RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abderrahmane Mira de Béjaïa
Faculté de Technologie
Département de Genie Électrique

Mémoire de fin de cycle, présenté par :
ATMANI BILLAL
En vue de l'obtention du diplôme de Master en Automatique

Thème

Optimisation des paramètres du Contrôleur PID par Algorithme Génétique Multi-objectifs

Présidente du jury : Mme BELLAHSEN, Maître Assistant, Université de Béjaïa.

Examinateurs : M. HADDAR, Maître Assistant, Université de Béjaïa.

Promoteur : M. GUENOUNOU W, Maître Assistant, Université de Béjaïa.

Septembre 2011

Remerciements

Ce mémoire ne pouvais pas avoir mené à bien sans la confiance, la patience, la rigueur et la générosité de M. Guenounou, à qui je veut apporter mes remerciements les plus sincères, vous avez su m'aider et orienter mon travail avec professionnalisme et gentillesse. je remercie tout les professeurs qui nous ont initier aux différentes branches de l'automatique et de l'éléctronique, tout le personnel du département de l'électronique

je ne saurais oublier mes collègues et amis avec les quels j'ai partager de très bon moment durant les cinq années de mes études à l'université de Bejaia,HICHAME et KARIM de m'avoir accueilli dans leur chambre, ainsi que mon binôme ZERARI FARES

Dédicaces

A la mémoire de mon grand père.
A la mémoire du défunt M. BELMEHDI que j'ai eu la chance de discuter mes choix de spécialité.
A mes parents, mes frères GHANI qui me "dépanne" quand je chôme ainsi que sa femme ZINA que je souhaite une bonne intégration à sa nouvelle famille, ABDELHAK, HABIB, NASSIM qui lui aussi participe au "Dépannage" et à mon unique et aimable soeur Sonia.
A mes amis
A tous ceux que j'ai oublier.
Longue vie aux chouchoux de la famille : SAMY,LINA et YANIS.

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

De nombreux procédés industriels peuvent être modélisés par des systèmes linéaires invariants dans le temps. En conséquence, au cours des dernières décennies, de nombreuses méthodes avancées de commande ont été développées pour de tels systèmes. Les plus connues sont la commande Linéaire Quadratique Gausienne, la commande H_∞, la commande par placement de pôles. Ces méthodes ont été appliquées avec succès, néanmoins deux inconvénients majeurs, qui les rendent insuffisamment utilisées dans l'industrie, peuvent être mentionnés.
Premièrement, les techniques de commande avancées mènent le plus souvent à des régulateurs d'ordre élevé et leur implantation peut être économiquement problématique.
Deuxièmement, ces méthodes reposent sur un modèle mathématique du système à commander qui n'est souvent obtenu qu'aux prix d'efforts considérables. En pratique, le modèle du système, utilisé pour la synthèse de la commande doit être précisément construit dans cette optique là. En effet, d'un côté s'il est "trop" complexe, il est inexploitable pour la synthèse, d'un autre côté, s'il est "trop" simple, il ne rend pas suffisamment bien compte du comportement du système.
C'est d'ailleurs pour ces inconvénients que la commande de type PID s'est généralisée dans l'industrie, on est donc très souvent contraint, par les inconvénients cités en haut, d'utiliser ce type de commande. Un autre avantage lié à l'utilisation des PID est l'existence de méthodes simples qui ne demandent que très peu de temps et d'effort à l'ingénieur pour faire un choix des gains de ces régulateurs. Parmi ces méthodes, on peut citer la méthode de Ziegler-Nichols(ZN) [1], et celle de Cohen et Coon (CC) qui sont très largement répandues dans l'industrie. En effet, ce type de méthode requiert simplement un essai expérimental relativement aisé à réaliser, puis à partir de l'obtention de deux ou trois grandeurs relatives à l'essai, les gains du régulateur sont déterminés par des abaques ou des formules très simples. Néanmoins avec ce type de méthodes, seule l'erreur entre la sortie du système et la sortie désirée peut être optimisée, d'autres grandeurs qui présentent un intérêt économique comme l'effort de commande ne peut être minimisées. L'utilisation de nouvelles méthodes de réglage pouvant considérer plus d'un objectif peut être une solution fiable pour contourner cet inconvénient. C'est dans ce cadre que se situent nos motivations pour la conception des régulateurs de type PID par algorithmes génétiques multi-objectifs.
Ce mémoire est organisé en quatre chapitres comme suit :
Le premier chapitre est consacré à l'étude du régulateur PID ainsi qu'à la descrip-

tion des méthodes classiques de réglage des ces gains. Des exemples de simulation sont utilisés pour appuyer cette étude.
Le deuxième chapitre concerne les algorithmes génétiques simples. Nous présentons une description détaillée des algorithmes génétiques simples puis nous rappelons les définitions relatives à leur fonctionnement.
Le troisième chapitre traite l'optimisation multi-objectifs par algorithmes génétiques. Nous décrivons quelques algorithmes de résolution tout en accordant plus d'importance à l'algorithme SPEA2 (Strength Pareto Evolutionary Algorithm-2) qui est utilisé dans ce travail.
Le quatrième chapitre est divisé en deux parties. La première partie traite de l'optimisation mon-objectif du régulateur PID par algorithme génétiques simple. La deuxième partie concerne l'utilisation de l'algorithme SPEA2 pour l'optimisation des paramètres du régulateur PID destiné à la commande d'un moteur à courant continu.

Chapitre 1

Régulateur pid

On s'intéresse dans ce chapitre à l'étude des actions du module PID composé par les trois actions de base. On mettra en évidence l'effet produit par chaque action dans une boucle de régulation, ses avantages ainsi que ses limitations.
Ce premier chapitre sert à nous introduire au coeur de la problématique,en effet,comment et par quel outil ajuster les paramètres du régulateur. Nous exposerons quelques méthodes usuelles utilisées pour ajuster ces paramètres à savoir Ziegler-Nichols, Cohen et Coon et la méthode du modèle.

1.1 Introduction

Le contrôleur PID est la forme la plus commune de rétroaction.C'était l'outil standard quand le contrôle des processus à émergé dans les années 40. Il est basé sur un mécanisme de rétroaction, largement répandu dans les systèmes de commande industriel ; grace notamment à sa simple structure et sa stratégie de commande.

Le calcul du contrôleur PID (algorithme) implique trois paramètres : le proportionnel, les deux valeurs intégrales et dérivées respectivement dénoté I et D. Ces valeurs peuvent être interprétées en termes d'erreur ; P dépend de l'erreur actuelle, I sur l'accumulation des erreurs passées, et D est une prévision de futures erreurs. La somme pondérée de ces trois actions est employée pour ajuster le processus.En accordant les trois paramètres dans l'algorithme du contrôleur ainsi que des conditions spécifiques, le contrôleur peut fournir une action de commande qui répond à nos attente. les conditions spécifiques peuvent être décrites en termes de réponse du contrôleur à une erreur, le degré avec lequel le contrôleur dépasse la consigne et le degré d'oscillation de système.

Le régulateur, dont la fonction de transfert est désignée par $G_c(s)$ est situé en amont du système à régler Ga(s).L'entrée du régulateur comprend forcément la consigne w(t) et la mesure y(t) de la grandeur réglée. Le plus souvent la comparaison e(t)=w(t)-y(t) directe est effectuée, appelée écart ou erreur.

Le régulateur à pour charge de maintenir le signal d'erreur e(t) aussi proche de zéro que possible ; dans ce but, il fournit au système à régler la commande u(t) telle que l'image y(t) de la grandeur réglée obtenue par mesure tende à correspondre à la consigne w(t).

La commande u(t) est construite sur la base des signaux de consigne w(t) et de mesure y(t) de la grandeur réglée selon la loi de commande u(t)=u(w(t)-y(t)). Appliquée au système à régler, la commande u(t) provoque donc une modification de la grandeur réglée y(t).

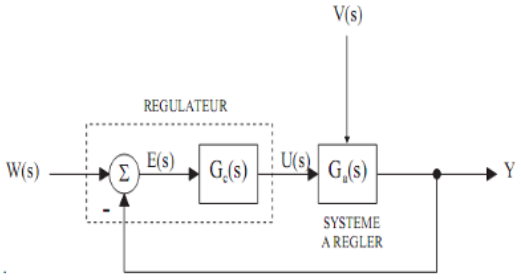


FIGURE 1.1 – Schéma fonctionnel

1.2 Régulateur à action proportionnelle

Le régulateur à action proportionnelle, ou régulateur P, à une action simple et naturelle, puisqu'il fournit une commande u(t) proportionnelle à l'erreur e(t).

La loi de commande du régulateur P est :

$$u(t) = \ k_p \cdot e(t) \tag{1.1}$$

et sa fonction de transfert est de la forme :

$$G_c(s) = \frac{U(s)}{E(s)} \tag{1.2}$$

Comme exemple illustratif,on considère la fonction de transfert suivante $G_a = \frac{2}{p^2+1.4p+2}$. On asservi le système par un régulateur proportionnel,avec $k_p = 15$

La figure 1.2 donne la réponse indicielle du système en boucle fermée sans le contrôleur

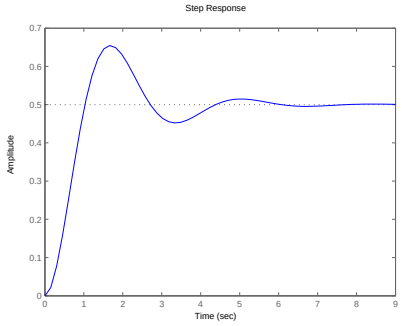


FIGURE 1.2 – Réponse indicielle du système sans contrôleur

Nous avons un dépassement et une erreur statique égale à 0.5 ; mais avec le régulateur, l'erreur est considérablement réduite de 0.5 à 0.01 comme le montre la figure 1.3.

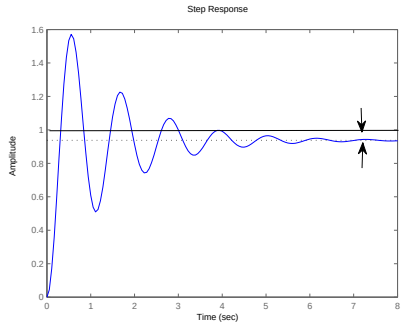


FIGURE 1.3 – Réponse indicielle du système asservi par un contrôleur type P

L'erreur statique subsiste toujours, car le signal de commande u(t) à appliquer au système à régler doit être non-nul pour que y(t) atteigne un niveau différent de zéro.

On serait tenté de prendre des valeurs de gain élevées pour accélérer la réponse du système et minimiser l'erreur, mais on est limité par la stabilité de la boucle fermée. En effet, une valeur trop élevée du gain augmente l'instabilité du système et donne lieu à des oscillations très importantes (voir figure 1.4)

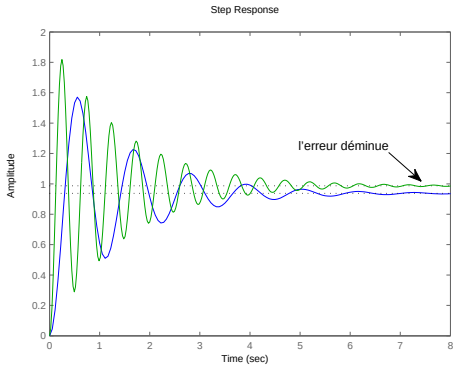


FIGURE 1.4 – Apparition de forte oscillation lors d'augmentation de la valeur k_p (passé de 15 à 80)

1.3 Régulateur à action intégrale

Pour remédier au problème de l'erreur statique, vus précédemment pour le cas d'un régulateur P qui présente une erreur permanente en régime permanent, on constante qu'on pourrait dans un premier temps augmenter la consigne de la valeur de l'erreur statique constatée

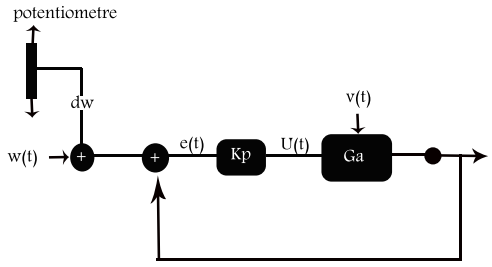


FIGURE 1.5 – Annulation de l'erreur statique par décalage de la consigne

On pourrait décider d'agir directement sur la commande u(t) "figure 1.6" en procédant comme suit :

- ajouter à la commande $u_p(t)$issue du régulateur P la quantité ajustable $u_i(t)$
- augmenter ou diminuer $u_i(t)$ progressivement jusqu'à ce que e(t)soit nulle
- $u_p(t)$ est alors nulle $u_p = 0$ et u_i est exactement égale à la valeur nécessaire à la compensation de l'erreur statique, et bien que l'erreur soit nulle, la commande $u(t) = u_i(t) + u_p(t)$ est bel et bien non nulle.

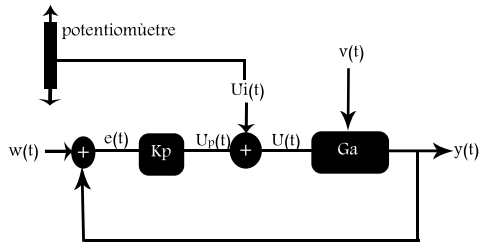


FIGURE 1.6 – Annulation de l'erreur statique par augmentation du signal de commande

L'observateur au potentiomètre accumule l'erreur et entreprend une action (K_i) proportionnelle à cette accumulation

La loi de commande est donc :

$$u(t) = \frac{1}{T_i} \int_{-\infty}^{t} e(\tau) d\tau \tag{1.3}$$

La commande proposée est formée des deux contributions U_p et U_i, contributions proportionnelle (P) et intégrale (I). Le régulateur est donc à actions proportionnelle et intégrale : c'est un régulateur PI
Loi de commande du régulateur PI :

$$u(t) = K_p \left\{ e(t) + \frac{1}{T_i} \int_{-\infty}^{t} e(\tau) d\tau \right\} \tag{1.4}$$

Fonction de transfert du régulateur PI

$$G_c(s) = k_p \cdot \frac{1 + ST_i}{ST_i} \tag{1.5}$$

On ajoute au régulateur précédent P une action intégrale avec $K_p = 15$ et K_i=1 et on aura la réponse indicielle suivante "figure 1.7"

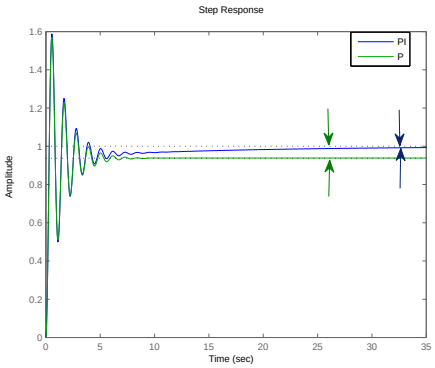


FIGURE 1.7 – Réponse indicielle du système asservi par un contrôleur PI

A travers la figure en haut, on constate que l'intégrateur à jouer son rôle principal qui est la compensation de l'erreur statique.
L'action P du régulateur PI n'est pas utile du point de vue de la précision en régime permanent ; cependant, le fait que l'action P permette la transmission instantanée du signal d'erreur rend le régulateur PI plus dynamique que le régulateur I pur discuté plus tard, mis a part dans quelques cas particuliers où le critère de performance "rapidité" n'est pas important et où l'on souhaite avoir une action relativement "molle" sur le système à régler.

La loi de commande du régulateur I est :

$$u(t) = \frac{k_p}{T_i} \int_{\infty}^{t} e(\tau) d\tau \tag{1.6}$$

et la fonction de transfert du régulateur I s'écrit :

$$G_c(s) = \frac{k_p}{ST_i} = \frac{k_i}{S} \tag{1.7}$$

Pour mètre en évidence l'action intégrale pure I, on effectue trois essais avec des K_i différents :($K_i = 0.1$, $K_i = 0.5$ et $K_i = 0.7$) sur le même système présenté en haut

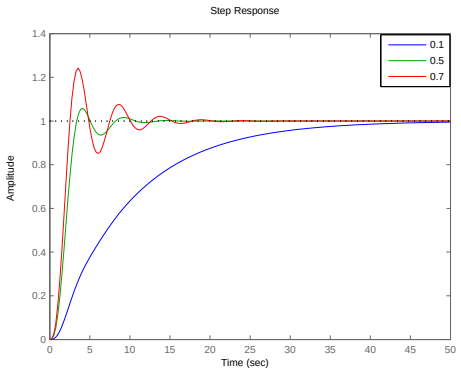


FIGURE 1.8 – Réponse indicielle du système asservi par un contrôleur I

k_i	$T_r(10 \Longrightarrow 90\% vf\ [sec])$	$T_e(\pm 5\% vf\ [sec])$	$D\%$
0.1	20.2	37.1	0
0.5	2	7.73	5.76
0.7	1.5	14	24.1

TABLE 1.1 – Résultats de la simulation de l'asservissement I

On constate à travers les résultats que l'amélioration des performances dynamiques (temps de montée, temps d'établissement,..) se fait au détriment du dépassement i.e. plus le système est rapide, grand est le dépassement.

1.4 Régulateur à action proportionnelle et dérivée PD

Considérons les deux situations suivantes "figure 1.9", où l'erreur $e(t_0)$ à la même amplitude, mais où :

- Elle croît dans le premier cas ;
- Elle décroît dans le second cas.

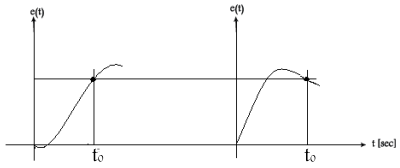


FIGURE 1.9 – Présentation de situations d'asservissement identiques en $t = t_0$ pour un régulateur D

On conçoit intuitivement qu'il serait illogique d'appliquer dans ces deux situations la même commande $U(t_0)$, bien que ce soit bel et bien l'action qu'entreprendrait un régulateur de type P

Il vient alors l'idée de formuler la commande $U(t_0)$ non pas en tenant compte exclusivement de l'amplitude de l'erreur (action P) à l'instant considéré $t = t_0$, mais aussi de son évolution, dans le but de savoir quelle est la tendance du signal d'erreur et d'en quelque sorte de la prévoir. Un bon moyen consiste à évaluer son taux de variation, à savoir sa pente en calculant la dérivée de l'erreur en $t = t_0$.

Pour ce faire, la dérivée par rapport au temps de/dt du signal d'erreur e(t) est calculée au moyen d'un bloc fonctionnel. Multipliée par un gain ajustable T_d afin de pouvoir doser son action, cette contribution est ensuite ajoutée à celle de l'action P.

La loi de commande résultante et bien celle du PD

$$u(t) = K_p \left\{ e(t) + T_d \cdot \frac{de}{dt} \right\} \tag{1.8}$$

La fonction de transfert du PD est :

$$G_c(s) = K_p \left\{ 1 + S \cdot T_d \right\} \tag{1.9}$$

Pour différents valeurs de k_d et pour $k_p = 15$, le graphe suivant de la figure 1.10, montre l'influence de l'action dérivée associée à l'action proportionnelle. Nous avons une diminution du taux de dépassement causée par l'action P sans diminuer la rapidité ou l'erreur ou l'erreur statique.

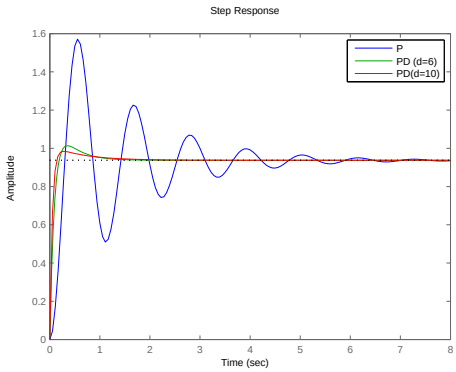


FIGURE 1.10 – Réponse indicielle du système asservi par le régulateur PD

Le tableau 1.2 donne une idée plus claire sur les réponses en termes de temps de réponse et du temps de montée.

k_d	$T_m(10 \Longrightarrow 90\% vf\ [sec])$	$T_r(\pm 5\% vf\ [sec])$	$D\%$
0	0.203	5.19	67.6
6	0.128	0.918	8.06
10	0.0902	0.898	4.99

TABLE 1.2 – Résultats de la simulation de l'asservissement PD

En plus du comportement moins oscillatoire du système asservi par le régulateur PD, on remarque que le système est plus rapide. On voit clairement que l'action dérivée apporte des améliorations sur les performances, car plus on augmente k_d plus le système est rapide et sans atteinte à la stabilité au régime établi.

1.5 Régulateur à action PID

Dans le but de tirer profil des avantages des régulateurs PI et PD vus précédemment, une combinaison entre les deux est possible.On parle alors de régulateur de type PID ; sa loi de commande :

$$u(t) = K_p \left\{ e(t) + \frac{1}{T_i} \int_{-\infty}^{t} e(\tau) d\tau + T_d \cdot \frac{de}{dt} \right\} \tag{1.10}$$

et sa fonction de transfert du régulateur PID

$$G_c(s) = k_p \cdot \frac{1 + sT_i + s^2 \cdot T_i \cdot T_d}{sT_i} \tag{1.11}$$

Il existe plusieurs méthodes de réglage des paramètres du régulateur PID, nous citerons que trois méthodes à savoir :*Ziegler et Nichols, Cohen et Coon* et *la méthode du modèle*

1.5.1 Méthodes de Ziegler et Nichols (ZN)

En 1942, Ziegler et Nichols [1] ont proposé deux approches heuristiques basées sur leur expérience et quelques simulations pour ajuster rapidement les paramètres des régulateurs P, PI et PID. La première méthode nécessite l'enregistrement de la réponse indicielle en boucle ouverte, alors que la deuxième demande d'amener le système bouclé à sa limite de stabilité.

Méthode de la réponse indicielle

Pour obtenir les paramètres du régulateur PID, il suffit d'enregistrer la réponse indicielle du processus seul puis de tracer la tangente au point d'inflexion de la courbe. On mesure ensuite sa pente P et le retard apparent L correspondant au point d'intersection de la tangente avec l'abscisse (figure 1.11)), On peut alors calculer les coefficients du régulateur choisi à l'aide du tableau 1.3.

type	k_p	T_i	T_d
P	$1/PL$	$\diagup$	$\diagup$
PI	$0.9/PL$	$3L$	$\diagup$
PID	$1.2/PL$	$2L$	$0.5L$

TABLE 1.3 – Valeur des paramètre PID dans le cas de la méthode de la réponse indicielle proposée par ZN

L'algorithme suivant résume les différentes étapes de la méthode :

Algorithm 1 Méthode de la réponse indicielle

Prélever à partir de la réponse indicielle
Le retard apparent : $L = t_1$
La constante de temps apparente : $T = t_2 - t_1$
La pente de la tangente au point d'inflexion :$P = \frac{y(\infty)}{T_3 - T_1}$
Le temps mort relatif : $\tau = \frac{L}{L+T} = \frac{t_1}{t_2}$

Comme exemple illustratif, nous considérons cette fois-ci la fonction de transfert utilisé dans [1] et qui égale à : $G_a = \frac{1}{(1+s)^3}$

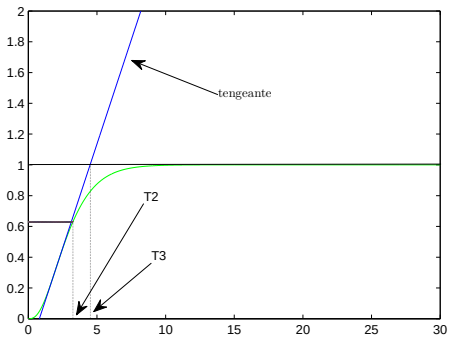


FIGURE 1.11 – Réponse indicielle en boucle ouverte de la fonction de transfert $G_a(s)$

$L\,[sec]$	$P\,[v/sec]$	$T\,[sec]$	$\tau\,[/]$
0.81	0.27026	2.44	0.2492

TABLE 1.4 – Résultats de la simulation (cas réponse indicielle)

A partir des tableaux 1.4 et 1.3 on a : $k_p = 5.6187$ $k_i = 3.4683$, et $k_d = 2.2756$

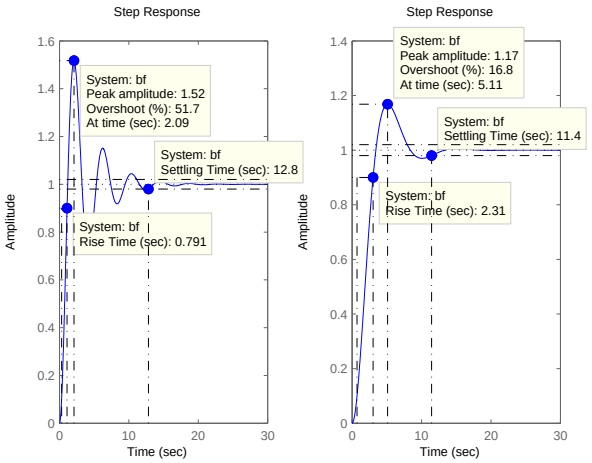


FIGURE 1.12 – Réponse indicielle en boucle fermée du système G avec ajustement (à droite) et sans réajustement (à gauche) des paramètres PID (Cas réponse indicielle)

Nous constatons que les valeurs proposées par Ziegler-Nichols causent un dépassement relativement grand 51% (figure 1.12 à gauche), la réduction du gain k_p par 5 à améliorer le dépassement 16,8% (figure 1.12 à droite) mais au détriment du temps de montée (2.31) sec au lieu de 0.791 sec.

Méthode du point critique

Cette méthode est basée sur la connaissance du point critique du processus. On boucle le processus sur un simple régulateur proportionnel dont on augmente le gain jusqu'à amener le système à osciller de manière permanente ; on se trouve ainsi à la limite de stabilité. Après avoir relevé le gain critique k_{cr} de régulateur et la période d'oscillation T_{cr} de la réponse, on peut calculer les paramètres du régulateur choisi à l'aide du "tableau 1.5" suivant

type	k_p	T_i	T_d
P	$0.5k_{cr}$	/	/
PI	$0.4k_{cr}$	$0.8T_{cr}$	/
PID	$0.6k_{cr}$	$0.5T_{cr}$	$0.25T_{cr}$

TABLE 1.5 – Valeur des Paramètre PID dans le cas de la méthode du point critique proposée par ZN

L'algorithme de cette méthode est donnée ci dessous

Algorithm 2 Méthode du point critique

Prélever à partir de la réponse fréquencielle :
la pulsation ω_π pour laquelle la phase vaut -180
le gain $G_\pi$correspondant à cette pulsation
Calculer à l'aide des résultats ci-dessus
Le gain critique : $k_{cr} = \frac{1}{G_\pi}$
La période d'oscillation critique : $T_cr = \frac{2\pi}{\omega_\pi}$

On reprends ici le même système utilisé précédemment (réponse indicielle) comme exemple illustratif de la méthode

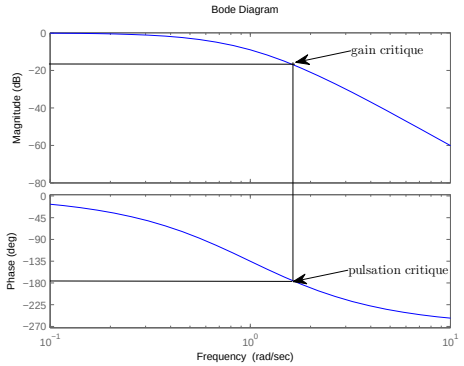


FIGURE 1.13 – Réponse fréquencielle

On prélève les valeurs des gains et pulsation critique à partir de la réponse fréquencielle, et on appliquant l'algorithme, on obtient les résultat suivant :

$G_{cr}[db]$	$\omega_{cr}[rad/sec]$	T_{cr}
18.1	1.73	8.3

TABLE 1.6 – Résultats de la simulation (cas méthode du point critique)

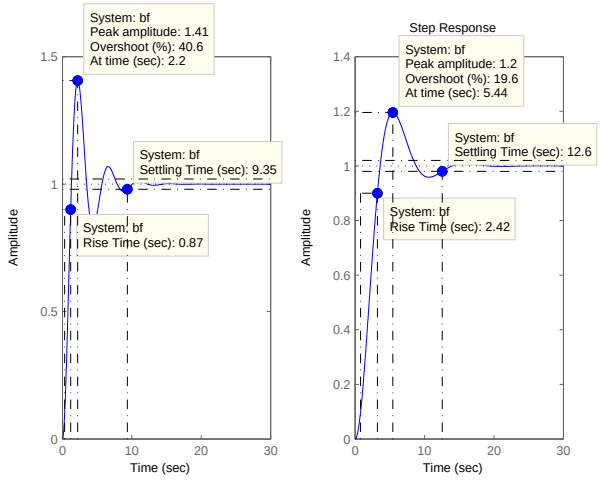


FIGURE 1.14 – Réponse indicielle en boucle fermée du système $G_a(s)$ avant et après ajustement des paramètres

Les paramètres obtenus avec cette méthode ont besoins d'un réajustement comme le cas de la méthode vue précédemment, la réponse représentée par la figure 1.14 gauche , est obtenue après avoir appliqué les règles du tableau 1.5 proposée par ZN ; on constate un grand dépassement (40.6%),après la diminution de K_p par un rapport de 3, la réponse à améliorée le dépassement(D%=19.5), mais le temps de montée à augmenté.

1.5.2 Méthode de COHEN et COON (CC)

La méthode de COHEN et COON [2]peut être utilisée sur des systèmes modélisables par un modèle de BROÏDA dont la fonction de transfert est :

$$G(p) = \frac{k}{1+\tau p} \cdot e^{-T_r p} \tag{1.12}$$

avec K est le gain statique, τ la constante de temps et T_r le retard.

$$\tau = 5.5(t_2 - t_1) \tag{1.13}$$

$$T_r = 2.8t_1 - 1.8t_2 \tag{1.14}$$

avec $t_1 = 28\%vf(y)$ et $t_2 = 40\%vf(y)$. y est la réponse indicielle du système
On note que la méthode ne s'applique que sur des systèmes stables car elle fait intervenir l'aspect statique du système. Le tableau suivant donne les réglages de COHEN et COON à partir des paramètres du modèle de Broïda et qui font intervenir le rapport $\mu = \frac{T_r}{\tau}$.

type	k_p	T_i	$k_d = T_d$
P	$\frac{1}{k\mu}(1 + \frac{\mu}{3})$	∞	0
PI	$\frac{1}{k\mu}(0.9 + \frac{\mu}{12})$	$T_r \cdot \frac{30+3\mu}{9+20\mu}$	0
PID	$\frac{1}{k\mu}(\frac{4}{3} + \frac{\mu}{4})$	$T_r \cdot \frac{32+6\mu}{13+8\mu}$	$T_r \cdot \frac{4}{11+2\mu}$

TABLE 1.7 – Valeur des Paramètre PID dans le cas de la méthode de COHEN et COON

On vas reprendre la fonction de transfert utilisée dans la méthode de Ziegler-Nichols, on calcule le modèle de Broïda, a savoir T_r et τ ainsi que le gain statique K

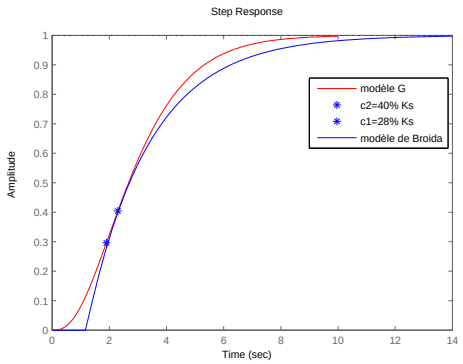


FIGURE 1.15 – Représentation par le modèle de Broïda

τ	T_r	μ
2.2	1.18	0.5364

TABLE 1.8 – résultats de la simulation

A partir du tableau 1.7 on obtient les paramètres du régulateur :$k_p = 2.7359\, k_i = 1.1383\, etk_d = 1.0696$

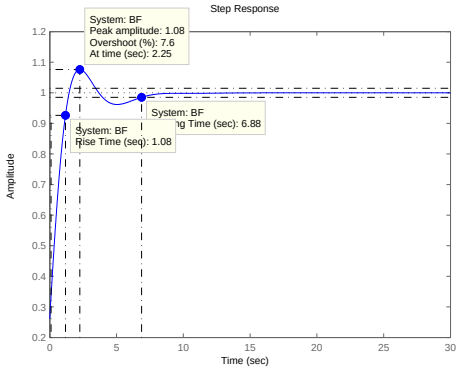


FIGURE 1.16 – Réponse indicielle en boucle fermée : méthode de COHEN et COON

La réponse obtenu est meilleur que celle de la méthode de Ziegler-Nichols.

1.5.3 Méthode du modèle : Orientation boucle fermée

Les méthodes directes (théoriques) sont très nombreuses et reposent sur la connaissance d'un modèle précis du système à commander. Les performances réelles obtenues dépendent de la qualité du modèle et de son aptitude à représenter le mieux possible le procédé.

Connaissant ce modèle, il est possible de définir les caractéristiques du régulateur qui permettra de contrôler au plus près le processus par une des méthodes directes de synthèse.

Parmi les méthodes directes, on présente ici la méthode du modèle[3] : elle est basée sur la donnée d'un modèle en boucle fermée à atteindre.

La fonction de transfert en boucle ouverte est :$H(s) = G_c(s).G_a(s)$ où $G_c(s)$ représente le régulateur et $G_a(s)$ représente le système a commandé.

La fonction de transfert en boucle fermée est :$F(s) = \frac{G_c.G_a}{1+G_c.G_a}$

Si la fonction de transfert en boucle fermée $F(s)$ est donnée, c'est-à-dire qu'elle à été élaborée de manière à répondre au cahier des charges, le régulateur R(p) est déterminé tout simplement par la relation suivante :

$$G_c(s) = \frac{F(s)}{G_a(s)(1 - F(s))} \tag{1.15}$$

Usuellement, le comportement souhaité en boucle fermée est celui d'un système d'ordre un ou d'ordre deux avec un gain statique unitaire, ce qui permet d'assurer une précision statique parfaite.

Une fois que la fonction de transfert en boucle fermée est établie, on détermine l'expression du régulateur $G_c(s)$ par la formule ci-dessus. On réorganise ensuite cette expression de manière à faire apparaître la structure du régulateur PID .

A titre d'exemple, on suppose que le système admet la fonction de transfert d'ordre 2 suivante : $G_a(s) = \frac{k_s}{(1+\tau_1 s)(1+\tau_2 s)}$; et on souhaite obtenir un comportement d'ordre 1 en boucle fermée avec un temps de réponse à 5% égal 0.6 s et une précision statique parfaite. Soit $F_d = \frac{1}{1+\tau s}$.
$\tau = 0.2\,[sec]$, puisque le temps de réponse à 5% d'un système d'ordre 1 est égal à trois fois sa constante de temps. Le gain statique doit être unitaire puisqu'on souhaite une précision statique parfaite.Soit donc $F_d = \frac{1}{1+0.2s}$
la fonction de transfert désirée en boucle fermée, où τ est choisi de manière à fixer le temps de réponse en boucle fermée. Le régulateur $G_c(s)$ est alors :$G_c(s) = \frac{1}{G_a(s)\tau s}$.

$G_c(s) = \frac{1}{k_s \tau}\left[\tau_1 + \tau_2 + \frac{1}{p} + \tau_1\tau_2 p\right]$.
ainsi on tire aisément les valeurs du régulateur PID parallèle :$k_p = \frac{\tau_1+\tau_2}{k_s\tau}$ $k_d = \frac{\tau_1\tau 2}{k_s\tau}$ et $k_i = \frac{1}{k_s\tau}$
On obtient $k_s = 1$ $\tau_1 = 0.5\,[sec]$ *et* $\tau_2 = 0.1\,[sec]$

si on remplace les valeurs de τ_1 et τ_2 dans $G_a(S)$ on auras : $G_a(s) = \frac{1}{(1+0.5s)(1+0.1s)}$ qui à comme réponse indicielle celle donnée par la figure 1.17 :

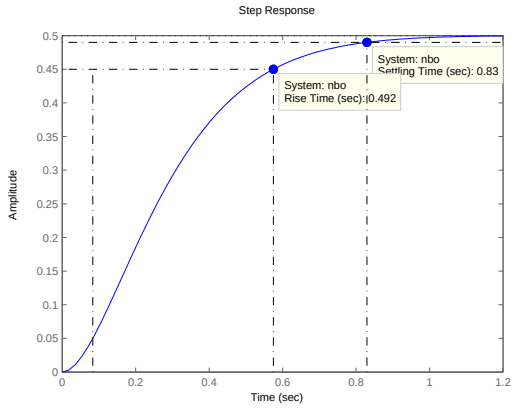


FIGURE 1.17 – réponse indicielle avec retour unitaire

On met en série avec $G_a(s)$ le régulateur $G_c(s)$($k_p = 15$ $k_i = 5$ et $k_d = 0.25$) on aura la réponse souhaitée dans la figure 1.18 suivante :

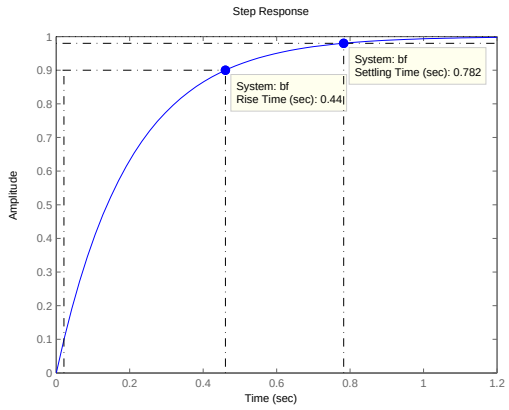


FIGURE 1.18 – Réponse indicielle de $G_a(s)$ en boucle fermée asservi par le régulateur PID

On tente cette fois ci de réguler le système de troisième ordre $G_a(s) = \frac{1}{(s+1)^3}$ on respectant le même cahier des charges. On applique la formule 1.15 on aura :$G_c(s) = \frac{(s+1)^3}{0.2s}$; on constate immédiatement l'iréalisabilité du régulateur physiquement car le degré du numérateur est supérieur à celui du dénominateur.

Par ailleurs, Il est à noter que R(p) s'obtient par inversion du modèle du procédé. Si on note $G_a(s) = \frac{N(s)}{D(s)}$, et $G_c(s) = \frac{F(s)}{G_a(s)(1-F(s))}$ on peut écrire

$$G_c(s) = \frac{D(s)}{N(s)} \frac{F(s)}{1 - F(s)} \tag{1.16}$$

On constate que les pôles et les zéros de $G_c(s)$ deviennent respectivement les zéros et les pôles de $G_c(s)$. Cette méthode ne convient pas par conséquent pour les systèmes pour lesquels la fonction de transfert G(p) possède un zéro instable

1.6 Conclusion

Les méthodes de réglage des paramètres PID vues dans ce chapitre ont des inconvénients qui laissent leur utilisation très restreinte ou avec précaution, car elles ne couvrent que certains processus à savoir la méthode Ziegler-Nichols qui ne s'applique qu'aux systèmes apériodique, et donne des résultats peux performants. L'automaticien doit réajuster les paramètres obtenus comme on l'a démontré à travers l'exemple 1.5.1 ; un inconvénient majeur est aussi dû au risque courus lors de l'application de cette méthode qui exige de ramener le système à la limite de stabilité pour tirer les paramètre du contrôleur PID. Aussi la méthode théorique du modèle, ne peut pas s'appliquée sur des systèmes qui ont un zéro instable. C'est le même problème qui revient avec la méthode de Cohen et Coon où elle s'applique que sur des systèmes stables.

Chapitre 2

Algorithme Génétique Simple

Dans ce deuxième chapitre nous aborderons le principe de base des algorithmes génétiques, utilisé comme méthode d'optimisation , nous allons voir comment l'adapter au problème d'optimisation de quelques fonctions test mono-objectif standards utilisées souvent dans la littérature, et voir quels serons les modifications possibles sur les parramètres d'un AG simple qu'on peut apporter pour qu'il sera plus performant. Aussi ce chapitre nous sert comme une base pour la bonne compréhension de la suite de ce mémoire.

2.1 Introduction

L'évolution biologique à engendrée depuis des millénaires des systèmes vivants, autonomes, aptes à résoudre des problèmes difficiles et capables de s'adapter à des environnements complexes, incertains et en constante transformation. La grande variété des situations auxquelles la vie s'est adaptée laisse penser que le processus de l'évolution est capable de résoudre de nombreuses classes de problèmes ; en d'autre mot, il se caractérise par sa robustesse. Les mécanismes de l'évolution reposent essentiellement sur la compétition qui sélectionne les individus les mieux adaptés à leur milieu en leur assurant une descendance sous forme d'une coopération mise en oeuvre par la reproduction sexuée. Les possibilités espérées de ces mécanismes ont conduit d'ès les années 1960 quelques chercheurs à vouloir les simuler afin de les appliquer à l'ingénierie. Trois écoles modélisant l'évolution d'une façon différente ont alors émergées indépendamment :

- Les algorithmes génétiques *(Genetic algorithm GA)* imaginé par *Holland*[4] ;
- La programmation évolutionnaire *(Evolutionary programing EP)* introduite par *Fogel* [5].
- Les stratégies d'évolution *(evolutionary Strategies ESs)* initiées par *Rechenberg et Chwefel* [6]

Ces trois approches font partie de la classe des Algorithmes Évolutionnistes, elles ne diffèrent que par l'absence ou la présence de tel ou tel opérateur. Bien que les buts originels de ces algorithmes est diffèrent, ils sont aujourd'hui utilisés pour accomplir des taches d'optimisation. Les variantes que connaissent actuellement les algorithmes génétiques recouvrent les principales caractéristiques qui différenciaient à l'origine ces différentes approches.

2.2 Algorithme génétique : Principe de base

Les caractéristiques héréditaires d'un être vivant dépendent exclusivement de son patrimoine génétique, ou genotype, constitué d'un ensemble de chromosomes formés de gênes. Ceci code des éléments du phenotype tel que par exemple la couleur des yeux. Ces gênes sont eux même formés de longues séquences spécifiques de quatre nucléotides et il existe une étape de décodage de genotype pour formé le phenotype. Dans le domaine des AG, les phénotypes sont les solutions plus ou moins optimales recherchées d'un problème particulier. La première étape de l'analogie génétique consiste donc à poser un ensemble d'individus, candidat à devenir la solution optimal du problème, où chaque individus i est constitué d'une chaîne de gênes mis bout a bout et codé d'une certaine façon(binaire, réel, etc). L'application de l'opération de décodage donne un ensemble de chromosomes qu'on appel aussi population. La transformation (genotype⇒ phenotype) se fait à travers l'application $F(X_i)$ ou F est le problème à optimiser représenté sous forme d'une fonction analytique et X_i est le chromosome i de la population.

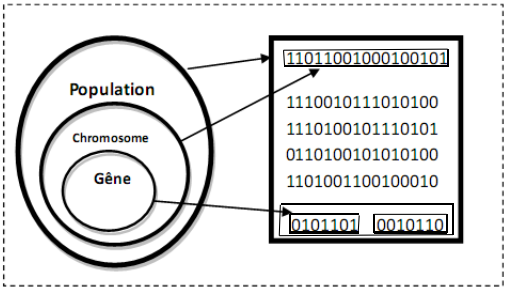


FIGURE 2.1 – Hiérarchie dans l'AG

Les individus évoluent ensuite simultanément au cours des itérations sous l'action d'opérateurs génétiques qui sont inspirés de la génétique naturelle.Ils sont dénommés opérateurs de reproduction, de recherche ou encore de recombinaison qui sont :

Le croisement : qui consiste à obtenir deux nouveaux individus enfants en combinant les chromosomes d'une paire d'individus parents.

La mutation : qui revient à changer avec une faible probabilité un ou plusieurs gênes d'un individu.

On procède ensuite à l'évaluation de la qualité de chaque solution potentielle. Cette étape traduit le fait que les individus les mieux adaptés au sein d'une population sont ceux qui satisfont au mieux la fonction objectif.

Le processus de sélection consiste enfin à choisir, parmi tout les éléments de la population, les individus les mieux adaptés afin de les reproduire. L'opération de sélection est donc une version artificielle de la sélection Darwinienne (dans les populations naturelles, seuls les individus les plus forts peuvent vivres jusqu'à l'âge adulte et ensuite se reproduire).

Après la création aléatoire d'une population initiale de N individus, l'algorithme génétique accomplit une génération lorsque les opérations définis ci-dessus sont appliqués successivement selon la séquence suivante illustrée par la figure 2.2.

Lors d'une génération, la population est remplacée soit par la totalité de ces descendent (algorithme generational) ou par une partie seulement de ceux-ci (algorithme stationnaire), néanmoins de génération en génération la taille de la population doit rester constante.

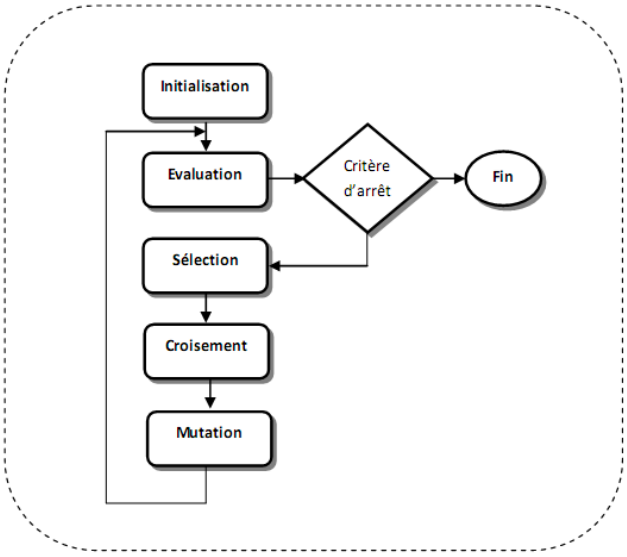


FIGURE 2.2 – Schéma de fonctionnement d'un algorithme génétique

Différent critères d'arrêt peuvent êtres choisis, l'AG est souvent arrêté lorsque l'on atteint un certains nombre de générations fixé apriori ou lorsque la population n'évolue plus assez rapidement. Les opérateurs évoqués dans l'organigramme forment un AG de base, ils peuvent êtres implémentés sous plusieurs formes :

2.2.1 Codage des paramètres

Selon Goldberg [7], l'utilisateur doit choisir le plus petit alphabet qui permet une expression naturelle des paramètres du problème (principe des alphabets minimaux). C'est pourquoi, l'alphabet binaire $\{0, 1\}$ est particulièrement bien adapté à la représentation des paramètres

Codage binaire standard

Pour chaque paramètre x_i situé dans l'intervalle $[x_i^{min}, x_i^{max}]$, on associe une chaîne binaire $b_0\, b_1\, b_2\, b_3\, b_4 ... b_{l-1}$ définie par l bits. A cette chaîne correspond une valeur entière naturelle :

$$N(x_i) = \sum_{i=0}^{l-1} 2^{l-1-i} \cdot b_i \tag{2.1}$$

Le paramètre réel x_i de l'espace de recherche relatif à $N(x_i)$ est obtenu par mise à l'échelle linéaire :

$$x_i = \frac{x_i^{max} - x_i^{min}}{2^l - 1} \cdot N_(x_i) + x_i^{min} \tag{2.2}$$

Cette méthode de codage est relativement facile à implanter mais elle présente l'inconvénient de limiter la précision des paramètres à une valeur ϵ_i correspondant à l'écart entre deux configurations réelles adjacentes obtenues, pour une variation du bit le moins significatif

$$\epsilon_i = \frac{x_i^{max} - x_i^{min}}{2^l - 1} \tag{2.3}$$

La longueur l de la sous-chaîne binaire nécessaire pour obtenir une précision ϵ_i pour le paramètre réel x_i correspond donc au plus petit entier naturel tel que :

$$l \geq \frac{1}{log2} log(\frac{x_i^{max} - x_i^{min}}{\epsilon_i} + 1) \tag{2.4}$$

La taille du chromosome est donc la somme des l_i.

Exemple :

Soient deux variables réelles et leur intervalle de définition respectivement$x_1 \in [-5, 5]$ et $x_2 \in [-10, 10]$. Si nous voulons une précision à la troisième décimale on aura pour chacune de ces variables des longueurs respectives de leurs gênes 14 et 18 (bits), ce qui fait que la longueur du chromosome égale à 32 bits.

codage Gray

Avec le codage binaire standard, deux configurations proches dans l'espace de représentation peuvent avoir des chromosomes très distincts. Par exemple, les chaînes "01111" et "10000"correspondent à deux configurations réelles voisines alors qu'elles diffèrent de 5 bits. Cette caractéristique peut s'avérer pénalisant pour la recherche locale. L'utilisation du code Gray est recommandé pour contourner ce problème. En effet, avec ce code, les entiers adjacents ne diffèrent que d'un bit, le passage entre deux configurations réelles voisines devient beaucoup plus facile puisqu'il suffit de modifier un seul bit dans le chromosome.

Codage réel

Le codage réel trouve ces origines dans les techniques de programmation évolutionnaire et les stratégies d'évolution.Il consiste à représenter chaque individu de la population par des nombres réels, il ne souffre plus des inconvénients du codage binaire, puisque l'espace de recherche est identique à celui de représentation. Enfin l'étape de décodage avant l'évaluation de la fonction objectif n'est pas nécessaire.

2.2.2 Initialisation

Les chromosomes des individus de la première génération sont habituellement initialiser de façon aléatoire, en respectant l'équiprobabilité d'obtenir une valeur égale à 0 ou 1 pour chaque bit.

2.2.3 Opérateur de sélection

Après avoir calculé la fonction objectif de chaque chromosome, on procède ensuite à la sélection des meilleur d'entre eux. Cette étape de sélection nécessite au préalable la définition de la fonction d'adaptation (fitness) f_{ad} qui à pour but de traduire la qualité de chaque chromosome de la population.

Sélection proportionnelle à l'adaptation

La sélection proportionnelle à l'adaptation introduite par Holland [4] est la plus connue, elle consiste à attribuer à chaque chromosome i une probabilité de sélections définie par :

$$P_s(i) = \frac{f_{ad}(i)}{\sum_{j=1}^{N} f_{ad}(j)} \tag{2.5}$$

où :
- $f_{ad}(i)$ désigne la fonction d'adaptation d'un chromosome i ;
- N est taille de la population.

Ensuite les parents sont choisis à l'aide de la roue de loterie pour laquelle chaque chromosome ou individus occupe une section dans la roue proportionnelle à sa fonction d'adaptation .
Un exemple de roulette de loterie biaisée relative à la population du tableau 2.1 est illustré dans la figure 2.3.

individu	F_{ad}	$P_s(i)$
i_1	50	0.5
i_2	30	0.3
i_3	20	0.2

TABLE 2.1 – Probabilité de sélection

Cette sélection, basée uniquement sur la performance, présente l'inconvénient de favoriser la prolifération de "super-individus" et de faire disparaître de nombreux genotypes dans la population. Ce qui cause une convergence prématurée de l'algorithme et

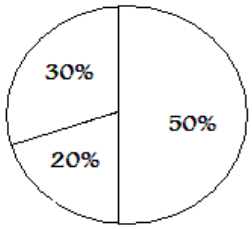


FIGURE 2.3 – Roue de loterie biaisée

de diminuer la diversité des individus. Aussi l'inconvénient majeur de cette technique est qu'elle nécessite une réécriture de la fonction d'adaptation selon le type d'optimisation (maximisation ou minimisation) et selon les valeurs (positives ou négatives) prise par la fonction objectif f_{obj}.Ainsi pour une maximisation, on prend :

$$f_{ad} = f_{obj} \; si \; f_{ad} \geq 0 \tag{2.6}$$

et

$$f_{ad} = \frac{1}{\|f_{obj}\|} \; si \; f_{ad} < 0 \tag{2.7}$$

Alors que pour une minimisation on utilise plutôt :

$$f_{ad} = \|f_{obj}\| \; si \; f_{ad} \leq 0 \tag{2.8}$$

et

$$f_{ad} = \frac{1}{\|f_{obj}\|} \; si \; f_{ad} > 0 \tag{2.9}$$

Sélection par tournoi

La sélection par tournoi [8] consiste à sélectionner m individus de façon aléatoire avec remise à chaque tour, le nombre de tour est égal au nombre d'individus à sélectionner. L'avantage de cette technique est que la fonction objectif peut être assimilée à la fonction d'adaptation.

Sélection par classement

La sélection par classement [9] consiste tout d'abord à trier les individus selon la valeur de leur fonction d'adaptation pour leur assigner ensuite un nombre R traduisant leur classement. La probabilité est alors calculée en fonction de ce classement et les parents candidats à la reproduction sont sélectionnés en utilisant la roue de loterie.
Il existe deux types de fonctions avec lesquels on calcule la probabilité de sélection.

- Linéaire :

$$P_s(i) = \frac{1}{N}\left\{S_p - 2(S_p - 1) \cdot \frac{R_i}{N}\right\} \tag{2.10}$$

- Non linéaire :

$$P_s(i) = \frac{S_p - 1}{1 - (2 - S_p)^N}(2 - S_p)^{R_i - 1} \tag{2.11}$$

où :

- N est la taille de la population candidate à la sélection ;
- $R(i)$ Correspond au classement de l'individu i ($R = 1$ est attribué au meilleur individu satisfaisant au mieux l'objectif).
- S_p désigne la " pression de sélection ", ce paramètre doit être fixé dans l'intervalle [1,2] ; son influence sur la probabilité de sélection des individus est illustrée par la figure 2.4

L'avantage essentiel de la sélection par classement est que la fonction objectif peut être directement assimilée à la fonction d'adaptation quelque soit le type d'optimisation et quelque soit les valeurs prisent par la fonction objectif. Par contre elle nécessite de fixer à priori le paramètre S_p, celui-ci doit être choisit judicieusement ; car il influa considérablement sur la vitesse de convergence ainsi que sur la qualité de l'optimum .

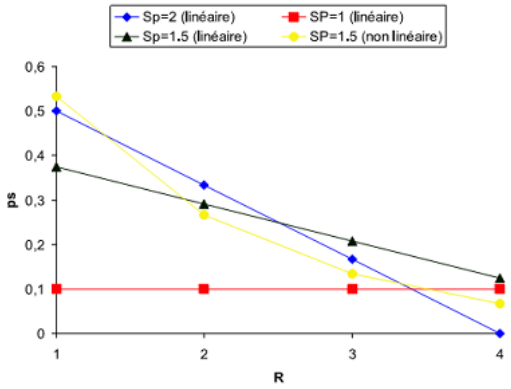


FIGURE 2.4 – Probabilité de sélection en fonction de la pression de sélection et du classement

Sélection par duplication

Pour nos applications dans la section 2.3 de ce chapitre, nous allons utiliser une méthode de sélection proche de celle de la roue de loterie, que nous avons développer dont le principe est le suivant :
après avoir évaluer la population,chaque chromosome est dupliquer proportionnellement à sa probabilité de sélection,puis triés d'une façon décroissante. En genère aléatoirement un chiffre entier ou sa valeur est entre [1, size(population)], qui correspond à l'indice du chromosome à sélectionner. L'algorithme suivant détaille les étapes

Algorithm 3 Sélection par duplication

Donnée : obj ;population ; nbr-ch=taille de "population"
fitness=abs(1/obj) ;
probabiliT=fitness./sum(fitness) ;
probabiliT=α(1./probabiliT) ;
duplication=arrondir(probabiliT) ;
[indice,duplication]=trier('duplication',décroissante)
aranger ("chromosome(i)",indice) ;
DUPLIQUE=dupliquer "chromosome(i)"selon"duplication(i)"
For $i \leftarrow 1$: nbr-ch
nouvel-indice=round(N-chr*rand)+0.5 ;
{Nouvelle-population}$\bigcup$ DUPLIQUE(nouvel-indice) ;
EndFor

où $\alpha = \frac{1}{\text{moyenne (probabiliT)}}$.

2.2.4 Opérateur de croisement

Le croisement [10] permet l'émergence d'une nouvelle population, à la recherche de nouveaux individus beaucoup plus robuste, en termes de fonction d'adaptation, rapprochant au mieux l'objectif (optimum). Il existe plusieurs types de croisement à savoir :

Croisement en un point

Ce type de croisement s'applique généralement dans le cas d'une représentation binaire des éléments de la population. Il consiste après avoir sélectionné deux parents à déterminer aléatoirement un entier $k \in [1, l-1]$ où l est la longueur des chromosomes.Les enfants sont alors créés en échangeant tout les caractères compris entre $k+1$ et l inclues

Exemple :
soient $k = 5$ et deux parents $P_{1,2}$:

P_1=1001010010
P_2=1101000100
Le croisement donne deux enfants $E_{1,2}$:
E_1=10010|00100
E_2=11010|10010
La figure 2.5 schématise le croisement en un point.

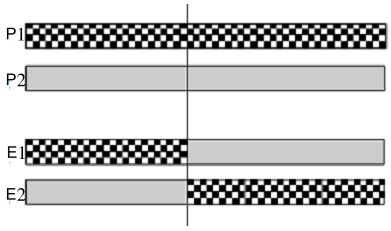


FIGURE 2.5 – Principe de croisement en un point

Croisement multi-points

Le croisement multi-points est une variante du croisement en un point, il consiste à effectuer la permutation en plusieurs points de la chaîne, dans un premier temps on détermine m entiers $k_i \in [1, l-1]$ correspondant en différents points de coupure et m+1 portions de chaque parent. Les enfants sont ensuite obtenus suite à l'échange de différentes portions.

Considérant l'exemple pris précédemment et soit $k_i = 3, 6, 8$:
P_1=100|101|00|10
P_2=110|100|01|00
ce qui nous donne les enfants suivant :
E_1=100|100|00|00
E_2=110|101|01|10

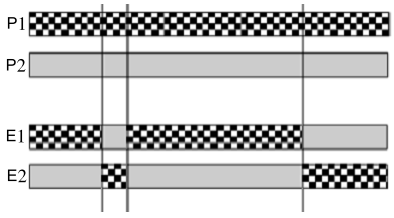


FIGURE 2.6 – Croisement multi-points

Croisement uniforme

Le croisement uniforme généralise le croisement multi-points vus précédemment, en offrant la possibilité à chacune des composantes de l'individu d'être un point de croisement éventuel.Il peut être employé dans le cas d'un codage binaire ou réel.

Ce schéma est basé sur la construction préalable d'un premier masque de croisement, composé de bits choisit aléatoirement et de longueur identique à celle des parents. La valeur prise par les bits de ce masque, pour la formation du premier enfant indique alors lequel des parents est candida .
Un second masque de croisement est alors construit par symétrie avec le premier, permet enfin de créer le second enfant.
Considérant à nouveau l'exemple des deux parents :
Déterminant aléatoirement le premier masque puis le second par symétrie :
M_1=0110001101
M_2=1001110010
S'il on décide que les bits 1,0 dans un masque correspondent respectivement aux parents P_1 et P_2, les deux enfants sont alors :
E_1=1101010110
E_2=1101000000

Considérant maintenant le cas d'un codage réel où les parents sont formés de trois variables de conception :
$P_1 = [-1, 6, 0.5]^T$
$P_1 = [-3, 4, 1.8]^T$
L'application des deux masques symétriques :
$M_1 = [001]^T$
$M_2 = [110]^T$

Permet de donner naissance aux deux enfants : $E_1 = [-3, 4, 0.5]^T$
$E_2 = [-1, 6, 1.8]^T$

la génération du deuxième masque aléatoirement et non par symétrie, s'appelle croisement discret. Il existe aussi d'autres schémas de croisement destinés uniquement dans le cas du codage réel tels que [9] :

- Recombinaison intermédiaire réelle ;
- Recombinaison linéaire étendue.

2.2.5 Opérateur de mutation

L'opérateur de mutation intervient directement après le processus de croisement, il s'applique sur la nouvelle population créée. Cet opérateur consiste à modifier avec

une probabilité très faible la valeur d'une composante d'un chromosome. Son rôle est d'explorer localement l'espace de recherche. On peut citer :

Mutation binaire

Dans le cas du codage binaire, chaque bit $a_i \in \{0, 1\}$ est remplacé par son complémentaire $a_i = 1 - a_i$. Dans l'exemple de la figure 2.7, une mutation à eu lieu sur le troisième gène du chromosome et elle à transformé ce gène de 1 en 0.[11]

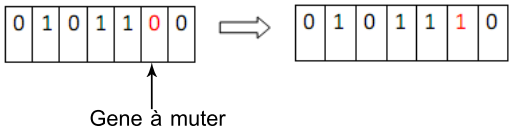


FIGURE 2.7 – Mutation binaire

Mutation non uniforme à codage réel

Dans le cas d'un codage réel, on utilise principalement la mutation non uniforme [11]. Nous supposons que le gène prend ses valeurs dans un intervalle $[x_k^{min}, x_k^{max}]$.
Le calcul de la nouvelle valeur d'un gène est un peu plus complexe. Le gène x_k subit des modifications importantes durant les premières génération puis graduellement décroissantes au fur et à mesure que l'on progresse dans le processus d'optimisation. Pour une génération t, on tire au sort une valeur binaire qui décidera si le changement doit être positif ou négatif. La nouvelle valeur $x_k^{'}$ du gène x_k est donnée par :

$$x_k^{'} = \begin{cases} x_k + \Delta(t, x_k^{max} - x_k) \; si \; rand = 0 \\ x_k - \Delta(t, x_k - x_k^{min}) \; si \; rand = 1 \end{cases} \tag{2.12}$$

où $\Delta(t, y)$ est une fonction qui définit l'écart entre la nouvelle valeur et la valeur initiale à la génération t et $rand$ est nombre aléatoire qui prend les valeurs 0 ou 1. Certains auteurs [12] proposent d'utiliser une fonction $\Delta(t, y)$ correspondante à une décroissance exponentielle de l'écart à travers les générations. Cette fonction est définie par :

$$\Delta(t, y) = y \times (1 - r^{(1 - \frac{t}{T})^{\beta}}) \tag{2.13}$$

Où :
- T est l'indice de génération pour laquelle l'amplitude de la mutation s'annule ;
- β est un paramètre de l'opérateur de mutation (souvent $\beta = 5$).

– r est un nombre produit aléatoirement dans l'intervalle [0,1].
– t est le numéro de la génération.

2.3 Fonctions-test mono-objectif

2.3.1 Fonction f_1 de Dejong :

La courbe de cette fonction est illustrée par la figure 2.8,elle est multi-variables, convexe et uni-modale [25]. Son expression est donnée par :

$$f_1 = \sum_{i=1}^{n} x_i^2 \tag{2.14}$$

tel que $x_i \in [-5.12; 5.12]$, $i = 1, \ldots, m$.
Elle présente un minimum global $\min(f_1) = f_1(0) = 0$

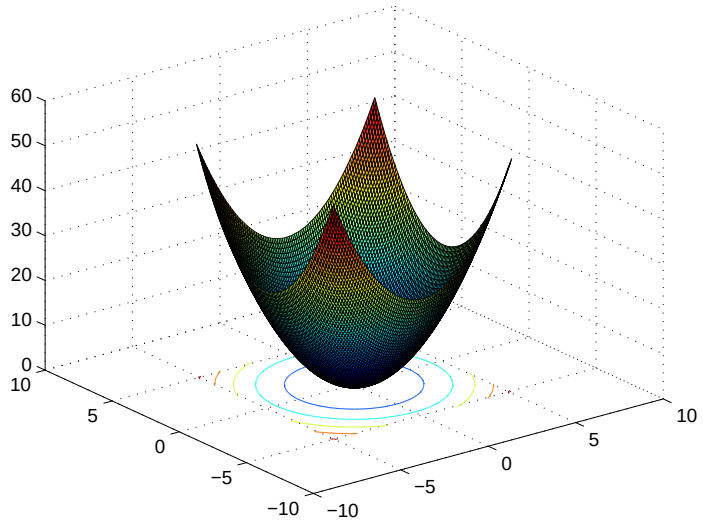


FIGURE 2.8 – Fonction f_1 de Dejong ; m=2 ;

2.3.2 fonction f_2 de Rastrigin :

Illustrée par la figure 2.9, cette fonction est multi-variable, convexe et multi-modale et définie par [25] :

Elle présente un minimum global : $\min(f_2) = f_2(0) = 0$

$$f_1 = n + \sum_{i=1}^{n} x_i^2 - \cos(2\pi x_i) \tag{2.15}$$

tel que $x_i \in [-5.12; 5.12]$, $i = 1, \ldots, m$

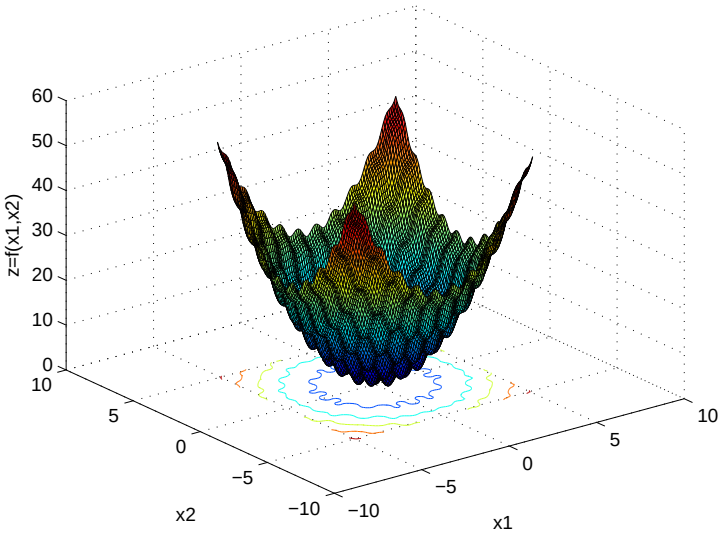


FIGURE 2.9 – Fonction f_2 de Rastrigin ; n=2 ;

Cette fonction est basée sur la fonction $f_1$de DeJong à laquelle on ajoute une modulation de cosinus pour produire de multiple minima locaux.

2.3.3 Fonction f_3 de Easom :

Cette fonction est illustrée à la figure 2.10, elle présente les mêmes propriétés que la fonction de Dejong(multi-variable,convexe, uni-modale), elle est définie par [25] :

$$f_{Eas} = -\cos(x_1).\cos(x_2).e^{-((x_1-\pi)^2+(x_2-\pi)^2)} \tag{2.16}$$

tel que $x_i \in [-10; 10]$, $i = 1, \ldots, m$

Elle présente un minimum global : $\min(f_3) = f_3(\pi, \pi) = -1$

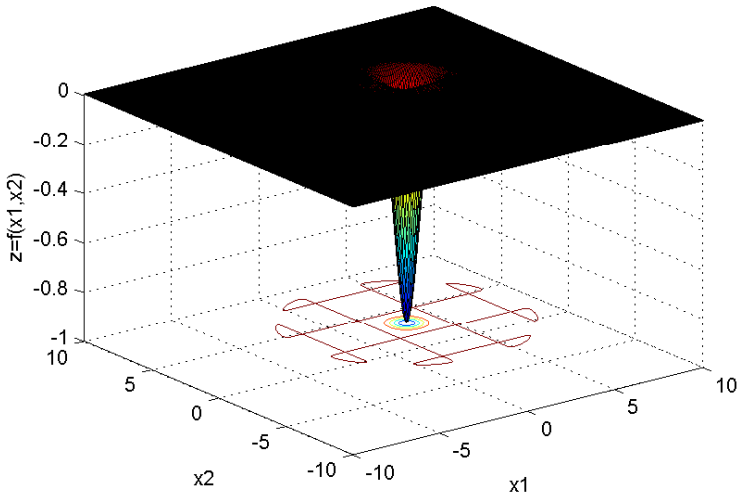


FIGURE 2.10 – Fonction f_3 de Easom ; n=2 ;

2.3.4 Paramètres de L'AG

Le codage utilisé est de type binaire. On prend pour un premier temps la précision $\epsilon = 10^4$, puis $\epsilon = 10^{-7}$ (voir tableaux 2.2 et 2.3), l'application de l'équation 2.4 donne des longueurs L différentes selon la fonction-test ainsi que la précision prise. La taille de la population égale à 20, générée d'une manière aléatoire

La sélection est de type proportionnelle, et comme il s'agit du problème de minimisation, nous utilisons l'inverse de la valeur de l'objectif IAE comme fonction d'adaptation i.e.$f_{ad} = \frac{1}{f_{obj}}$.

Le croisement est de type multi-points, pour explorer au mieux l'espace de recherche, avec pc=0.65 et la mutation est de type binaire simple avec pm=0.35.

Un archive "élite" est mis en place pour contenir le meilleur chromosome de chaque génération, pour en suite tirer le meilleur chromosome de cet archive à la fin de la simulation.

Résultat :
Pour chaque test, l'optimum est identifié (voire tableau 2.2 et 2.3) après un nombre raisonnable d'itération (200 étirations), et à l'aide d'une population contenant peut d'individus (nbr-ch=20). Notre algorithme démontre ici sa capacité à localiser un optimum "caché" (f_1), ou "perdu" dans une zone étroite par rapport à un large domaine de définition (f_3). Il prouve en outre son aptitude à s'extraire de minima locaux pour localiser efficacement l'optimum global (f_2)

fonction-test	x_1	x_2	f
f_1	$0.4883.10^{-5}$	$0.4883.10^{-5}$	$0.4768.10^{-10}$
f_{Rast}	$0.4883.10^{-5}$	$0.4883.10^{-5}$	$0.9889.10^{-09}$
f_{Eas}	3.1189	3.0253	-0.9791

TABLE 2.2 – Résultats de la simulation pour $\epsilon = 10^{-4}$

fonction-test	x_1	x_2	f
f_1	$-0.4768.10^{-8}$	$-0.4768.10^{-8}$	$0.8882.10^{-15}$
f_{Rast}	$-0.4768.10^{-8}$	$-0.4768.10^{-8}$	$0.8882.10^{-15}$
f_{Eas}	3.1240	3.1240	-0.9991

TABLE 2.3 – Résultat de la simulation pour $\epsilon = 10^{-7}$

La figure ci-dessous nous renseigne sur la vitesse de convergence de l'algorithme vers l'optimum global ; et on constate que l'algorithme "perd son patrimoine génétique" ; ce qui nous laisse penser à l'élitisme comme archive qui, non seulement pour contenir le meilleur chromosome de chaque génération et en suite tirer le meilleur chromosome de cet archive à la fin de la simulation, mais un archive qui participe à la sélection et permettre à l'algorithme de préserver le meilleur chromosome des générations antérieurs.

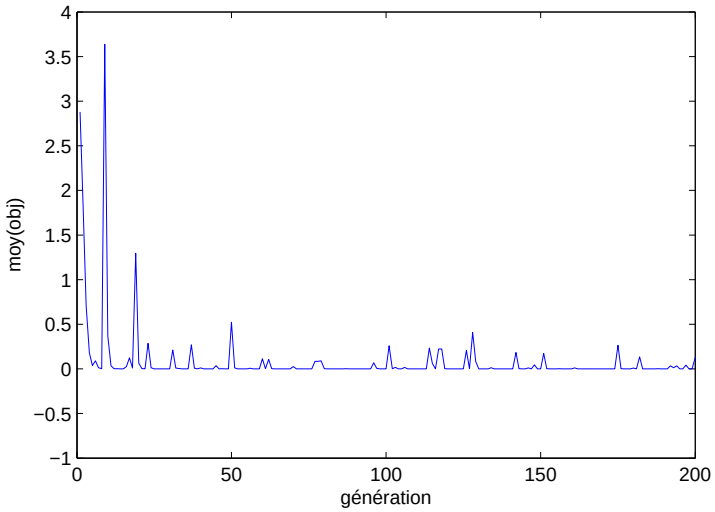


FIGURE 2.11 – Convergence de l'algorithme vers l'optimum global dans le cas de f_1

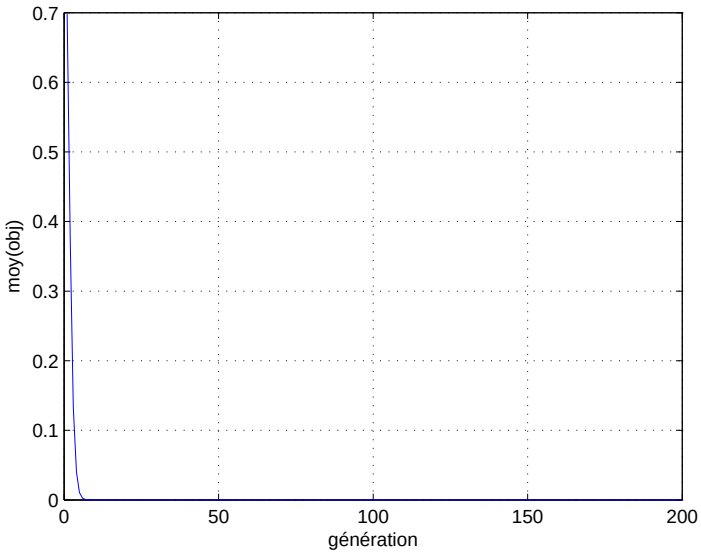


FIGURE 2.12 – Convergence de l'algorithme vers l'optimum global dans le cas de $f_1$avec élite

On vois que l'algorithme à préserver le chromosome "solution" depuis la dixième génération.

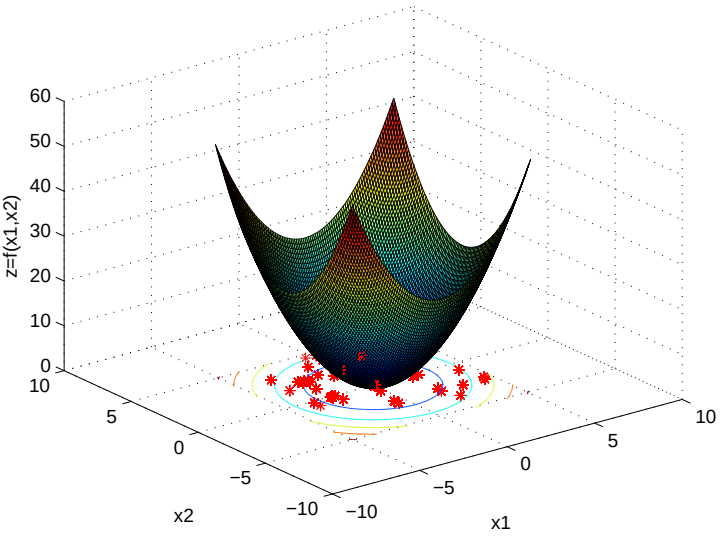


FIGURE 2.13 – Population à la cinquième génération

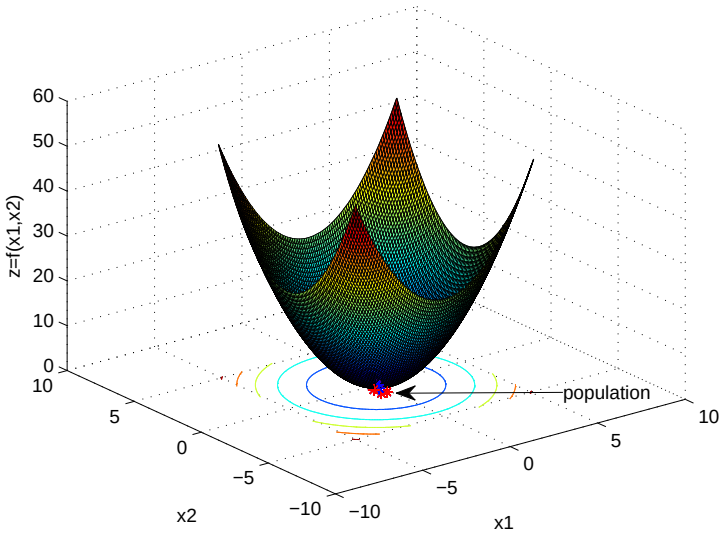


FIGURE 2.14 – Population à la fin de simulation :f_1

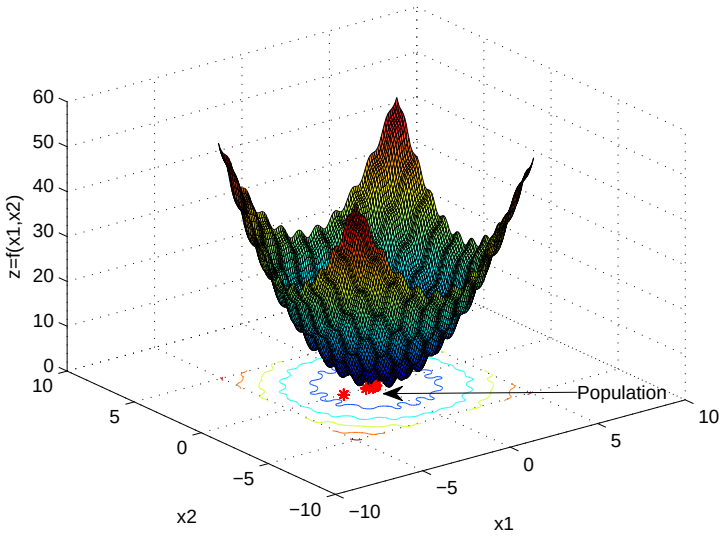


FIGURE 2.15 – Population à la fin de simulation :f_{Rast}

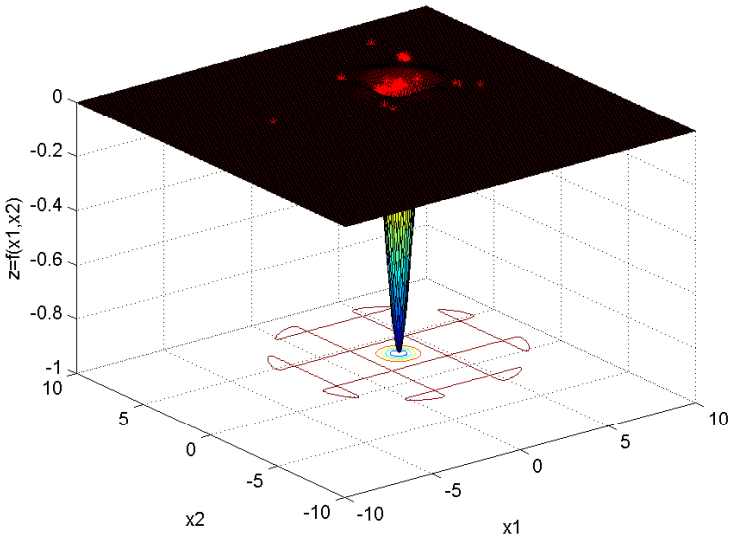


FIGURE 2.16 – Population à la fin de simulation : f_{Eas}

2.4 Conclusion

Dans ce chapitre, nous avons décrit le fonctionnement et les différents opérateurs d'un algorithme génétique, technique d'optimisation mono-objectif. Pour ne pas rester dans un cadre descriptif, des exemples d'optimisation ont été traités à la fin du chapitre. les résultats obtenus ont permis d'une part de montrer qu'un AG est capable de localiser des optimums dans des espaces de recherche divers et d'autres part de valider les algorithmes écrits sous matlab.
Dans le chapitre suivant, nous allons décrire le problème d'optimisation multi-objectifs et quelques algorithmes génétiques de résolution rapportés dans la littérature.

Chapitre 3

Algorithme génétique Multi-Objectifs

Après avoir décrit dans le chapitre précédent le fonctionnement d'un AG simple, nous aborderons ici l'optimisation multi-objectifs : formalisme mathématique et principe de base ; puis nous allons voir quelques méthodes usuelles où nous détaillerons deux méthodes appartenant aux deux grandes familles des techniques développées, à savoir la méthode d'agrégation pondérée(technique à priori) ainsi que la méthode SPEA (technique a posteriori).

3.1 Introduction

L'optimisation des problèmes réels implique souvent non pas un, mais plusieurs critères qui doivent êtres satisfaits simultanément. La solution d'un tel problème n'est donc plus unique mais multiples, chacune des solutions représente en effet un compromis acceptable entre les différents objectifs contradictoires. La résolution d'un problème multi objectif (PMO) s'effectue à l'aide d'un processus comprenant :

- Une étape de recherche au cours de laquelle un ensemble de solutions optimales est déterminées ;
- Une étape de décision permettant au concepteur de sélectionner une solution optimale offrant le compromis souhaité.

Selon l'ordre dans lequel ces deux étapes sont envisagées, on distingue trois méthodes de résolution :

- Les techniques a priori ($d\acute{e}cision \Longrightarrow recherche$) : à l'aide d'indication préalablement données par le concepteur, l'optimisation consiste à résoudre un problème mono objectif dérivé du problème multi objectif initial.
- Les techniques a posteriori ($recherche \Longrightarrow d\acute{e}cision$) : l'optimisation multi objectif est réalisée jusqu'à son terme sans informations préliminaire. Les résultats de la recherche est donc un ensemble de solution optimales parmi lesquels le concepteur effectue finalement son choix.
- Les techniques interactives ($recherche \Longleftrightarrow d\acute{e}cision$) : après chaque boucle d'optimisation, le concepteur modifie ces choix au vue des solutions candidates qui lui sont proposées. Cette technique bien qu'appliquant une approche originale, présente l'inconvénient de monopoliser l'intension du concepteur tout au long de la recherche. A cet effet, elles ne sont pas utilisées dans le domaine de l'ingénierie, donc elle ne sera pas détaillée dans le cadre de ce travail.

3.2 Formalisme mathématique

Pour des raisons de simplicité, nous supposons que toutes les fonctions objectives sont à minimiser. Le problème d'optimisation multi objectif peut être posé sous la forme suivante :

$$\begin{array}{l} \text{déterminer } x^* \\ \text{optimisant} f_m(x) \text{ tel que } m = 1, \cdots M \\ \text{sous} \left\{ \begin{array}{l} h_j(x) = 0 \text{ tel que } j = 1, \cdots k \\ g_i(x) \leq 0 \text{ tel que } i = 1, \cdots L \\ x_i^{min} \leq x \leq x_i^{max} \end{array} \right. \end{array}$$

L'application de la fonction vectorielle

$$f(x) = [f_1(x), \cdots f_M(x)]^T$$

à l'ensemble des configuration admissible

$$Q_{ad} = \left\{ x \in \Re^n | x_i^{min} \leq x \leq x_i^{max},\ h_j(x) = 0 \text{ et } g_i(x) \leq 0 \right\}$$

forme alors l'ensemble des objectifs réalisables

$$F_{ad} = f(Q_{ad})$$

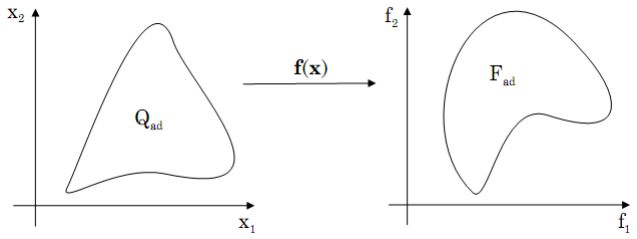


FIGURE 3.1 – Q_{ad} *et* f_{ad}

3.3 Optimalité de Pareto

La notion d'optimalité de Pareto à été formuler à la fin du XIX siècle et constitue l'origine de la recherche dans le domaine de l'optimisation multi-objectifs.afin d'assurer une consistance dans nos notation, on définie [13] dans ce qui suit la notion d'optimum et de dominance de Pareto.

Définition 1 (Dominance de Paréto)**.** *Le vecteur* $f^a = [f_1^a, \cdots f_M^a]^T$ *domine le vecteur* $f^b = [f_1^b, \cdots f_M^b]^T$ *si et seulement si* f^a *est partiellement inférieur* $(\prec_p)$ *à* f^b

$$\forall k \in \{1, \cdots M\} : f_k^a \leq f_k^b \wedge \exists k \in \{1, \cdots M\}\ f_k^a < f_k^b \tag{3.1}$$

Définition 2 (optimum de Pareto)**.** *Une solution* x^* *est un optimum de Pareto si et seulement s'il n'existe pas d'autres solutions admissibles* $x \in Q_{ad}$ *pour laquelle* $f(x)$ *domine* $f(x^*)$*.*

En d'autre terme, une configuration x^* est optimale au sens de Pareto lorsqu'il n'existe pas d'autres configurations admissible qui améliorent simultanément tout les objectifs par rapport aux valeurs obtenues par x^*.

L'ensemble des solutions x^* ainsi définies (appelées parfois solutions efficaces ou non-inférieurs) forment l'ensemble de Pareto $Q_p \subset Q_{ad}$:

$$Q_p = \{x^* \in Q_{ad} \forall x \in Q_{ad} : f(x^*) <_p f(x)\} \tag{3.2}$$

La fonction vectorielle $f(x)$ appliquée à l'ensemble de Pareto détermine l'ensemble des solutions non dominées appelé généralement front de Pareto .

$$F_p \subset f(Q_p) \tag{3.3}$$

$$F_p = \{f(x^*) \in F_{ad} | f(x^*) \text{ est non dominé}\} \tag{3.4}$$

Ces différentes notions sont illustrées par la figure 3.2 dans le cas de deux fonctions objectifs.

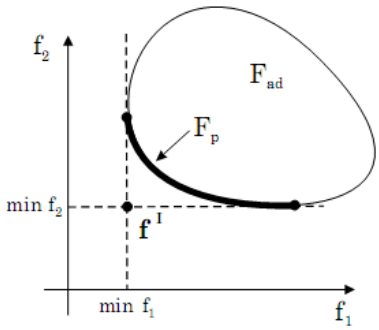


FIGURE 3.2 – Exemple d'un front de Pareto

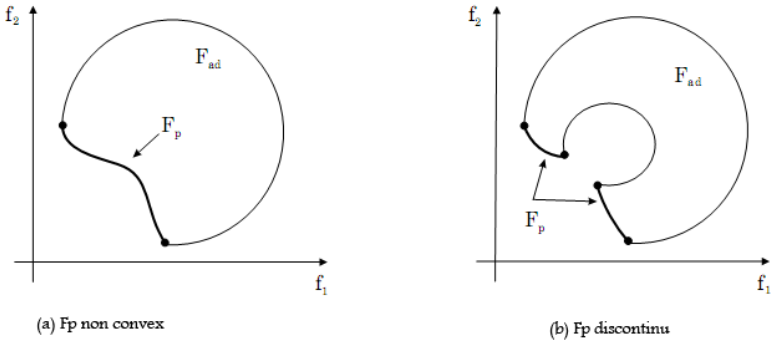


FIGURE 3.3 – Irrégularités des fronts de Pareto

Remarquant que l'ensemble des solution non dominées ne possède a priori aucune propriété de régularité. En effet, un problème multi-objectif régulier ayant à la fois un espace de conception convexe et des fonctions objectifs continues, peux parfaitement générer un front de Pareto non convexe "figure 3.3 (a)" ou discontinu "figure 3.3(b)".

3.4 Convergence et Diversité

L'approximation de front de Pareto par un algorithme évolutionnaire quelconque doit satisfaire deux objectifs simultanément, qui sont d'ailleurs contradictoires, à savoir, trouver **toutes** les solutions x^* du **front global**, pour se faire, des méthodes de diversité ont été imaginées. On site parmi elles.

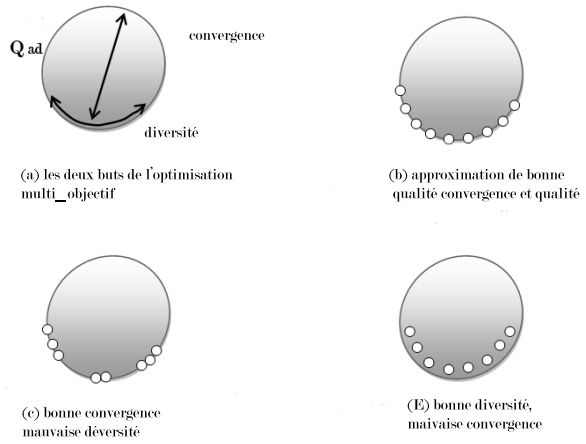


FIGURE 3.4 – Dualité convergence,diversité

3.4.1 La fonction de partage *(sharing)*

Le sharing [11] consiste à ajuster la fitness des individus pour éviter qu'ils se concentrent dans une niche principale. La technique de partage de la fitness *(fitness sharing)*, introduite par Goldberg et Richardson, réduit la fitness de chaque individu d'un facteur correspondant environ au taux d'agrégation de la population autour de son voisinage :

$$f'_{ad} = \frac{f_{ad}}{m_i} \tag{3.5}$$

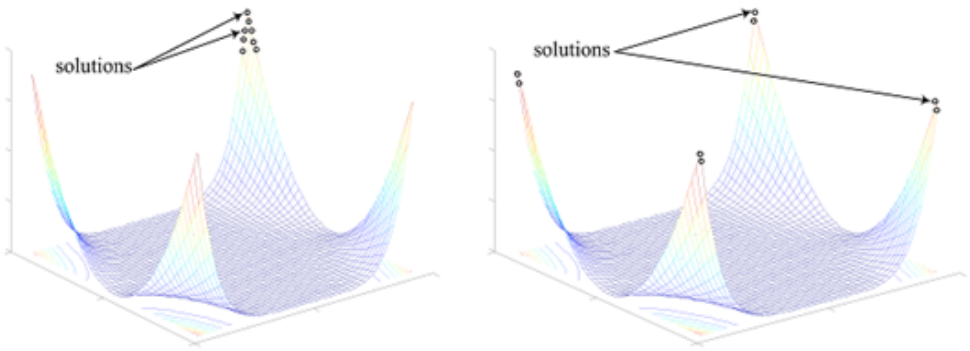
Où le compteur de niche m_i se calcule de la manière suivante :

$$m_i = \sum_{j=1}^{N} sh(d_{ij}) \tag{3.6}$$

Avec N désigne la taille de la population et sh mesure la similarité entre deux individus i et j en fonction de la distance d et le rayon de niche σ :

$$sh_d = \begin{cases} 1 - (\frac{d_{ij}}{\alpha_{shar}})^{\alpha} \; si \; d_{ij} < \alpha_{shar} \\ 0 \; sinon \end{cases} \tag{3.7}$$

La figure 3.5 montre deux exemples de répartition de populations dans le cas d'une fonction multi-modale : le premier sans sharing et le deuxième avec sharing.



(a) sans shanring (b) avec sharing

FIGURE 3.5 – Partage de la population

3.4.2 Méthode du plus proche voisin

Dans les méthodes du plus proche voisin, la distance entre une solution et son $k^{ième}$ plus proche voisin est prise en compte pour estimer la densité d'une solution. Dans SPEA2 [14], l'estimateur de densité est basé sur l'inverse de cette distance. La préservation de diversité opérée au sein de NSGA2 [13] est quand à elle est basée sur la distance de crowding proposé par Holland pour éviter une certaine dérive génétique. Cette mesure estime la densité d'une solution par rapport au volume de l'hyper-rectangle défini par ces plus proches voisins directs, une valeur de diversité infinie étant affectée aux solutions extrêmes.

3.4.3 L'élitisme

L'élitisme à pour but principal de préserver les solutions non-dominées trouvées par l'algorithme au cours de la recherche.

3.5 Technique d'optimisation

Comme en introduction, on peut classer les méthodes d'optimisation multi-objectifs en trois familles. Nous nous limitons notre exposé aux deux familles les plus utilisées à savoir :

3.5.1 Méthode a priori

A l'aide d'indications préalablement données par le concepteur, l'optimisation consiste à résoudre un problème mono objectif dérivé du problème multi objectif initial. On distingue plusieurs méthodes à savoir :

- Méthode des distances ;
- Méthode des contraintes ;
- Méthode de l'ordonnancement lexicographique ;
- Méthode d'agrégation pondérée.

Méthode d'agrégation pondérée

C'est l'une des premières méthodes utilisée pour résoudre les PMO. Elle consiste à transformer le problème MO en un problème mono objectif en combinant les composantes f_i du vecteur objectif du problème en une seule fonction scalaire f. Il existe dans la pratique, différentes façons de construire la fonction f. La plus classique et la plus utilisée se ramène à une simple somme pondérée des objectifs f_i (agrégation additive)[15].

$$f = \sum_{i=1}^{i} \omega_i \times f_i \tag{3.8}$$

Les paramètres ω_i sont les poids de pondération. Comme le montre la figure 3.6 ; le choix de différentes valeurs de ω_i permet au concepteur de déterminer plusieurs solutions non dominées.

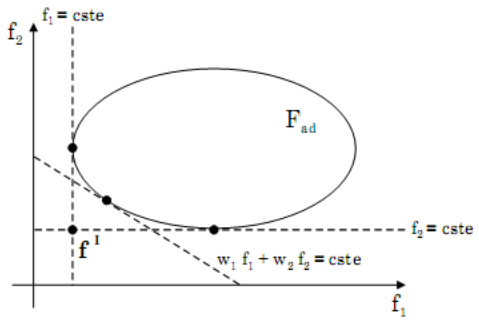


FIGURE 3.6 – Combinaison linéaire

3.5.2 Méthode à posteriori

Toutefois, l'approche précédente n'est généralement pas satisfaisante car le nombre d'exécutions successives nécessaires pour déterminer les différents compromis conduit

à un nombre d'évaluations de critères prohibitif. Ainsi, pour surmonter cette difficulté, on préfère utiliser des méthodes permettant de trouver l'ensemble de solutions Paréto optimales en une seule exécution. Parmi ces méthodes on peut citer :

VEGA (*Vector evaluated genetic algorithm*)

Le VEGA [16] proposé par Schaffer (1985), à été la première méthode non agregative utilisant les algorithmes génétiques pour résoudre un problème d'optimisation multi-objectif. Cet algorithme considère une population de N individus. A chaque génération, la population est divisée en un nombre de sous populations égal au nombre d'objectifs. Chaque sous population i est sélectionnée en considérant un seul objectif f_i. Ensuite, ces sous populations sont regroupées afin d'obtenir une nouvelle population de N individus et les opérateurs de croisement et de mutation sont appliqués.

L'avantage de cet algorithme est qu'il est facile à implementer et à combiner avec n'importe quel mode de sélection (tournoi, roulette, rang), mais son inconvénient majeur est qu'il a tendance à générer des solutions qui excellent dans un seul objectif, sans tenir compte des autres objectifs (points extrêmes du front). Toutes les solutions de performance moyenne (ne possédant aucun objectif fort) et qui peuvent être de bons compromis, risquent de disparaître avec ce type de sélection.

MOGA (*Multiple Objectives Genetic Algorithm*)

Cet algorithme , proposé par Fonseca et Fleming (1993)[17], utilise la notion de dominance pour ranger les individus de la population. Il diffère de l'algorithme génétique standard uniquement dans la manière dont la fitness est assignée pour chaque solution. Pour démarrer l'algorithme, les relations de domination sont d'abord calculées pour chaque solution. Puis, pour une solution i, un rang égal à un plus le nombre de solutions n_i qui dominent la solution i est attribué. Une fitness est ensuite attribuée à chaque solution en fonction de son rang, les individus avec les rangs les plus faibles ayant les meilleures fitness. Afin de maintenir la diversité entre les solutions non dominées, les auteurs utilisent une fonction de partage (*Sharing*). La méthode permet d'obtenir des solutions de bonne qualité et s'implante facilement. Toutefois, les performances sont très dépendantes de la valeur du paramètre σ_{shar} utilisé dans le sharing.

NPGA (*Nished Pareto Genetic Algorithm*)

Cette méthode [18] proposée par Horn et Napfiotis (1994) utilise une sélection par tournoi en se basant sur la notion de dominance de Paréto. Le NPGA exécute les mêmes étapes que l'AG standard, la seule chose qui diffère étant la méthode de sélection. A chaque tournoi, deux individus candidats A et B sont pris au hasard dans la population courante. Au lieu de limiter la comparaison aux deux individus (comme c'est le cas pour l'AG standard), une sous population (ou ensemble de comparaison) de taille t_{dom} est également choisie au hasard. Les deux candidats sélectionnés sont comparés

à chaque individu du sous-groupe. Si l'un des candidats est dominé par l'ensemble de comparaison et le second ne l'est pas, ce dernier est alors positionné dans la population suivante. Dans les autres cas, une fonction de partage est appliquée pour choisir le candidat gagnant. Le paramètre t_{dom} permet de contrôler la pression de sélection ou de dominance. L'algorithme NPGA est considéré comme étant l'algorithme le plus rapide parmi les approches précédentes car à chaque génération la comparaison n'est appliquée que sur une portion de la population. Le principal inconvénient de cet algorithme est qu'il nécessite, en plus de spécifier le paramètre de sharing σ_{shar}, un autre paramètre supplémentaire qui est la taille du tournoi t_{dom}

NSGA II (*Non dominated Sorting Genetic Algorithm II*)

En proposant le NSGA II [19], Deb et al ont tenté de résoudre toutes les critiques faites sur NSGA : non élitiste, complexité de calcul et utilisation de sharing qui implique le réglage d'un ou plusieurs paramètres. Dans cet algorithme, à chaque génération t une population de parents (P_t) de taille N et une population d'enfants (Q_t) de même taille sont assemblées pour former une population (R_t)de taille $2N$, comme indiqué sur la figure 3.8. Cet assemblage permet d'assurer l'élitisme. La population(R_t) est ensuite répartie en plusieurs fronts $(F_1, F_2, \cdots)$ par une procédure de tri, plus rapide que celle proposée dans la première version de NSGA. Une nouvelle population parent (P_{t+1}) est formée en ajoutant les fronts au complet (premier front F_1, second front F_2, etc$\cdots$) tant que ceux-ci ne dépassent pas N. Si le nombre d'individus présents dans (P_{t+1})est supérieur à N, une procédure de crowding est appliquée sur le premier front suivant F_i non inclus dans (P_{t+1}). Le but de cet opérateur est d'insérer les $(N - (P_{t+1}))$ meilleurs individus de F_i qui manquent dans la population (P_{t+1}). Une fois que les individus de la population (P_{t+1}) sont identifiés, une nouvelle population enfant (Q_{t+1})est créée par **sélection, croisement** et **mutation**. La sélection par tournoi est utilisée mais le critère de sélection est maintenant basé sur l'opérateur de comparaison ($\preccurlyeq_n$) défini ci-dessous. Le processus se répète d'une génération à une autre jusqu'à satisfaction d'un critère d'arrêt.

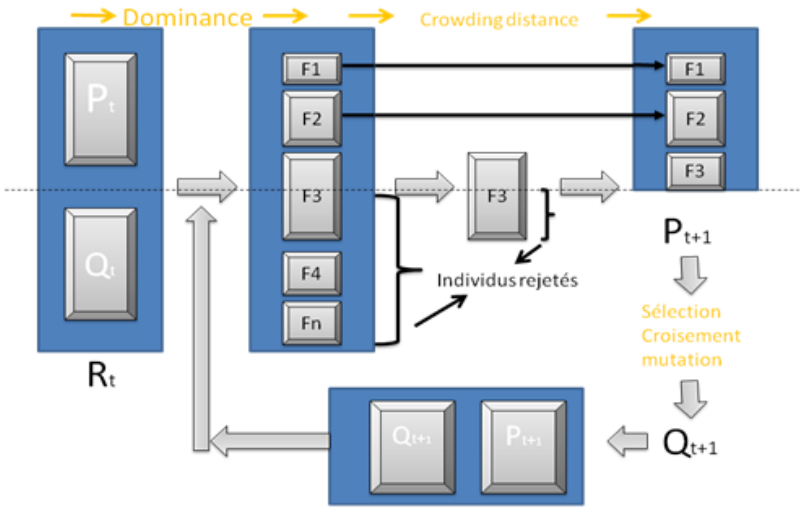


FIGURE 3.7 – Diagramme de NSGA II

Procédure de tri (*non-dominated sort*) La répartition de la population en plusieurs fronts s'effectue de la manière suivante :

Algorithm 4 Tri

Donnée : R
Début
  Pour tout individus p de R **faire**
    Initialiser $S_p = \varnothing$ (ensemble qui contient les individus dominés par p)
    Initialiser n_p (nombre d'individus qui domine p)
    Pour touts individu q de R **faire**
      si $p \prec q$ **alors**
        $S_p \leftarrow q$
      sinon
        $n_p = n_p + 1$
      fin Si
    fin Pour
    si $n_p = 0$ **alors**
      $F_i \leftarrow p$ (p sera au premier front)
    fin si
  fin Pour
  $R \leftarrow S_p$
  $i = i + 1$
fin Début

Distance de peuplement (*cowding distance*) La dernière critique faite sur le NSGA est l'utilisation du *sharing*. Une méthode qui exige le réglage d'un ou plusieurs paramètre(s). Dans NSGA-II, Deb et al remplacent la procédure de *sharing* par une procédure de *crowding*, basée sur un calcul de distance (distance de crowding) qui ne nécessite aucun paramétrage et qui est également d'une complexité algorithmique moindre que celle de sharing. La distance de crowding d'une solution particulière i se calcule en fonction du périmètre de l'hypercube ayant comme sommets les points les plus proches de i sur chaque objectif. Sur la figure suivante "figure 3.9", est représenté l'hypercube en deux dimensions associé au point i. Le calcul de la distance de crowding nécessite, avant tout, le tri des solutions selon chaque objectif, dans un ordre ascendant. Ensuite, pour chaque objectif, les individus possédant des valeurs limites se voient associés une distance infinie. Pour les autres solutions intermédiaires, on calcule une distance de crowding égale à la différence normalisée des valeurs des fonctions objectifs de deux solutions adjacentes. Ce calcul est réalisé pour chaque objectif. La distance de crowding d'une solution est obtenue en sommant les distances correspondantes à chaque objectif.

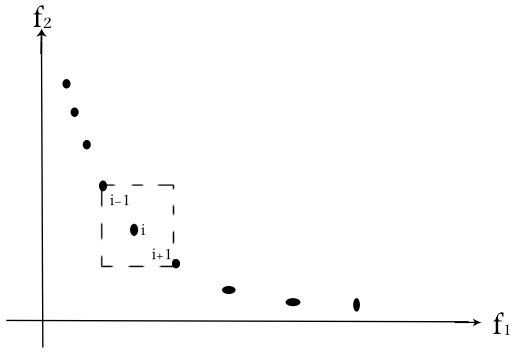


FIGURE 3.8 – Distance de peuplement

L'algorithme 5 reprend toutes les étapes ci-dessus

Algorithm 5 Distance de peuplement

Donnée : f_m(objectifs) ;$l = |I|$ nombre de solution dans le front I
Début
  Pour chaque solution i **faire**
    poser $I\,[i]_{distance} = 0$
  fin Pour
  $I = trier(i, m)$,trier I par ordre croissant selon le critère m (I indice)
  $I\,[1]_{distance} = I\,[L]_{distance} = \infty$
  Pour $i = 2$ jusqu'à $l - 1$ **faire**
    $I[i]_{distance} = \frac{f_m^{i+1} - f_m^{i-1}}{f_m^{max} - f_m^{min}}$
  fin Pour
fin Début

f_m^{max} désigne la valeur maximale de l'objectif m, par analogie, f_m^i désigne la $i^{ème}$ valeur de l'objectif m

Sélection L'opérateur utilisé pour guider le processus de sélection est comme suit : chaque solution i de la population est identifiée par son rang i_{rank} et sa distance de crowd $i_{distance}$ L'opérateur $\preccurlyeq_n$, défini ci-dessous, permet d'établir un ordre de préférence entre deux solutions :

$$i \preccurlyeq_n j \text{ si } \begin{cases} i_{rank} < j_{rank} \\ ou \\ i_{rank} = j_{rank} \text{ et } i_{distance} > j_{distance} \end{cases} \tag{3.9}$$

Donc on sélectionne d'abord celle qui à le plus petit rang (front), si les deux solutions sont dans le même front, on sélectionne celui qui a la plus grande distance de crowd.

SPEA II (*Strength Pareto Evolutionary Algorithm II*)

Récemment Zitzler et al. [20] ont proposé une version améliorée de SPEA [21]. Les modifications majeures se situent au niveau de :

- L'assignation de la valeur d'adaptation (*fitness*) ;
- Technique d'évaluation de la densité par le k^{ieme} plus proche voisin [22].
- Nouvelles méthodes de troncation d'archives.

Comme sera montré dans ce chapitre, l'algorithme SPEA2 fournit une bonne exécution en termes de convergence et diversité.
Le diagramme suivant montre globalement le déroulement de la méthode en question.

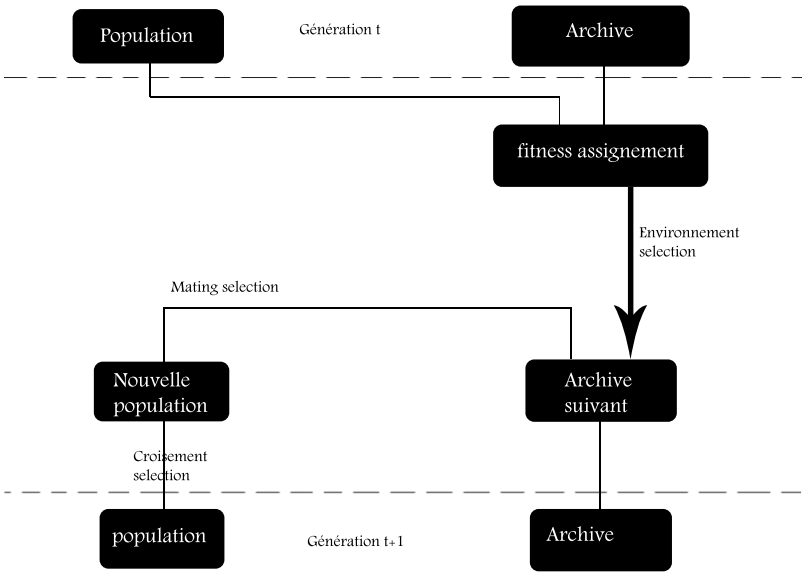


FIGURE 3.9 – Diagramme d'évolution de SPEA II

Une population initiale P_t est créée ainsi qu'une archive Q_t vide à la génération t_0 est mis à jour itérativement à travers les générations, puis combinée où l'opération d'assignation de la fitness attribut à chaque individus une valeur d'adaptation, puis une archive Q_{t+1} de taille fixe définit au préalable $\tilde{\mathsf{N}}$ est crée par les individus non dominés, une sélection par tournois est effectuée sur ces individus pour donner une population temporaire où les opérations de croisement et de mutation donneront la population P_{t+1}, elle sera ensuite combinée avec l'archive Q_{t+1} et le processus se répète jusqu'à satisfaction d'un critère d'arrêt. Les solutions du front Pareto-optimale sont dans l'archive Q_{t+n} qui assure l'élite où N est le nombre de génération utilisé dans la majorité des cas comme critère d'arrêt.

La première étape de l'assignation de la valeur d'adaptation s'effectue sur l'ensemble $\{P \bigcup Q\}$. Cette valeur est déterminée en calculant pour chaque solution $x_i \in \{P \bigcup Q\}$ la valeur S_i qui représente le nombre de solution que x_i domine

$$S_i = \left|\left\{j | j \in P \bigcup Q, i \prec j\right\}\right| \tag{3.10}$$

Où $|.|$ représente la taille.

A partir de la valeur de S_i (*strength value*), en définie une autre métrique R_i appelée valeur d'adaptation brute (*raw fitness*) qui égale à la somme des force S_j où j sont les individus qui dominent i, par conséquent, si un individu n'est dominé par aucun autre il ; sera attribué une valeur R nulle.

$$R_i = \sum_{j \in \{P \cup Q\} \wedge j \prec i} |S_j| \tag{3.11}$$

A un stade avancé de nombre d'itération, cette technique peut échouée lorsqu'aucun individu ne domine l'autre, une information additionnelle de densité est ajoutée pour distinguer entre les individus ayant des valeurs d'adaptation brute R identique

$$D_k(i) = \frac{1}{\sigma_i^k + 2} \tag{3.12}$$

σ_i^k représente la distance entre i et son k^{ieme} voisin, le chiffre 2 au dénominateur sert à obtenir $D_k(i) < 1$ et $k = \sqrt{|P| + |Q|}$[23].La procédure pour l'obtention de la mesure $D_k(i)$ se déroule en quatre parties comme indiquée :

Algorithm 6 Calcul de la $k^{ième}$ distance

Donnée : f_m(objectifs) ;
Début
Calculer σ_{ij} distance euclidienne
Trier(σ_{ij},<)
Initialiser σ_i^k où $k = \sqrt{|P| + |Q|}$
Affecter $D_k(i) = \frac{1}{\sigma_i^k + 2}$
fin Début

Finalement, la valeur de la fitness (adaptation) est somme de R_i et $D_k(i)$

$$F_i = D_k(i) + R_i \tag{3.13}$$

Mise à jour de l'archive (*Environmental selection*)
On commence par sélectionner tout les individus non dominés (ceux qui ont une valeur d'adaptation inférieur à 1) et ainsi mètre à jour l'archive de la génération $t + 1$:

$$Q_{t+1} = \left\{ i | i \in \left\{ Q_t \bigcup P_t \right\} \wedge F_i < 1 \right\} \tag{3.14}$$

Trois cas se présentent alors comme conséquences :
- Si $|Q_{t+1}| = \tilde{\mathsf{N}} \Rightarrow$ fin de l'étape *fitness assignment*
- Sinonsi $|Q_{t+1}| < \tilde{\mathsf{N}} \Rightarrow$ sélectionner les meilleurs individus de $\left\{ \tilde{\mathsf{N}} - |Q_t| \right\}$
- Sinonsi $|Q_{t+1}| > \tilde{\mathsf{N}} \Rightarrow$ une procédure de troncation itérative supprimeras les individus dominés jusqu'à ce que $|Q_t| = \tilde{\mathsf{N}}$

Le critère de dominance utilisé pour la troncation est basé sur la distance σ_{ij}

$$i \prec_d j \Leftrightarrow \begin{cases} \forall 0 < k < |Q_{t+1}| : \sigma_i^k = \sigma_j^k \\ \vee \\ \exists 0 < k < |Q_{t+1}| : \left\{ \forall 0 < l < k : \sigma_i^l = \sigma_j^l \wedge \sigma_i^k > \sigma_j^k \right\} \end{cases} \tag{3.15}$$

Où $\prec_d$ représente la dominance selon la distance euclidienne.

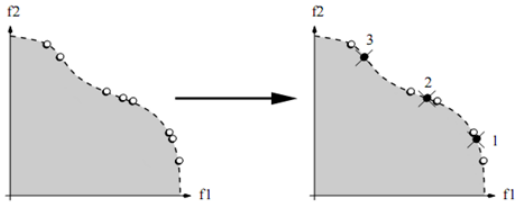


FIGURE 3.10 – Illustration de la méthode de troncation dans SPEA2 $\tilde{\mathsf{N}} = 5$

l'algorithme suivant résume les étapes décrite ci-dessus :

Algorithm 7 Environmental selection

Donnée : P ; population ;Q : archive ;$\tilde{\mathsf{N}}$: taille de l'archive
Début
  $Q_{t+1} = \{i | i \in \{Q_t \bigcup P_t\} \wedge F_i < 1\}$
  Si $|Q_{t+1}| < \tilde{\mathsf{N}}$
    compléter Q_{t+1} par les $\left\{\tilde{\mathsf{N}} - |Q_t|\right\}$ meilleurs individus
  Sinon si $|Q_{t+1}| > \tilde{\mathsf{N}}$
    répéter jusqu'à
      $Q_{t+1} = Q - x_j | x_i \prec_d x_j \forall j \in Q$
    critère d'arrêt
  fin Si
fin Début

Sélection (*Mating selection)* Une sélection basée sur un tournoi binaire avec un nombre de tour égale N, le critère de comparaison est cette fois ci basé sur la valeur d'adaptation F_i, celui qui possède un F_i faible sera sélectionné pour construire une population temporaire. A cette dernière on applique les opérateurs de mutation et de croisement comme dernière étape pour avoir enfin une population P_{t+1}.

L'algorithme suivant résume les étapes de SPEA II :

Algorithm 8 SPEA II

Donnée : N :taille de la population.
$\tilde{\mathsf{N}}$:taille de l'archive
T : nombre de génération
nbr-gen=0 ;
Sortie : A Front de Paréto
Etape 1 : Initialisation
générer une population initiale P et mêtre $Q = []$
While $nbr.gen \leq T$
Etape 2 : fitness assignment
calculer $\left(S_i, R_i, D_i^k\right) \Rightarrow F_i$ (Equ 3.10, 3.11, 3.12 et 3.13)
Etape 3 : environnemental Selection
Algorithme $N^{0=}7$
$A \leftarrow Q$
nbr-gen=nbr-gen+1 ;
EndWhile
Etape 4 : Mating Selection
Tournoi binaire
Etape 5 : Croisement et Mutation
Etape 6 : Fin

3.6 Fonction-test multi-objectif

La fonction de Schaffer SCH1 : La fonction SCH1 (figure 3.11) est un problème de minimisation ayant deux fonctions coût et une seule variable de décision. Elle est proposée par Schaffer. Malgré sa simplicité, elle est parmi les fonctions tests les plus utilisées. Le front de Pareto correspondant étant continu et convexe.

$$SCH : \begin{cases} f_1(x) = x^2 \\ f_2(x) = (x-2)^2 \end{cases} \text{ où } x \in [-5, 5] \tag{3.16}$$

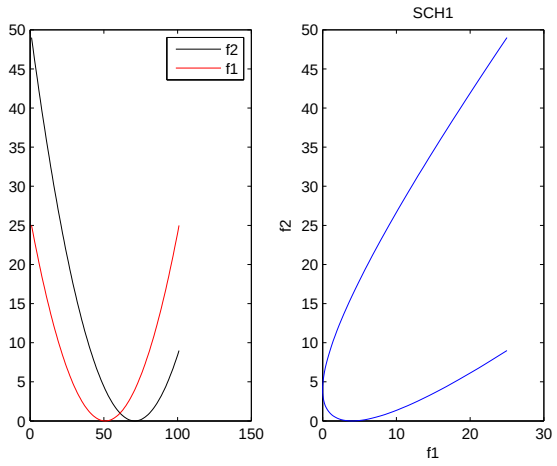


FIGURE 3.11 – Fonction de schaffer "SCH1"

Application de la méthode agrégation pondérée
les paramètres de l'AG sont regroupés dans le tableau suivant :

paramètres	valeur
taille de la population(nbr-ch)	20
précision	0.01
intervalle	[-5 , 5]
nembre de génération (nbr-gen)	200
poid de pondération	linspace(1,0,20)

TABLE 3.1 – Paramètres de la simulation

La figure 3.12 illustre la répartition des solutions non dominées (front de paréto) associée à la fonction de shaffer

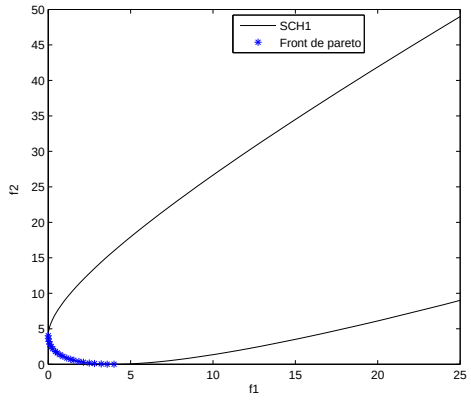


FIGURE 3.12 – Espace des objectif : solution optimale de SCH1

On change le poids de pondération ;ω_i on auras :

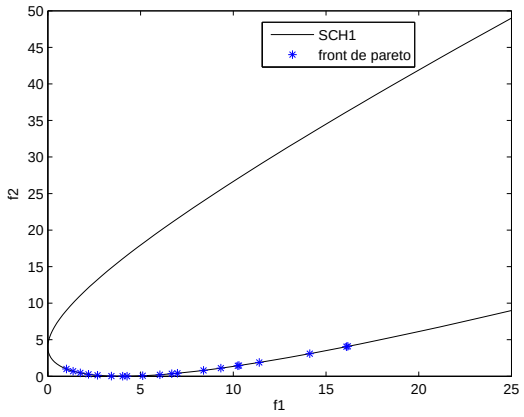


FIGURE 3.13 – Solution optimale de SCH1 pour ω_i=linspace(2,0.5,20)

On remarque que certaines solutions obtenues ne font pas partie du front de pareto et il est difficile de localiser toutes les solutions car le choix du vecteur des poids de pondération ne se fait pas de manière aléatoire. La connaissance de la forme et de la position de la surface de Pareto est très importante pour pour faire un bon choix judicieux des poids de pondération, ce qui n'est pas évident pour la plus grande majorité des cas d'optimisation.

L'algorithme suivant résume les étapes de la simulation

Algorithm 9 Optimisation Multi-objectif par pondération

Donnée : obj=[ ] ;Nouvelle-population=[ ] ;max=nombre de génération maximal ;
nbr-ch=taille de "population" ; nbr-gen=0 ;poid=linespace(x,y,nbr-ch)
meilleur-chromosome=[ ] ;meilleur-population[ ].
For $i \leftarrow 1$: nbr-chr
Etape 1 : Initialisation
bits=calculebits(interval,précision) ;
créer une population ;
α = poid(i)
While nbr-gen $\leq max$
Etape 2 : Décodage
Etape 3 : Evaluation
obj=evaluer(population) ; (Eq 3.8)
Etape 4 : Pré-sélection
indice=min(obj) ;
{meilleur-chromosome} $\bigcup$ population(indice) ;
{population} $\bigcup$ meilleur-chromosome ;
réévaluer la population ;
Etape 5 : Sélection par roue
Etape 6 : Croisement
Etape 7 : Mutation
nbr-gen=nbr-gen+1 ;
EndWhile
Etape 8 : Sélection finale
sélectionner le meilleur chromosome de la
population "meilleur-chromosome" ;
{meilleur-population} $\bigcup$ meilleur chromosome ;
nbr-gen=0 ;
End For

L'étape n°4 sert à archiver les meilleurs chromosomes à chaque itération de la boucle "while", et les combiner avec la population pour qu'ils participent à la sélection.
L'étape n°8 sert à extraire le meilleur chromosome à chaque itération de la boucle "for" ; c'est la population qui contient les solutions optimales.

Application de la méthode SPEA-II

Dans cet algorithme, on reprend les mêmes paramètres utilisés dans la simulation précédente(tableau 3.1) et les résultats obtenus sont illustrés par les figures 3.14 à 3.16

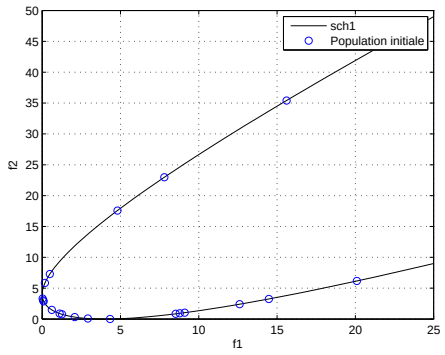


FIGURE 3.14 – Population initiale (espace objectif)

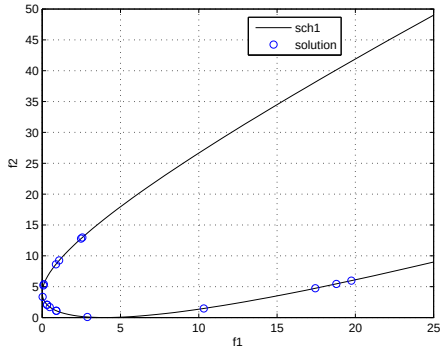


FIGURE 3.15 – Population à la cinquième génération (espace objectif)

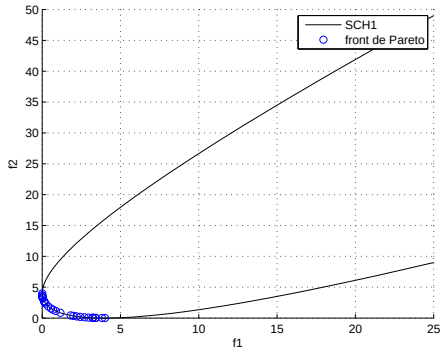


FIGURE 3.16 – Population finale (espace objectif)

On constate une bonne convergence de la population vers le front de pareto avec une diversité nettement meilleur que celui obtenus par la méthode agrégative..

La fonction ZDT3 : On testera cette fois-ci nos algorithmes sur un cas d'optimisation particulier ; a savoir la fonction ZDT3 [25].
La fonction ZDT3 est un problème de minimisation ayant deux fonctions coût et une seule variable de décision. Ce problème est caractérisé par un front de Pareto non convexe, discontinu et difficile à trouver.
La fonction ZDT3 est définie par le système d'équation (3.17) suivant :

$$ZDT3 : \begin{cases} f_1(x_1) = x_1 \\ g(x_2 \cdots x_n) = 1 + 9 \sum_{i=2}^{n} \frac{x_i}{n-1} \\ h(x) = 1 - \sqrt{\frac{f_1}{g}} - \left(\frac{f_1}{g}\right) sin(n\pi f_1) \end{cases} \quad o\grave{u}\ x_i \in [0,1]\ \ et\ n = [2:30] \quad (3.17)$$

Son front de pareto est représenté dans la figure 3.17

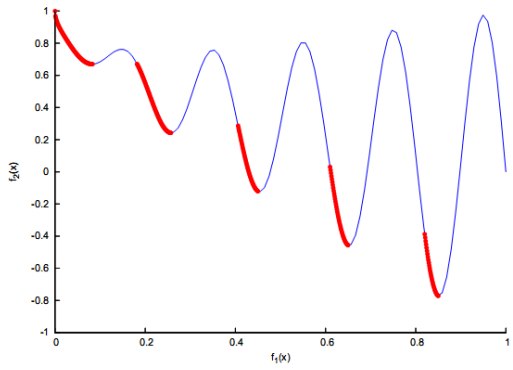


FIGURE 3.17 – Forme du front de pareto de la fonction ZDT3

Application de la méthode agrégation pondérée

Nous utilisons aussi ici les même paramètres indiqués au tableau 3.1 sauf pour l'intervalle de variation de la variable de décision qui est égale à [0,1] :

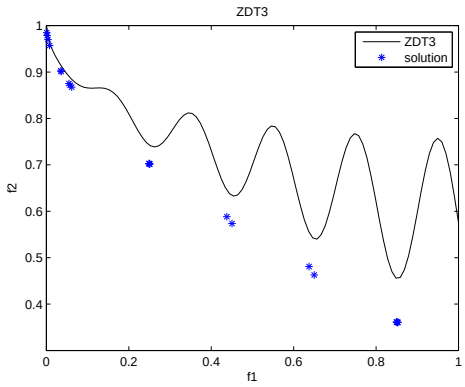


FIGURE 3.18 – Espace des objectif : solution optimale de ZDT3

D'après la figure3.18, cette méthode ne permet pas d'identifier les portions non convexes du front de Pareto. En effet comme on peut le constater sur la figure 3.19, aucun des points de l'arc compris entre $f^A et f^B$ ne peut être déterminé par la minimisation d'une combinaison linéaire des deux fonctions objectifs. On note aussi que le temps de calcul est excessif pour chaque nouvelle du poids correspond une nouvelle exécution de l'application.

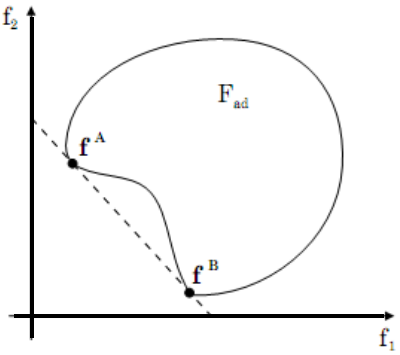


FIGURE 3.19 – Cas d'un front non convexe

Application de la méthode SPEA-II

les figure 3.20 donnent les résultats obtenus par quelques échantillons de générations

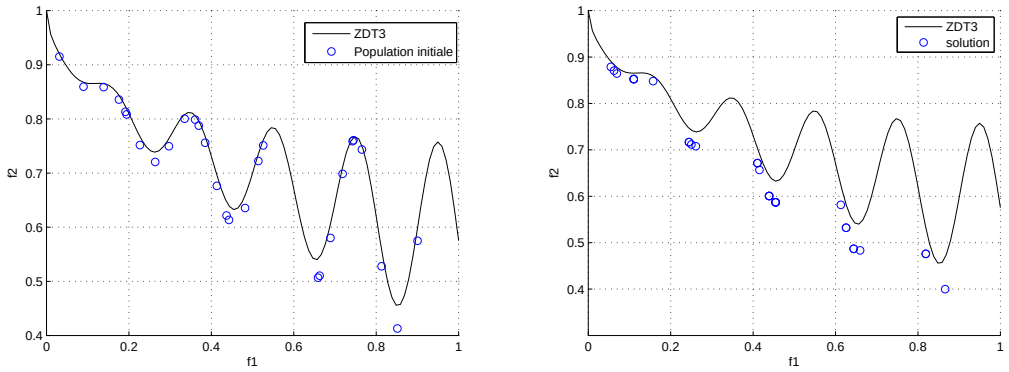


FIGURE 3.20 – Population initiale "à gauche" et population à la cinquième génération "à droite" (espace objectif)

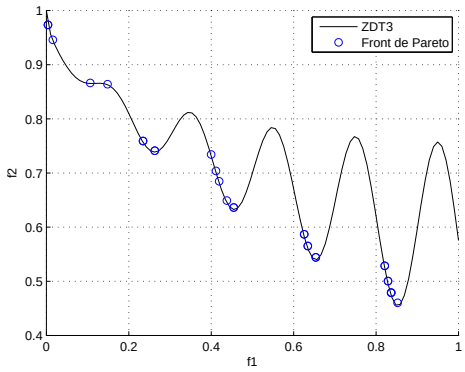


FIGURE 3.21 – Population finale (espace objectif)

L'algorithme converge vers le front de Pareto, malgré que se dernier est de nature non convexe et discontinu.

3.7 Conclusion

les techniques priori souffrent toutes du même inconvénient : au terme de l'étape de recherche, une seule solution optimale est calculée. L'identification d'une autre solution du front de Pareto nécessite donc une réinitialisation du problème (nouvelle valeur du poids de pondération), et le processus d'optimisation doit ensuite être effectué. De plus si on ne dispose d'aucune information sur la forme du front de Pareto, une répartition homogène des points le discrétisant est très difficile à obtenir ou bien la convergence

vers se front ne sera pas assurée(exemple SCH).

Les technique a posteriori consiste à déterminer, sans informations préalables, un ensemble de solutions optimales parmi les-quelles le concepteur sélectionne la solution la plus adéquate, celle qui le satisfait le plus. Il y a donc un gain de temps non négligeable

Chapitre 4

Application au problème d'automatique

La validation faite par les fonctions-test mono-objectif et multi-objectifs respectivement dans le deuxième et troisième chapitre des algorithmes développés à cet effet, nous permet d'aborder l'optimisation des problèmes du domaine de l'automatique ; où nous allons voir dans la première partie de ce chapitre, sous le titre "optimisation mono-objectif", comment adapter un AG simple pour trouver les paramètres optimaux d'un PID qui minimisent l'erreur de la réponse indicielle d'un système.
Dans la deuxième partie(optimisation multi-objectifs), nous allons chercher les paramètres optimaux du PID tout en considérant la minimisation de l'effort de la commande en plus de l'erreur

4.1 Introduction

L'ajustement des paramètres du contrôleur PID est un vieux défi dans le domaine de la commande, on vas voir dans ce chapitre comment adapter un AGs et un AG multi-objectifs à cette fin. Nous allons considérer pour la suite de notre travail le schéma suivant :

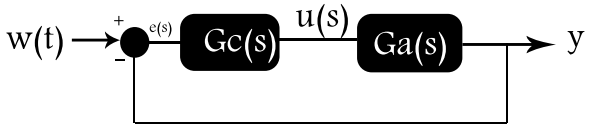


FIGURE 4.1 – Schema fonctionnel

où le contrôleur G_c se situe en amont avec le système à réguler G_a,.

4.2 Optimisation mono-objectif

Nous utilisons dans cette section un algorithme génétique simple(AGs).Nous concéderons la même fonction de transfert utilisée dans le chapitre 1,$G_a(s) = \frac{1}{(1+s)^3}$ et on cherche les paramètres optimaux du PID de façon à est-ce qu'ils donnent une meilleure réponse.
Le choix de l'objectif à minimiser est une étape très importante, puisque le résultat final dépend fortement, nous avons utiliser l'intégrale de l'erreur absolue(IAE) comme critère d'optimisation

$$IAE = \int |e(t)|\, dt \tag{4.1}$$

Nous avons utiliser un codage de type binaire, avec une précision $\epsilon = 0.01$. L'application de l'équation 2.4 donne une longueur $L = 14$ *bits* chaque chromosome contient trois gênes, qui correspondent aux paramètres du PID K_p, K_i et K_d. La taille de la

population est égale à 10.

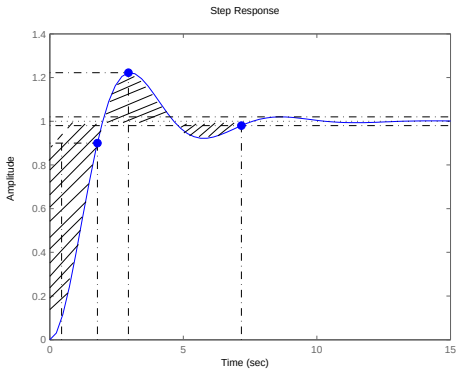


FIGURE 4.2 – Valeur de IAE (surface hachée)

Le décodage se fait dans l'intervalle $[x_{min}, x_{max}] = [0, 100]$; les valeurs obtenues sont remplacées dans K_p, K_i et K_d pour avoir le contrôleur PID :

$$G_c(s) = \frac{K_i + K_p s + K_d s^2}{s} \tag{4.2}$$

Ce dernier est mis en série avec $G_a(s) = \frac{1}{(1+s)^3}$, le tout est bouclés par retour unitaire pour ensuite obtenir la réponse indicielle de la boucle fermée. A partir de cette réponse l'erreur ainsi que le critère IAE sont évalués, c'est l'étape d'évaluation.

La sélection est de type proportionnelle, puisqu'il s'agit du problème de la minimisation, la fonction d'adaptation est l'inverse de la valeur de l'objectif IAE i.e.$f_{ad} = \frac{1}{f_{obj}}$.

Un archive "élite" est mis en place pour conserver le meilleur chromosome de chaque génération.

L'algorithme suivant résume les étapes de la simulation :

Algorithm 10 Optimisation d'un PID par algorithme génétique simple

Donnée : obj=[ ] ;Nouvelle-population=[ ] ;max=nombre de génération maximal ;
nbr-ch=taille de "population" ; nbr-gen=0 ;temps de la simulation
Etape 1 : Initialisation
bits=calculebits(interval,précision) ;
population=créer-pop(bits,nbr-ch) ;
While nbr-gen $\leq$ max
Etape 2 : Décodage
Etape 3 : Evaluation
For $i \leftarrow 1$: nbr-ch
construire le contrôleur PID : G_c ;
BO=series(G,G_c) ;
BF=feedback(BO,1) ;
Y=step(BF) ;
erreur=$1 - Y$;
$f_{obj} = IAE$ =sum(abs(temps.erreur)) ;Eq(4.1)
$\{f_{obj}\} \bigcup IAE$;
End For
Etape 4 : Selection proportionnelle :
Selection par duplication (Algorithme N°3)
Etape 5 :Croisement
population=croisement(Nouvelle-population) ;
Etape 6 :Mutation
population=mutaion(population) ;
nbr-gen=nbr-gen+1 ;
EndWhile

Le tableaux suivant résume les paramètres de l'AG.

paramètres	valeur
taille de la population(nbr-ch)	10
précision	0.01
intervalle	[0 , 100]
nembre de génération (nbr-gen)	100

TABLE 4.1 – paramètres de l'AGs

Résultats :
Nous avons trouver après 100 générations la réponse suivante :

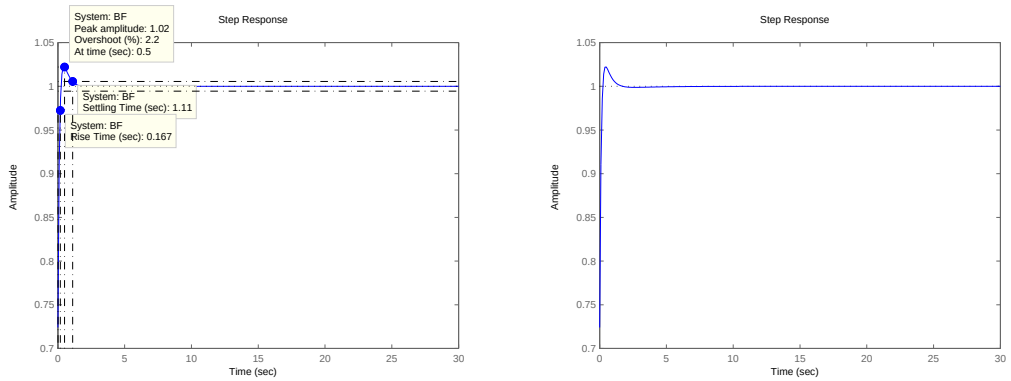


FIGURE 4.3 – Réponse indicielle en boucle fermée (IAE=0.5610, $K_d = 7.8740$,$K_p = 99.9023$ et $K_i = 41.7323$)

La figure 4.3 donne la réponse indicielle correspondante au meilleur chromosome de la dernière génération. Nous avons une réponse très satisfaisante, soient un faible dépassement de 2.2% et de temps de réponse et de montée minimes(T_m=0.167,T_r=1.11)

Le tableau 4.2 compare les résultats de notre AG avec ceux obtenus par la méthode ZN et la méthode de CC.

	D (%)	$T_m(10 \Rightarrow 90\% vf[sec]$)	$T_r \mp 5\% vf[sec]$
AGs	2.2	0.167	1.11
Z.N	5.81	2.31	11.4
C.C	7.6	1.08	6.88

TABLE 4.2 – Comparaison des Résultats des méthodes AGs, Z.N et C.C

le dépassement obtenu (2.2%) est meilleur que celui obtenus par la méthode de Ziegler Nichols (réponse indicielle :5.81% et point critique :17.7%) et dépasse largement celui obtenu par la méthode de Cohen et Coon (7.6%). par contre nous avons vus que la méthode du modèle, à échouée lorsque nous avons voulus obtenir les paramètres PID concernant la fonction de transfert de troisième ordre utilisée dans cette section.Même chose concernant le temps d'établissement ($\pm 5\% vf$) et le temps de réponse ; le résultat obtenus par l'AGS est meilleur que les autres.

4.3 Optimisation multi-objectif

Dans cette partie, l'objectif est de trouver le régulateur optimal offrant une réponse satisfaisante en termes d'erreur tout en minimisant le signal de commande. Pour cela

deux critères sont considérés :

$$IAE = \int |e(t)|\, dt \tag{4.3}$$

$$IAC = \int |u(t)|\, dt \tag{4.4}$$

Le premier (IAE) est l'intégrale de l'erreur absolue que nous avons déjà utilisé dans le cadre de l'optimisation mon-objectif. Le deuxième (IAC) est l'intégrale absolue de la commande, sa minimisation permettra d'économiser l'effort de commande. Notre application va porter cette fois-ci sur un modèle à moteur à courant continu.

Modélisation du Moteur à courant continu

Le schéma d'un moteur CC à excitation séparée constante est donné sur la figure 4.4. Le signal d'entrée est la tension aux bornes de l'induit $u_a(t)$ et le signal de sortie est dans le cas de cet exemple la vitesse angulaire $\omega \left[\frac{rad}{sec}\right]$.

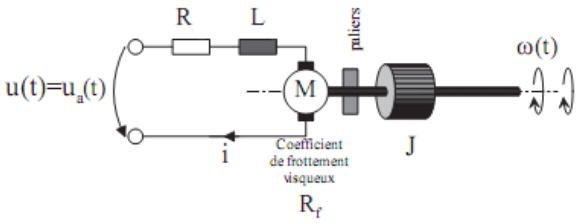


FIGURE 4.4 – Schéma du moteur à courant continu

L'équation électrique, liant la tension $u_a(t)$ aux bornes de l'induit, le courant d'induit $i(t)$ et la force électromotrice $e(t)$ est donnée par :

$$R.i(t) + L\frac{di(t)}{dt} + e(t) = u_a(t) \tag{4.5}$$

Où R est la résistance de l'induit du MCC, L son inductance et $e(t)$ la force électromotrice, qui est proportionnelle à la vitesse de rotation du rotor :

$$e(t) = K_e.\omega(t) \tag{4.6}$$

Où K_e est la constante électrique du moteur (constante de vitesse) et $\omega(t)$ la vitesse de rotation.

L'équation mécanique rendant compte des couples agissant sur le rotor s'écrit :

$$J\frac{d\omega(t)}{dt} = C(t) - C_0(t) - R_f.\omega(t) \tag{4.7}$$

Où $C(t)$ est le couple moteur, $C_0(t)$ le couple résistant (charge et perturbations),R_f le coefficient de frottement visqueux et J le moment d'inertie du rotor. Par construction, le couple $C(t)$ est proportionnel au courant d'induit $i(t)$:

$$C(t) = K_m.i(t) \tag{4.8}$$

Où K_m est la constante du couple moteur.

Aux équations du modèle mathématique correspond le schéma fonctionnel de la figure 4.5 ci-contre.

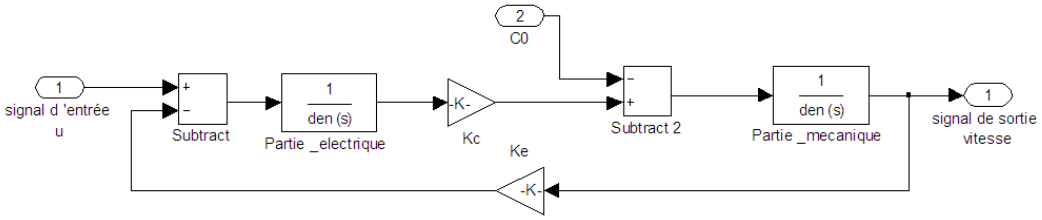


FIGURE 4.5 – Schéma fonctionnel détaillé du moteur DC

Les caractéristique du moteur [26] sont représentées dans le tableau ci-dessous :

tension nominale	$24[V]$
Vitesse à la tension nominale	3820 $[tr/min]$
Tension maximale	32 V
Vitesse limite	5000 $[tr/min]$
Courant maximal à vide	22 $[mA]$
Courant maximal permanent	3.3 $[A]$
Courant de démarrage maximal	12.6 $[A]$
Couple permanent maximale	200 $[mN.m]$
Résistance aux bornes	1.91 $[Ohm]$
Inductance	0.62 $[mH]$
Constante de vitesse	60.3 $[mV/tr/min]$
Constante du couple	60.3 $[mN.m.A^{-1}]$
Coefficient de frottements visqueux	$2.5.10^{-5}$
Inertie du moteur	10^{-4} $[Kg.m^2]$

TABLE 4.3 – Caractéristique technologique du moteur

4.3.1 Simulation en boucle ouverte

La réponse du moteur à la consigne v=32 volts en boucle ouverte donne :

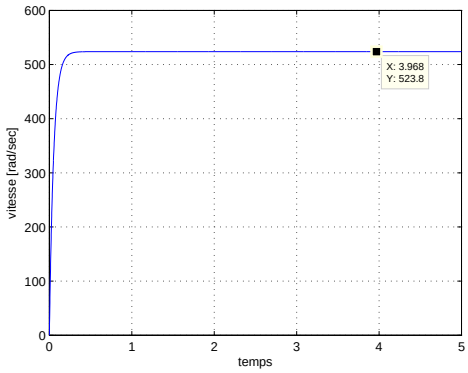


FIGURE 4.6 – Vitesse du moteur pour une commande de 32 volts en boucle ouverte

On constate à partir de l'allure de la réponse qu'il s'agit d'un moteur très rapide du moment que ce dernier atteint sa vitesse maximale après 0.1 s.

4.3.2 Simulation en boucle fermée(Asservissement de la vitesse)

Le système bouclé est maintenant commandé par le correcteur de type PID dont les paramètres optimaux sont obtenus par un algorithme génétique. Comme nous somme devant un problème à deux objectifs (IAE et IAC), l'utilisation d'un algorithme génétique multi-objectifs s'impose. Par manque d'espace, seuls les résultats obtenus par SPEA 2 seront présentés ci-après.
Le tableau suivant donne les paramètres de simulation.

Taille de la population	10
précision	10^{-9}
Probabilité de croisement (Pc)	0.75
Probabilité de mutation (Pm)	0.01
Intervalle de décision	[0 2; 0 2 ; 0 2]
Nombre d'étiration	3000

TABLE 4.4 – Tableau des paramètres de SPEA 2

Résultat La figure 4.7 donne la répartition de la solution finale dans l'espace objectifs, nous avons presque une répartition totale des solutions sur le front de Pareto. Néanmoins la forme de ce dernier est un peu complexe (non convexe et discontinue).

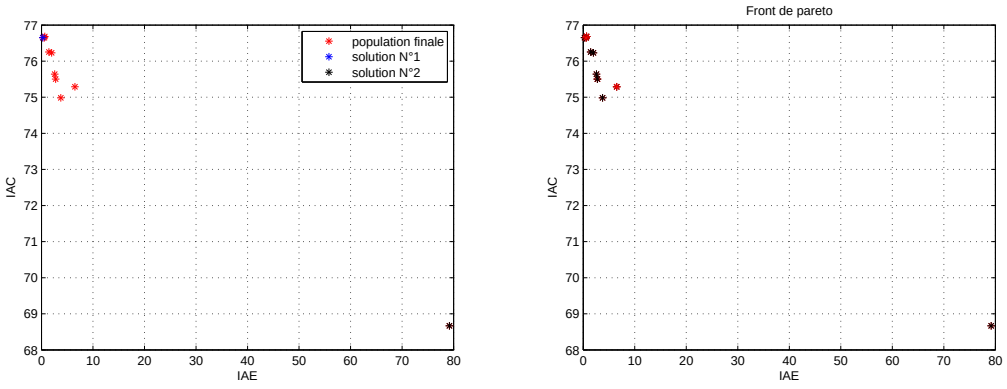


FIGURE 4.7 – Population finale à gauche et front de pareto à droite (espace objectif)

Les figure 4.8 et 4.9 donnent respectivement les sorties du système et les signaux de commande correspondants aux solutions de la population finale. Nous avons des réponses avec des caractéristiques différentes en termes de dépassement et de temps

de réponse que nous pouvons ressentir à partir des signaux de commande qui diffèrent entre eux.

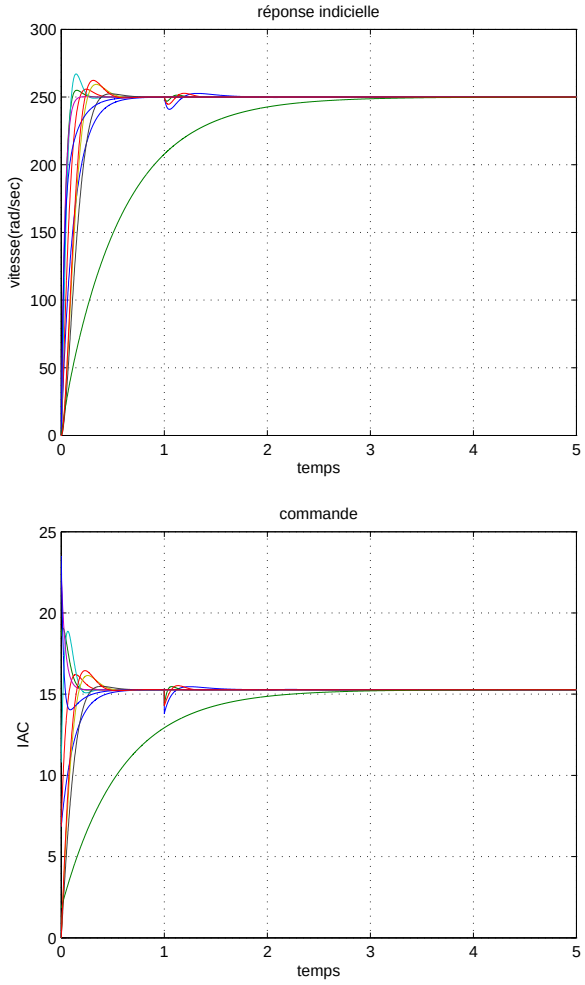


FIGURE 4.8 – Sorties du système (en haut) et signaux de commande correspondants à la population finale(en bas).

On choisit maintenant deux solutions différentes de la population finale et qui délimitent le front de Pareto puis on représente les sorties et les signaux des commande correspondants. 0n constate que la première solution est plus performante en terme de rapidité, mais de coût plus élevé que la seconde. Cette constatation est intuitive car il s'agit bien de deux objectifs contradictoires du moment que la minimisation d'un objectif se fait au détriment de l'autre. Ceci est bien illustré par la figure 4.9 représentant la répartition des solutions du front de Pareto.

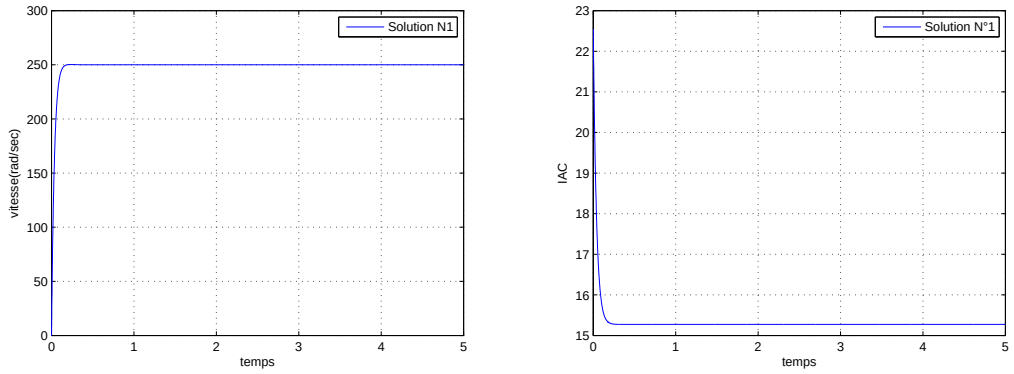


FIGURE 4.9 – Echantillons de solution (signaux de sortie(à droite) et de commande (à gauche))

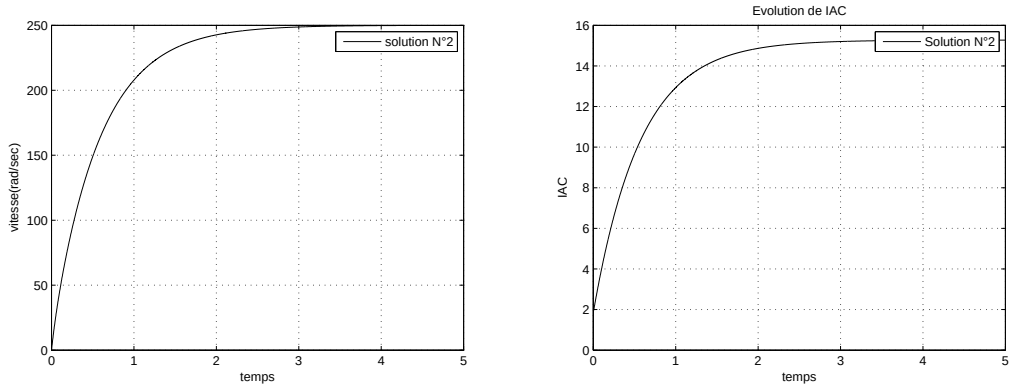


FIGURE 4.10 – Échantillons de solution (signaux de sortie(en haut) et de commande (en bat))

Le tableau suivant donne les valeurs des gains K_d, K_p et K_i) de chaque solution ainsi que les deux objectifs (IAE et IAC) correspondants à ces deux solutions de test.

Échantillons	K_d	K_p	K_i	IAE	IAC
solution N°1	0	0.0898	1.839976	0.2640	76,6496
solution N°1	0	0.0073	0.1099	79.2013	68.6672

TABLE 4.5 – Tableau des paramètres de SPEA 2

En plus des deux objectifs qui diffèrent, notamment IAE, d'une solution à une autre, on remarque que les deux gains K_p et K_i correspondant à la première solution sont assez

éloignés de ceux de la deuxième solution. Seul le gain K_d qui a convergé vers la même valeur pour les deux solutions (Kd=0). C'est d'ailleurs à cette valeur que les solutions de la population finale ont convergé comme le montre la figure 4.11.

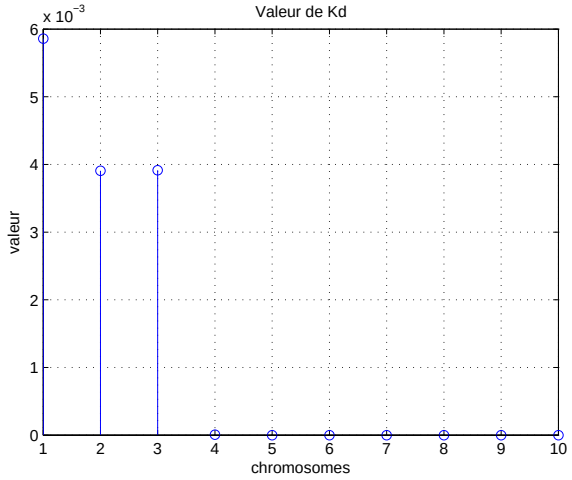


FIGURE 4.11 – Valeurs de K_d correspondantes aux solutions de la population finale(espace de décision)

4.4 Conclusion

Dans ce chapitre, nous avons décrit deux méthodes d'optimisation des paramètres d'un régulateur de type PID par algorithme génétique. Dans la première méthode, nous avons utilisé un algorithme génétique simple pour rechercher les trois paramètres (K_d, K_p et K_i) qui donnent une réponse optimale en sens d'un critère de performance temporelle exprimé par IAE. Les résultats obtenus à travers un exemple de commande d'un système défini par une fonction de transfert du troisième ordre sont très satisfaisants et meilleurs que ceux issus des méthodes classique à savoir celles de ZN et CC. Dans la deuxième méthode, la recherche des paramètres optimaux est effectuée par SPEA2, algorithme d'optimisation multi-objectifs puisque nous avons considéré deux critères à la fois, lAE et l'effort de commande exprimé par l'intégrale absolue de la commande ICA. Un modèle de moteur à courant continu a été utilisé comme exemple de simulation.

Conclusion générale

Les travaux que nous avons menés dans ce mémoire ont pour but de développer une commande optimale de type PID i.e. des paramètres K_d, K_p et K_i qui permettent de minimiser des critères tel que ceux utilisés, à savoir le IAE (integral absolute of error) et IAC (integral absolute of command).

Après avoir retrouver les optimums globaux des fonctions-test mon et multi-objectif respectivement dans le deuxième et troisième chapitre, nous avons utilisé l'algorithme mono-objectif pour l'optimisation des paramètres du PID qui minimisent l'erreur de la réponse du système considéré, les paramètres obtenus conduisent à des résultats meilleurs comparés aux méthodes de Ziegler-Nichols, Cohen et Coon détailler dans le premier chapitre. L'application de la méthode SPEAII sur un moteur à courant continu en optimisant cette fois-ci deux objectifs, à savoir l'erreur et l'effort de commande fourni par le régulateur PID, un ensemble de solution qui reflètent plusieurs compromis entre tous les critères est obtenu à la fin de la simulation où on peut tirer une solution adéquate aux objectifs ou contraintes de la commande.

L'inconvénient majeur des algorithmes génétiques réside dans son mode de fonctionnement offline, une simple variation des paramètres du système peut le déstabiliser ; sauf si on prédit ces variations et modéliser le système au scénario de pire des cas, ce qui complique la stratégie de commande.

Bibliographie

[1] J.G Ziegler AND N.B Nichols : *Optimum settings for automatic controllers.* Trans.ASME,1942.

[2] G.H Cohen AND G.A Coon :*Theoretical consideration of retarded control.* Trans.ASME,1953.

[3] Rolands Burns : *advanced control Engineering.* Butterworth Heinemann,2001.

[4] Holland : *adaptation in natural and artificial system.* Press,Ann Arbor,1975.

[5] L.fogel AND A.Owen AND M.Walsh : *artificial intelligence through simulated evolution.* Press,Wiley and sons, New-york city,1996.

[6] H. schewefel : *Numerical optimization of computers models.* Press,Wiley and sons, Chichester,1981.

[7] D.E Goldberg : *Genetic Algorithms in Search Optimization and Machine Learning.* Press,Addison Wesley, 1989.

[8] Bruno SARENI : *Méthode d'optimisation multimodales associées à la modélisation numériqie en électromagnétisme.*thèse Doctorat : Ecole doctorale d'électronique, électrotechnique et d'automatique de LYON, 02/01/1999.

[9] Abdullah Konaka. AND W.David Coit AND Alice E. Smithc : *Multi-objective optimization using genetic algorithms : A tutorial.* Press,Reliability Engineering and System Safety 91 992-1007,2006.

[10] Mitchell Melanie : *An Introduction to Genetic Algorithms.* The MIT Press Cambridge,1998

[11] Ouahib GUENOUNOU : *Méthodologie de conception de contrôleurs intelligents par l'approche génétique ; application à un bioprocédé.* thèse Doctorat, Ecole Doctoral Système, l'Université Toulouse III - Paul Sabatier,22 avril 2009

[12] D.Corne AND J.D. Knowles AND M.J. Oates : *The pareto envelope-based selection algorithm for multi-objective optimization.* In : Proceedings of the sixth International Conference on Parallel Problem Solving from Nature ,2000

[13] Eckart zitzler : *Evolutionary Algorithms for Multiobjective Optimization : Methods and Applications* thèse Doctorat In : Swiss Federal Institute of Technology Zurich. Diss. ETH N^o. 13398,1999

[14] Eckart Zitzler, Marco Laumanns, et Lothar Thiele : *SPEA2 : Improving the Strength Pareto Evolutionary Algorithm* In : Computer Engineering and Networks Laboratory (TIK) TIK-Report 103 Swiss Federal Institute of Technology (ETH) Zurich. Gloriastrasse 35, CH-8092 Zurich, Switzerland, mai 2001.

[15] Coello, C : *A Coprehensive Survy of Evolutionary-based Multiobjective Optimization techniques* In : Knowledge and Information systems, Aout 1999

[16] Schaffer, J.D : *Multiple objective optimization with vector evaluated genetic algorithms* In : Proceedings of the 1st International Conference on Genetic Algorithms, 1985.

[17] Fonseca et Fleming : *Genetic algorithm objective optimization : formulation, discution and generalization* In : $5^{ième}$ conférence international des algorithmes génétiques (ICGA).Urban Champaign, IL, USA, 1993

[18] J. Horn et N. Napfiotis : *Multiobjective optimization using the niched pareto genetic algorithm* In : Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, 1994

[19] Deb, K., A. Pratap, S. Agarwal et T.Meyarivan : *A Fast and Elitist Multiobjective Genetic Algorithm NSGA-II* In : IEEE Transactions on Evolutionary Computation, 2002

[20] Eckart Zitzler, Marco Laumanns, and Lothar Thiele : *SPEA2 : Improving the Strength Pareto Evolutionary Algorithm* In : TIK-Report 103 Computer Engineering and Networks Laboratory . Department of Electrical Engineering Swiss Federal Institute of Technology (ETH) Zurich. ETH Zentrum, Gloriastrasse 35, CH-8092 Zurich, Switzerland, mai 2001

[21] Eckart Zitzler ET Lothar Thiele : *An Evolutionary Algorithm for Multiobjective Optimization : the Strength Pareto Approch* In : TIK-Report 43 Computer Engineering and Networks Laboratory. Swiss Federal Institute of Technology (ETH) Gloriastrasse 35 CH-8092 Zurich, Switzerland, May 1998

[22] Saku Kukkonen ET Kalyanmoy Deb : *A Fast and Efective Method for Pruning of Non-Dominated Solutions in Many-Objective Problems* In : TIK-Report 43 Computer Engineering and Networks Laboratory, 2007

[23] Silverman, B. W : *Density estimation for statistics and data analysis* In : Chapman and Hall :London, 1986

[24] KarlJ. Astrom et Tore Hagglund : *PID Controllers : Theory, Design and Tuning* In : Instrument Society of America, 1995

[25] Ingenet Test-case Databade.http ://www-sop.inria.fr/sinus/ingenet/