(abest^d - x_k^d) \tag{2.14}$$

où la position de abest est utilisée pour sauvegarder la meilleure position trouvée par ALPSO-II jusque là. Dans ALPSO-II, une seule particule apprend d'une position de $pbest_{rand}$ une position meilleure que sa propre meilleure pbest de position historique. Grâce à cette stratégie, plus de ressources de calcul sont données aux particules les moins performantes pour améliorer tout l'essaim.

En introduisant le nouvel opérateur d'exploration, ALPSO-II permet à une particule d'explorer le paysage non recherché remis en forme avec une probabilité supérieure à celle d'ALPSO où cette particule va apprendre d'une particule aléatoire au lieu de son voisinage le plus proche [8]

2.3.2 Surveillance de l'état des particules

D'une manière générale, la réinitialisation est une méthode pour augmenter la diversité de la population dans les algorithmes évolutionnaires. Cependant, il y a des situations où en utilisant cette méthode, la protection des individus réinitialisés n'est pas assurée et ces individus sont éliminés. Nous disposons de plusieurs façons de vérifier le moment où nous devons effectuer une réinitialisation.

La première méthode est de vérifier la diversité de la population. Si la diversité est inférieure à un seuil, nous effectuons une réinitialisation. La seconde méthode consiste à surveiller la particule *gbest*, si elle ne s'améliore pour un certain nombre d'itérations, la réinitialisation peut être lancée. Pour les algorithmes E O P , nous pouvons surveiller la vitesse des particules. Si l'amplitude de la vitesse d'une particule est inférieure à une valeur de seuil, on peut réinitialiser cette particule.

Quelle que soit la méthode que nous utilisons, nous devons définir une valeur de seuil pour effectuer cette opération. Cependant, il est très difficile d'obtenir une valeur de seuil optimale pour un problème particulier. En outre, les valeurs de seuil pour les différents problèmes peuvent être différentes.

Le problème commun des approches ci-dessus, c'est qu'elles ne peuvent pas savoir si une particule est à l'étape de l'évolution ou à l'étape de convergence. Si cette particule converge, nous pouvons effectuer une réinitialisation. Afin de surveiller l'état de particules, nous introduisons un mécanisme pour vérifier si une particule est à l'état de convergence.

Dans ALPSO-II, il existe un mécanisme de contrôle de la performance des quatre i opérateurs d'apprentissage. L'approche consiste à surveiller les rapports des quatre opérateurs d'apprentissage de sélection. Une fois qu'une particule converge vers un optimum local, et qu'aucun des quatre opérateurs ne peut l'aider à sauter hors de ce dernier, leur rapport de sélection va revenir à l'étape initiale où ils ont des valeurs égales à 1/4. Par conséquent, nous pouvons utiliser cette information pour examiner si une particule a convergé ou pas. En utilisant cette approche, nous pouvons facilement éviter les problèmes cités précédemment pour la réinitialisation des particules. Pour atteindre cet objectif, en calculant les rapports de sélection normale comme dans ALPSO, nous avons besoin de créer un rapport de sélection de surveillance pour chaque opérateur d'apprentissage. Dans ALPSO-II, toute définition, fonctionnement de calcul, et mise à jour des rapports de sélection de suivi sont les mêmes que dans ALPSO pour calculer et mettre à à jour les rapports de sélection normale, sauf le calcul de $p_i^{'k}$ valeur de progression de l'opérateur i pour particule k à l'itération t, qui est défini comme suit :

$$p_i^{'k}(t) = \begin{cases} |f(\overrightarrow{x}_k(t)) - f(\overrightarrow{x}_k^{pbest})| & si\ l'operateur\ i\ est\ choisit\ par\ \overrightarrow{x}_k(t) \\ & et\ ce\ dernier\ est\ mieux\ que\ \overrightarrow{x_k}^{pbest} \\ 0 & sinon. \end{cases} \tag{2.15}$$

Pour distinguer les définitions relatives à la mise à jour des rapports de sélection de contrôle dans les deux algorithmes différents, nous mettons un symbole prime après chaque définition, par exemple, $p'_k(t)$ et $p_k(t)$ représentent la valeur de la surveillance de progrès et de la valeur de progrès commune de l'opérateur i pour la particule k à l'itération t, respectivement.

Dans ALPSO-II, les rapports de sélection communs et les rapports de sélection de surveillance sont mis à jour en même temps et une fois qu'ils sont mis à jour, tous les paramètres de composants sont remis à l'état initial : les valeurs de progrès, les valeurs de récompenses, les taux de réussite sont tous misent à 0.

La ré-initialisation d'une particule est effectuée une fois la variance des rapports de sélection de contrôle est inférieur à $0,05$.

2.3.3 Contrôle du nombre de particules qui apprennent de la position de abest

Dans l'algorithme d'ALPSO, tout en accomplissant la recherche locale pour une particule dépend de la performance des opérateurs de recherche locaux (par exemple l'opérateur de l'exploitation et l'opérateur d'exploration), il y a encore une chance de faire une très bonne recherche globale.

Comme nous le savons, les particules, qui sont loin de la position de *abest*, ne peuvent pas obtenir des avantages en apprenant de lui surtout pour les problèmes multi-modales. Dans ALPSO-II, pour faire équilibrer la recherche globale et la recherche locale, nous permettons seulement un certain nombre de particules (Q), qui sont à proximité de la position de *abest*, à apprendre de ce dernier.

En fait, ALPSO-II permet juste aux Q particules d'utiliser les quatre opérateurs d'apprentissage et les autres particules n'utilisent pas l'opérateur de la convergence.

2.3.4 Analyse des résultats par la méthode ALPSO-II

Le tableau suivant nous montre les résultats obtenus par la méthode d'ALPSO-II sur le réseau 30-bus.

TABLE 2.1: Les valeurs de Pg obtenu par ALPSO-II

Numéro de nœud	P_g (MW)
1	44.79
2	58.30
13	22.31
22	32.31
23	15.86
27	15.81
Pg total(MW)	189.38
Le cout ($\$/h$)	565.97

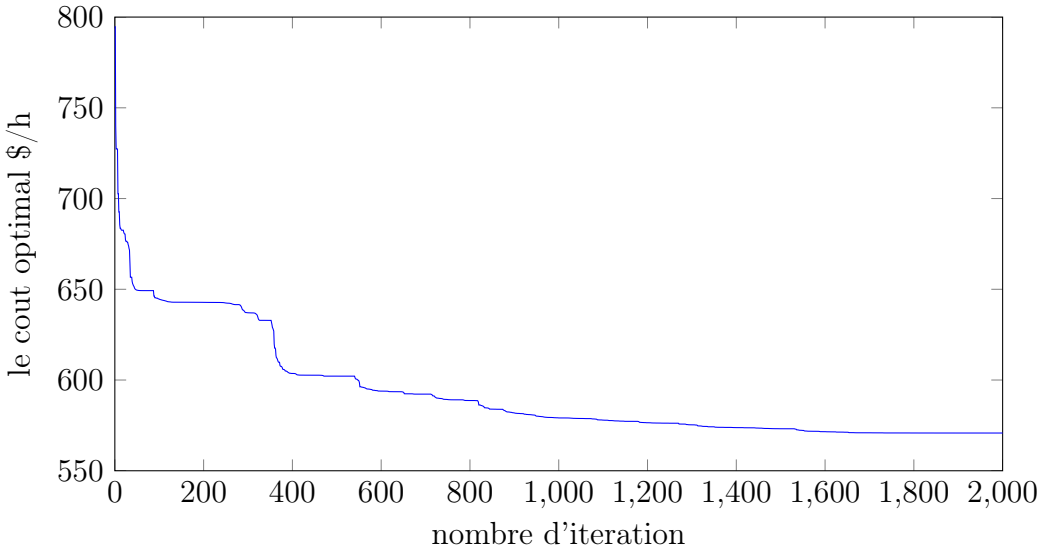

FIGURE 2.1: Méthode ALPSO

La puissance totale générée est : 189.38 MW et la puissance demandée est : 189.20 MW. Par conséquence, les pertes sont 0.2 MW. En comparaison, la puissance générée totale par la méthode de Newton est égale 191.62MW. Donc, on conclut que ALPSO-II réduit les pertes mieux que la méthode de Newton.

Conclusion

Ce mémoire nous a permis de traiter le problème d'écoulement de puissance optimal qui est un des problèmes les plus en vue dans le domaine du fonctionnement des réseaux électriques, surtout que les besoins en énergie électrique augmentent continuellement avec les besoins socio-économiques croissants dans toutes les sociétés du monde. Ce mémoire a essayé d'être une courte rétrospective de méthodes ayant eu à résoudre ce problème, que ce soit des méthodes conventionnelles basées sur des techniques analytiques de fonctions issues de la modélisation des systèmes d'énergie de puissance ou de méthodes non conventionnelles basées principalement sur des techniques d'intelligence artificielle. Notons que nous nous sommes concentrés sur le cas de la répartition économique, qui est un des problèmes les plus importants de l'écoulement de puissance optimal.

Nous nous sommes intéressés à la technique d'optimisation par essaim de particules avec un paramètre d'adaptation pour améliorer l'efficacité de la technique de base. Cette technique a donné de meilleurs résultats que les méthodes conventionnelles mais le temps de calcul est un peu élevé.

Nous proposons comme éventuelles perspectives :

— l'amélioration et la diminution du temps de calcule des méthodes intelligentes
— combiner entre les méthodes conventionnelles et non conventionnelles par le développement d'une sorte d'algorithmes hybrides

Références

[1] M. JA, E.-H. ME, and A. R, "A review of selected optimal power flow literature to 1993,part i :nonlinear quadratic programming approach," *IEEE trans Power Sys*, pp. 96–104, 1999.

[2] H. Atun and T.yalcinoz, "Implementing soft computing techniques to solve economic dispatch problem in power systems," *xpert Systems with Applications*, vol. 35, 2008.

[3] O. Guenounou, *Méthodologie de conception de contrôleurs intelligents par l'approche génétique- application à un ioprocédé*. PhD thesis, Université Toulouse III, Paul Sabatier.

[4] M.Basu, "Economic enviremental dispatch using multiobjective differential evolution," *Applied soft Computing*, vol. 11, pp. 2845–2853, 2011.

[5] M. Clerc and J. Kennedy, "The particle swarm : Explosion, stability and convergence in a multi-dimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, 2001.

[6] Y. S. Russell C. Eberhart and J. Kennedy, *Swarm Intelligence*. The Morgan Kaufmann Series in Artificial Intelligence, San Francisco, CA, USA : Morgan Kaufmann, 2001.

[7] C. Li and S. Yang, "Adaptive learning particle swarm optimizer-ii for global optimization," *Congress on Evolutionary Computation (CEC), IEEE*, pp. 1–8, 2010.

[8] C. Li and S. Yang, "An adaptive learning particle swarm optimizer for function optimization," *Proc. of the 2009 IEEE Congress on Evolutionary Computation*, pp. 381–388, 2009.

Annexe A

Système 30 nœuds

```
function mpc = case30
%CASE30    Power flow data for 30 bus, 6 generator case.
%   Please see CASEFORMAT for details on the case file format.
%
%   Based on data from ...
%     Alsac, O. & Stott, B., "Optimal Load Flow with Steady State Security",
%     IEEE Transactions on Power Apparatus and Systems, Vol. PAS 93, No. 3,
%     1974, pp. 745-751.
%   ... with branch parameters rounded to nearest 0.01, shunt values divided
%   by 100 and shunt on bus 10 moved to bus 5, load at bus 5 zeroed out.
%   Generator locations, costs and limits and bus areas were taken from ...
%     Ferrero, R.W., Shahidehpour, S.M., Ramesh, V.C., "Transaction analysis
%     in deregulated power systems using game theory", IEEE Transactions on
%     Power Systems, Vol. 12, No. 3, Aug 1997, pp. 1340-1347.
%   Generator Q limits were derived from Alsac & Stott, using their Pmax
%   capacities. V limits and line |S| limits taken from Alsac & Stott.

%   MATPOWER
%   $Id: case30.m 2408 2014-10-22 20:41:33Z ray $

%% MATPOWER Case Format : Version 2
mpc.version = '2';

%%-----  Power Flow Data  -----%%
```

```
%% system MVA base
mpc.baseMVA = 100;

%% bus data
% bus_i type Pd Qd Gs Bs area Vm Va baseKV zone Vmax Vmin
mpc.bus = [
1 3 0 0 0 0 1 1 0 135 1 1.05 0.95;
2 2 21.7 12.7 0 0 1 1 0 135 1 1.1 0.95;
3 1 2.4 1.2 0 0 1 1 0 135 1 1.05 0.95;
4 1 7.6 1.6 0 0 1 1 0 135 1 1.05 0.95;
5 1 0 0 0 0.19 1 1 0 135 1 1.05 0.95;
6 1 0 0 0 0 1 1 0 135 1 1.05 0.95;
7 1 22.8 10.9 0 0 1 1 0 135 1 1.05 0.95;
8 1 30 30 0 0 1 1 0 135 1 1.05 0.95;
9 1 0 0 0 0 1 1 0 135 1 1.05 0.95;
10 1 5.8 2 0 0 3 1 0 135 1 1.05 0.95;
11 1 0 0 0 0 1 1 0 135 1 1.05 0.95;
12 1 11.2 7.5 0 0 2 1 0 135 1 1.05 0.95;
13 2 0 0 0 0 2 1 0 135 1 1.1 0.95;
14 1 6.2 1.6 0 0 2 1 0 135 1 1.05 0.95;
15 1 8.2 2.5 0 0 2 1 0 135 1 1.05 0.95;
16 1 3.5 1.8 0 0 2 1 0 135 1 1.05 0.95;
17 1 9 5.8 0 0 2 1 0 135 1 1.05 0.95;
18 1 3.2 0.9 0 0 2 1 0 135 1 1.05 0.95;
19 1 9.5 3.4 0 0 2 1 0 135 1 1.05 0.95;
20 1 2.2 0.7 0 0 2 1 0 135 1 1.05 0.95;
21 1 17.5 11.2 0 0 3 1 0 135 1 1.05 0.95;
22 2 0 0 0 0 3 1 0 135 1 1.1 0.95;
23 2 3.2 1.6 0 0 2 1 0 135 1 1.1 0.95;
24 1 8.7 6.7 0 0.04 3 1 0 135 1 1.05 0.95;
25 1 0 0 0 0 3 1 0 135 1 1.05 0.95;
26 1 3.5 2.3 0 0 3 1 0 135 1 1.05 0.95;
27 2 0 0 0 0 3 1 0 135 1 1.1 0.95;
28 1 0 0 0 0 1 1 0 135 1 1.05 0.95;
29 1 2.4 0.9 0 0 3 1 0 135 1 1.05 0.95;
30 1 10.6 1.9 0 0 3 1 0 135 1 1.05 0.95;
```

```
];

%% generator data
% bus Pg Qg Qmax Qmin Vg mBase status Pmax Pmin Pc1 Pc2 Qc1min Qc1max
Qc2min Qc2max ramp_agc ramp_10 ramp_30 ramp_q apf
mpc.gen = [
1 23.54 0 150 -20 1 100 1 80 0 0 0 0 0 0 0 0 0 0 0 0;
2 60.97 0 60 -20 1 100 1 80 0 0 0 0 0 0 0 0 0 0 0 0;
22 21.59 0 62.5 -15 1 100 1 50 0 0 0 0 0 0 0 0 0 0 0 0;
27 26.91 0 48.7 -15 1 100 1 55 0 0 0 0 0 0 0 0 0 0 0 0;
23 19.2 0 40 -10 1 100 1 30 0 0 0 0 0 0 0 0 0 0 0 0;
13 37 0 44.7 -15 1 100 1 40 0 0 0 0 0 0 0 0 0 0 0 0;
];

%% branch data
% fbus tbus r x b rateA rateB rateC ratio angle status angmin angmax
mpc.branch = [
1 2 0.02 0.06 0.03 130 130 130 0 0 1 -360 360;
1 3 0.05 0.19 0.02 130 130 130 0 0 1 -360 360;
2 4 0.06 0.17 0.02 65 65 65 0 0 1 -360 360;
3 4 0.01 0.04 0 130 130 130 0 0 1 -360 360;
2 5 0.05 0.2 0.02 130 130 130 0 0 1 -360 360;
2 6 0.06 0.18 0.02 65 65 65 0 0 1 -360 360;
4 6 0.01 0.04 0 90 90 90 0 0 1 -360 360;
5 7 0.05 0.12 0.01 70 70 70 0 0 1 -360 360;
6 7 0.03 0.08 0.01 130 130 130 0 0 1 -360 360;
6 8 0.01 0.04 0 32 32 32 0 0 1 -360 360;
6 9 0 0.21 0 65 65 65 0 0 1 -360 360;
6 10 0 0.56 0 32 32 32 0 0 1 -360 360;
9 11 0 0.21 0 65 65 65 0 0 1 -360 360;
9 10 0 0.11 0 65 65 65 0 0 1 -360 360;
4 12 0 0.26 0 65 65 65 0 0 1 -360 360;
12 13 0 0.14 0 65 65 65 0 0 1 -360 360;
12 14 0.12 0.26 0 32 32 32 0 0 1 -360 360;
12 15 0.07 0.13 0 32 32 32 0 0 1 -360 360;
12 16 0.09 0.2 0 32 32 32 0 0 1 -360 360;
```

```
14 15 0.22 0.2 0 16 16 16 0 0 1 -360 360;
16 17 0.08 0.19 0 16 16 16 0 0 1 -360 360;
15 18 0.11 0.22 0 16 16 16 0 0 1 -360 360;
18 19 0.06 0.13 0 16 16 16 0 0 1 -360 360;
19 20 0.03 0.07 0 32 32 32 0 0 1 -360 360;
10 20 0.09 0.21 0 32 32 32 0 0 1 -360 360;
10 17 0.03 0.08 0 32 32 32 0 0 1 -360 360;
10 21 0.03 0.07 0 32 32 32 0 0 1 -360 360;
10 22 0.07 0.15 0 32 32 32 0 0 1 -360 360;
21 22 0.01 0.02 0 32 32 32 0 0 1 -360 360;
15 23 0.1 0.2 0 16 16 16 0 0 1 -360 360;
22 24 0.12 0.18 0 16 16 16 0 0 1 -360 360;
23 24 0.13 0.27 0 16 16 16 0 0 1 -360 360;
24 25 0.19 0.33 0 16 16 16 0 0 1 -360 360;
25 26 0.25 0.38 0 16 16 16 0 0 1 -360 360;
25 27 0.11 0.21 0 16 16 16 0 0 1 -360 360;
28 27 0 0.4 0 65 65 65 0 0 1 -360 360;
27 29 0.22 0.42 0 16 16 16 0 0 1 -360 360;
27 30 0.32 0.6 0 16 16 16 0 0 1 -360 360;
29 30 0.24 0.45 0 16 16 16 0 0 1 -360 360;
8 28 0.06 0.2 0.02 32 32 32 0 0 1 -360 360;
6 28 0.02 0.06 0.01 32 32 32 0 0 1 -360 360;
];

%%-----  OPF Data  -----%%
%% generator cost data
% 1 startup shutdown n x1 y1 ... xn yn
% 2 startup shutdown n c(n-1) ... c0
mpc.gencost = [
2 0 0 3 0.02 2 0;
2 0 0 3 0.0175 1.75 0;
2 0 0 3 0.0625 1 0;
2 0 0 3 0.00834 3.25 0;
2 0 0 3 0.025 3 0;
2 0 0 3 0.025 3 0;
];
```