UNIVERSITÉ DE MONTRÉAL

AN EFFICIENT GLOBAL OPTIMIZATION METHOD BASED ON MULTI-UNIT EXTREMUM SEEKING

FARHAD ESMAEIL-ZADEH-AZAR

DÉPARTEMENT DE GÉNIE CHIMIQUE

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION

DU DIPLÔME DE PHILOSOPHIÆ DOCTOR (Ph.D.)

(GÉNIE CHIMIQUE)

AOÛT 2010

© Farhad Esmaeil-Zadeh-Azar, 2010.

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée:

AN EFFICIENT GLOBAL OPTIMIZATION METHOD BASED ON MULTI-UNIT EXTREMUM SEEKING

présentée par : ESMAEIL-ZADEH-AZAR Farhad

en vue de l'obtention du diplôme de : Philosophiæ Doctor

a été dûment acceptée par le jury d'examen constitué de :

M. HENRY Olivier, Ph.D., président

M. PERRIER Michel, Ph.D., membre et directeur de recherche

M. SRINIVASAN Bala, Ph.D., membre et codirecteur de recherche

M. GOURDEAU Richard, Ph.D, membre

M. MCLELLAN James, Ph.D, membre

DEDICATION

To Farahnaz and Asad my Mother and Father

for their endless love to me

Everything is possible. Nothing is irreversible.

"It's kind of fun to do the impossible."

Walt Disney

ACKNOWLEDGEMENTS

It is a pleasure to thank several people without their help this thesis could not have been written. They constantly helped me to realize my perspective towards education and life.

First of all, I would like to express my deep and sincere gratitude to my academic adviser, professor Michel Perrier, who kindly accepted to supervise this Ph.D. program. I appreciate the attention he brought to me for being not only a teacher but also a friend, for all his advice, patience, encouragement, financial support and for all the things that I learnt from him in science and life. I was impressed by his wisdom, kindness, knowledge and enthusiasm throughout my research activities and consideration that made my doctoral study possible. The excellence and magnitude of his academic actions during his professional carrier is world-wide recognized. I am very lucky and proud to be his student. *Avoir bien voulu communiquer en français lors de nos réunions, professeur Perrier a également accepté d'etre une source essentielle d'encouragement pour l'apprentissage du français, une des langues les plus belles du monde. Merci Michel.*

I am deeply grateful to my academic co-adviser professor Bala Srinivasan. His overflowing wealth of ideas, his brilliant insight into optimization and control and his invaluable ability to teach research methodologies and scientific writing have significantly influenced my thesis. He has been a continual source of inspiration for me with his generous help, advice and feedback. His profound and extensive knowledge on the interdisciplinary research techniques have been exceptionally valuable for me. Foremost, his availability is exponentially appreciated. I am very thankful for the initial motivation and inspiring discussions that he provided for me and for his tireless guidance on many aspects of multi-unit optimization.

I would like to acknowledge the research assistantship source of Natural Sciences and Engineering Research Council of Canada (NSERC) that have kindly supported my educational effort over my doctoral studies at École Polytechnique de Montreal.

There are many friends whose valuable friendship I would like to acknowledge. I have made many friends during my stay in Montreal who have provided wonderful communication and unforgettable moments. Special thanks to my fantastic friends Jean-Philipe Laviolette, Armin Setayeshgar, Arash Moosavi, Mohsen Marami, Samir Chalfoun, Rached Jomni and Redouane

Khris for their true friendship. Thanks to all my friends for their inspiration through these years and to my other friends who have not been directly mentioned here. Their friendship has made all of the difference. Members of the departmental staff whom I would like to thank include Chantal Bénard, Louise Beaudry-Parent, Lyne Henley, Agnès Devarieux and Lionel Valero who make the atmosphere in our department much more pleasant. Many thanks are also due to the fellow graduate students in the department and my office colleagues at École Polytechnique including Lyne Woodward, Bertrand Pigeon, Roberto Pinto, Massi Si-Mehand and François Reney who created a friendly and pleasant working environment. They certainly made work more enjoyable than it would otherwise have been. I wish all of them the best of luck in their future pursuits.

Finally and most of all I would like to express immense gratitude to my parents who gave me the opportunity to receive a good education and made this dissertation possible. My father who has given unflagging enthusiasm, help and constant support and my mother who has been the limitless source of love and innumerable sacrifices were both necessary and sufficient to complete this doctoral study. Thanks to my dear sister with all my heart, for her words of encouragement throughout the course of this endeavor.

I know that I could not achieve so many wonderful things without the everlasting love from my father and my mother, the most precious thing in my life. I know that I can never thank them enough, but I just tried to express my heartfelt gratitude to them for their unfailing believe in me and my abilities throughout my life. This thesis is dedicated to them.

RÉSUMÉ

Les problèmes d'optimisation industrielle, telle que la maximisation de la production de produits chimiques et pétrochimiques, montrent généralement plusieurs points optimaux locaux. Le développement de méthode pour la sélection du point optimal global a toujours fait l'objet de nombreuses recherches. Plusieurs techniques déterministes et stochastiques ont été explorées à cette fin. Les techniques stochastiques ne garantissent pas toujours la convergence vers la solution globale, mais sont efficaces pour les dimensions supérieures. D'autre part, les méthodes déterministes se rendent à l'optimum global, mais le défi est d'employer un cloisonnement efficace de l'espace afin de réduire le nombre d'évaluations fonctionnelles.

Cette thèse propose une approche originale en matière d'optimisation globale, numérique et déterministe basée sur des techniques d'optimisation locale en temps réel et en particulier, sur des techniques sans modèle appelé les systèmes de commande extrémale. Pour les problèmes sans contrainte, les systèmes de commande extrémale représente le problème d'optimisation comme un contrôle du gradient. La façon dont le gradient est estimé constitue la différence principale entre les différentes alternatives qui sont proposées dans la littérature scientifique. Pour les méthodes de perturbation, un signal d'excitation temporelle est utilisé afin de calculer le gradient. Une alternative existe dans le cadre d'optimisation multi-unité où le gradient est estimé par la différence finie de la sortie de deux unités identiques, mais dont les données d'entré se distinguent par un décalage.

Le point de départ de cette recherche a été motivée par les systèmes de commandes extrémales locales. Ces commandes sont basées sur une perturbation qui peut être utilisée comme un outil pour l'optimisation globale des polynômes scalaires du quatrième ordre avec un optimum global. L'objectif de cette thèse est d'étendre ce concept et de développer une technique d'optimisation globale déterministe pour une classe générale de systèmes multi-variables, statiques, non linéaires et continus. Dans cette thèse, il est d'abord démontré que si le décalage est réduit à zéro pour une optimisation multi-unité scalaire, le système converge vers l'optimum global. Le résultat est également étendu aux problèmes scalaires avec contraintes qui sont caractérisés par des régions non-convexes. Dans ce cas, une stratégie de commande de "Switching" est utilisée pour faire face aux contraintes.

L'étape suivante consiste à étendre l'algorithme à plus d'une variable. Pour les systèmes à deux entrées, l’optimisation globale mono-variable a été répétée sur la circonférence d'un cercle de rayon réduit. Avec trois variables, l'optimisation à deux variables mentionnées ci-dessus a été répétée sur la surface d'une sphère de rayon réduit. L’échéance de séparation entre les différentes couches (optimisation mono-variable, ce qui réduit le rayon du cercle et le rayon de la sphère) a démontré la nécessité de garantir la convergence. Les concepts théoriques sont illustrés pour l'optimisation globale de plusieurs exemples de référence. Les résultats de la comparaison avec d'autres méthodes de concurrence ont montré l'efficacité de la nouvelle technique en termes du nombre d'évaluations fonctionnels.

ABSTRACT

Industrial optimization problems, e.g., maximizing production in chemical and petrochemical facilities, typically exhibit multiple local optimal points and so choosing the global one has always attracted many researchers. Many deterministic and stochastic techniques have been explored towards this end. The stochastic techniques do not always guarantee convergence to the global solution, but fare well computationally for higher dimensions. On the other hand, the deterministic methods get to the global optimum, while the challenge therein is to employ an efficient partitioning of the space in order to reduce the number of functional evaluations.

This thesis proposes an original approach to numerical deterministic global optimization based on real-time local optimization techniques (in particular, model-free techniques termed the extremum-seeking schemes). For unconstrained problems, extremum-seeking schemes recast the optimization problem as the control of the gradient. The way the gradient is estimated forms the main difference between different alternatives that are proposed in the literature. In perturbation methods, a temporal excitation signal is used in order to compute the gradient. As an alternative, in the multi-unit optimization framework, the gradient is estimated as the finite difference of the outputs of two identical units driven with the inputs that differ by an offset.

The starting point of this research was motivated by the perturbation-based extremum seeking schemes which can be used as a tool for global optimization of scalar fourth order polynomials, with one local and one global optimum. The objective of this thesis is to extend this concept and develop a deterministic global optimization technique for a general class of multi-variable, static, nonlinear and continuous systems. In this thesis, it is first shown that in the scalar multi-unit optimization framework, if the offset is reduced to zero, the scheme converges to the global optimum. The result is also extended to scalar constrained problems, with possible non-convex feasible regions, where a switching control strategy is employed to deal with the constraints.

The next step consists of extending the algorithm to more than one variable. For two-input systems, univariate global optimization was repeated on the circumference of a circle of reducing radius. With three variables, the two-variable optimization mentioned above is repeated on the surface of a sphere of reducing radius. Time-scale separation between the various layers

(univariate optimization, reducing the radius of the circle and reducing the radius of the sphere) was shown to be necessary to guarantee convergence. The theoretical concepts are illustrated on the global optimization of several benchmark examples. The comparison results with other competitive methods showed the efficiency of the new technique in terms of number of function evaluations.

CONDENSÉ EN FRANÇAIS

Des algorithmes d'optimisation ont été développés dans pratiquement toutes les disciplines de l'ingénierie et de la science depuis plusieurs décennies. Parmi ces algorithmes, on retrouve une très grande diversité. La recherche des valeurs optimales des variables manipulées peut être faite en se basant sur des modèles mathématiques ou en utilisant des méthodes expérimentales. De nombreuses méthodes soit déterministes soit probabilistes, ont été développées pour effectuer une optimisation dite globale (Floudas et al, 2008). La complexité de calcul, le coût et la précision de ces méthodes diffèrent. Néanmoins, ces classes de méthodes sont basées sur des modèles et typiquement ne vont pas atteindre l'optimum global du système physique réel, car il existe un décalage entre le modèle mathématique et la réalité.

La fonction objectif qui décrit le rendement économique d'un processus industriel est typiquement non linéaire et contient différents paramètres tels que les conditions d'opération, les prix des produits et le prix des matières premières. En général, cette fonction peut présenter plusieurs optima locaux (maxima, minima et les points de selle), mais le maximum global est généralement recherché. L'optimisation du procédé avec une méthode locale peut mener à une baisse du profit, puisque la méthode n'est pas toujours capable de trouver l'optimum global. Ceci n'est pas seulement dû à une imprécision dans le modèle, car cette baisse peut se produire même si le processus réel est modélisé à la perfection. Trouver la valeur optimale globale des variables de contrôle d'un processus industriel est un objectif pour de nombreuses applications d'ingénierie.

Les méthodes d'optimisation à base de modèle ne sont pas toujours capables de trouver les meilleures conditions fonctionnement d'un procédé. En outre, il n'y a parfois pas de modèle approprié pour le procédé et les seules données mesurables ne sont que des paramètres d'entrée et de sortie. Les problèmes d'optimisation où les propriétés du système sont peu connues permettent une approche alternative connue sous le nom d' « optimisation boîte noire ». Les scénarios « boîte noire » sont pertinents lorsque la fonction objectif : (1) n'est pas disponible sous une forme fermée (les valeurs de la fonction sont déterminées par des calculs complexes, des

simulations ou des expériences) ou (2) est très complexe et/ou mal comprise (Zitler, 2003). Dans un scénario « boîte noire », la fonction objectif ($f: U \rightarrow Y$) est traitée comme une boîte noire, telle une procédure exécutable dans un ordinateur pour lequel le code de programmation n'est pas connue ou n'est pas accessible.

Dans certaines applications, l'objectif principal est donc de trouver la valeur optimale d'une fonction objectif qui est difficile à modéliser ou complètement inconnue. Ces problèmes ont une fonction objectif qui peut ne pas être facilement différentiable. C'est pourquoi une stratégie sans modèle d'optimisation globale pour les systèmes « boîte noire » est nécessaire afin d'ajuster le processus à son meilleur point de fonctionnement. Les algorithmes « boîte noire » utilisent moins d'hypothèses sur la fonction objectif comparativement à une classe plus générale de problèmes (Laguna et al., 2010). Aucune information locale ou globale sur la fonction objectif n'est prise en compte. Dans ce cas, il est difficile de déterminer si une solution globale optimale a été trouvée, à moins que l'espace de décision tout entier ait été prélevé. À cette fin, les méthodes stochastiques d'optimisation globale peuvent être utilisées, mais la convergence vers l'optimum global réel n'est pas toujours garanti (Schneider, 2006). Toutefois, l'augmentation de la capacité de calcul des ordinateurs entraîne également une hausse de la capacité à concevoir des algorithmes pour des systèmes mal définis. Différentes méthodes déterministes et stochastiques ont été développées pour traiter ces types de problèmes d'optimisation (Kargupta et Goldberg, 1997).

Les méthodes stochastiques tentent de résoudre les problèmes d'optimisation en introduisant des éléments aléatoires dans l'algorithme. Les méthodes stochastiques ne garantissent pas toujours la convergence vers l'optimum global. Les algorithmes de Monte Carlo et « multi-start » sont deux méthodes typiques de recherche aléatoire (Schoen, 1991; Zilinskas, 1989). Les algorithmes « Bayesians » (Betro, 1983), les méthodes de « Clustering » (Rinnooy et Timmer, 1987), « Simulated annealing » (Kirpatrik et al., 1983), les algorithmes génétiques (Holland, 1973; Goldberg, 1989) et les stratégies d'évolution (Rechenberg, 1973) sont d'autres exemples de techniques de recherche avec adaptation d'échantillonnage. Ces techniques ont été largement utilisées pour résoudre les problèmes d'optimisation de type « boîte noire ». Ces algorithmes exploitent les informations recueillies à partir d'échantillons de l'espace de recherche.

Les algorithmes déterministes recherchent systématiquement la région de faisabilité à partir des informations recueillies sur la fonction objectif. DIRECT est un algorithme déterministe qui utilise les arguments de constante de « Lipschitz » (de zéro à l'infini) afin de décider quelles régions de l'espace de recherche méritent une exploration à chaque itération (Johns et Perttunen, 1993). De cette façon, l'algorithme explore la région de recherche de manière efficace puisqu'il concentre ses évaluations de fonctions supplémentaires dans les régions qui semblent « potentiellement optimales ».

Le problème d'optimisation considéré dans cette étude est l'optimisation globale d'un système statique et continu, où la fonction objectif est non convexe. Les variables manipulées peuvent être estimées en ligne en se basant sur des mesures disponibles. En outre, la différentiabilité du système n'est pas nécessaire. Le problème d'optimisation peut contenir de contraintes d'inégalité et les mesures appropriées des contraintes sont également disponibles. Enfin, la connaissance initiale des caractéristiques mathématiques du processus est supposée très limitée, de sorte que l'utilisation d'un modèle de base pour l'optimisation est considérée comme non nécessaire.

La littérature scientifique propose plusieurs méthodes d'optimisation globale (Floudas et al, 2008). Toutefois, plusieurs de ces méthodes ne sont pas applicables à des problèmes d'optimisation « boîte noire » en raison de leurs hypothèses sur les propriétés de la fonction objectif. Comme mentionné plus haut, les méthodes d'optimisation globale qui tiennent compte d'une information préalable sur les caractéristiques et la structure de la fonction objectif ne peuvent être considérées comme une optimisation « boîte noire ». D'un autre côté, un débat existe toujours quant au degré d'efficacité des techniques d'estimation du gradient des processus continus qui peuvent être utilisés à des fins d'optimisation globale sans l'intermédiaire d'un modèle.

D'autre part, des méthodes d'optimisation sans modèle ont été étudiées sous le nom de commande extrémale, où le concept de base consiste à reformuler le problème d'optimisation sans contrainte

en un problème de contrôle du gradient où ce dernier doit atteindre zéro. Bien que cette méthode soit assez ancienne (Leblanc, 1922), elle a reçu un intérêt renouvelé récemment (Ariyur & Krstic, 2003; Guay et al, 2004; Srinivasan, 2007). En outre, de nombreuses publications ont rapporté des applications (e.g. Ariyur & Krstic, 2003; Propović et al, 2003). L'absence de modèle des méthodes de commande extrémale les rend aptes à gérer l'optimisation « boîte noire » (Guay et Dochain, 2010). Toutefois, la valeur de la fonction objectif doit être mesurée en ligne. À cet égard, les schémas de commande extrémale sont des méthodes d'optimisation qui contrôlent le gradient à zéro. Les commandes extrémales sont des méthodes en temps réel qui remanient le problème d'optimisation en un problème de contrôle et profitent de la réduction de la sensibilité en rejetant des perturbations. La façon d'estimer le gradient est la différence principale entre ces différentes techniques.

Deux méthodes principales d'estimation du gradient seront discutées dans cette thèse. Pour la commande extrémale locale basée sur une perturbation, un signal d'excitation externe est utilisé pour calculer le gradient numérique. C'est une technique traditionnelle (Leblanc, 1922; Ariyur et Krstic, 2003), où une variation temporelle, c'est-à-dire un signal de vibration avec une amplitude constante et préfixée, est ajouté à l'entrée. Le gradient est obtenu par une corrélation entre les entrées et les sorties. Comme alternative, Srinivasan (2007) a proposé une méthode d'optimisation multi-unités, où le gradient est calculé sur la base des différences finies entre un ensemble d'unités parallèles qui fonctionnent avec des valeurs d'entrée différentes par un décalage constant et fixé à l'avance. Le gradient est poussé à zéro par un intégrateur et il a été démontré que la séparation d'échelle de temps n'est pas nécessaire. De plus, l'optimisation multi-unités pourrait se traduire par une convergence plus rapide (Woodward et al, 2009). Toutefois, la convergence de ces deux techniques dépend de leur état initial, ce qui amène le système à converger vers l'optimum local le plus proche.

Les deux stratégies de commande extrémale mentionnées ci-dessus mènent à l'optimum local, car elles sont basées sur le gradient. L'optimum local qui est atteint dépend des conditions initiales du procédé où commence l'optimisation. Ainsi, l'algorithme peut être piégé dans un optimum local au lieu d'atteindre le global. Ces situations conduisent à une performance inférieure du procédé et motivent le développement de stratégies de commande extrémale globale. À cette fin, certains

schémas ont été proposés récemment pour l'optimisation globale d'une classe limitative de systèmes non-linéaires. Une méthode de commande extrémale globale basée sur les perturbations a été analysée par Tan et al (2006 a, 2006 b), ce constitue un prolongement de leurs travaux sur les propriétés de stabilité semi-globale des contrôleurs de commande extrémale (Tan et al, 2005). L'idée de base est de réduire l'amplitude du signal de perturbation asymptotiquement vers zéro. Cette méthode superpose une perturbation périodique asymptotiquement appliquée sur le processus, connue sous le nom de signal « dither », afin d'observer ses effets sur la fonction objectif. Une corrélation entre les mesures de la fonction objectif et cette perturbation peut estimer le gradient à l'état stationnaire. La perturbation à la baisse est temporaire et une bonne estimation du gradient à l'état d'équilibre nécessite différentes échelles de temps entre la fréquence des perturbations, des filtres de coupure et d'adaptation. La vitesse de convergence de cette méthode est généralement lente. Il a été montré que, bien que cette stratégie ait été testée avec succès sur une collection de problèmes non-linéaires scalaires, elle est seulement applicable à une classe limitative de systèmes statiques.

Développer une technique d'optimisation globale pour une classe plus générale de systèmes non-linéaires a été la motivation principale dans la présente étude. D'ailleurs, l'utilisation d'une stratégie qui peut éliminer la séparation des échelles de temps et accélérer la convergence vers l'optimum dans ce contexte serait très efficace. Compte tenu de la définition du problème ci-dessus, la méthode d'optimisation multi-unités, où le gradient estimé par les mesures est contrôlé à zéro, fournit le cadre de la présente thèse. Cette méthode nécessite la présence d'unités identiques pour optimiser le processus. Dans le schéma local de cette méthode, la perturbation constante entre les valeurs des variables d'entrée des unités identiques ne constitue pas un décalage temporel. Le gradient est alors estimé par des différences finies entre les mesures de la fonction objectif de ces multiples unités. Effectuer des modifications à cette méthode afin de la rendre convergente à l'optimale globale constitue l'idée principale pour résoudre le problème mentionné. À cette fin, la présente thèse présente une technique d'optimisation globale déterministe en utilisant le cadre d'adaptation multi-unité pour une classe générale de systèmes non linéaires. La restriction de la commande extrémale fondée sur des perturbations qui mènent à une convergence vers un faux optimum a été supprimée dans la nouvelle stratégie.

La présente thèse propose une méthode d'optimisation globale et déterministe qui utilise l'esprit des stratégies de la commande extrémale en temps réel pour le contrôle du gradient. L'objectif principal est d'effectuer l'optimisation globale de systèmes statiques et non linéaires continus en utilisant des outils d'optimisation locale en temps réel. À cette fin, l'extension de la commande extrémale multi-unités classique et locale à une technique d'optimisation déterministe globale et sans modèle est prise en considération. Toutefois, il ne s'agit pas d'une méthode d'optimisation en temps réel qui suit la variation de l'optimum global en permanence. De ce point de vue, cette réalisation peut être considérée comme une stratégie « d'optimisation globale » pour les systèmes « boîte noire ». Des exemples illustratifs sont présentés pour certains modèles mathématiques de systèmes non linéaires. Ces exemples simulent les processus réels pour obtenir des mesures où le gradient est exclusivement estimé par les valeurs d'entrée/sortie en ligne. Ceci est moins restrictif que les méthodes d'optimisation basées sur des modèles qui utilisent certaines propriétés mathématiques d'un modèle virtuel pour estimer le gradient. Dans l'algorithme présenté, après l'acquisition de données à partir des sorties mesurées, il n'est pas nécessaire d'estimer les paramètres inconnus d'un modèle ou mettre à jour les coefficients d'un modèle comme le nécessitent les optimiseurs classiques en temps réel. Les données acquises sont directement utilisées pour optimiser une fonction objectif afin de trouver les nouveaux points d'opération optimale qui sont transférés au système de contrôle pour les mettre en œuvre dans le processus. C'est la raison pour laquelle l'algorithme est considéré comme une stratégie d'optimisation sans modèle. Il est démontré que l'algorithme présenté converge vers un voisinage très petit de l'optimum global d'un système statique non linéaire et continu. La nouvelle méthode utilise le concept de la réduction du décalage asymptotique vers une petite valeur positive dans l'approche de l'optimisation multi-unités. Il est montré que, avec une modification mineure de l'algorithme d'adaptation, l'algorithme converge vers un petit voisinage de l'optimum global sans conditions préalables sur la fonction objectif. Enfin, des relaxations de l'algorithme sont présentées pour le rendre numériquement efficace.

La première contribution de la présente thèse est l'extension de la commande extrémale multi-unités locale à l'optimisation globale de systèmes statiques, non linéaires, continus et scalaires. Le chapitre 2 décrit l'optimisation globale d'une fonction scalaire non linéaire et sans bruit. Dans ce chapitre, une technique d'optimisation globale et déterministe pour une classe

générale de systèmes statiques, non linéaires et continus a été élaborée. La méthode présentée est un algorithme sans modèle du procédé qui utilise les données de mesure de la fonction objectif pour estimer le gradient. Cette technique permet de surmonter le défaut classique de la commande extrémale en temps réel, soit la convergence aux optima locaux selon les conditions initiales définies. La méthode présentée utilise la structure d'optimisation multi-unités, où certains décalages prédéfinis sont mis en place entre les entrées de deux unités identiques et le gradient est estimé par différences finies. Toutefois, au lieu de déduire le gradient par l'ajout d'un signal de perturbation et par le calcul de la corrélation entre l'entrée et la sortie, cette nouvelle méthode introduit des entrées légèrement différentes à la position de deux unités identiques en parallèle, ce qui permet le calcul du gradient. Dans cette technique, l'excitation est la différence entre les entrées introduites sur les deux systèmes.

La méthode utilise quelques propriétés intéressantes des différences finies jusqu'à présent inexploitées. L'objectif principal est de réguler le point de fonctionnement des systèmes multi-unités simultanément à l'optimum global. L'intention est de rejoindre la région réalisable de manière adaptative. L'idée de base consiste à réduire l'amplitude du signal de décalage asymptotiquement vers zéro, ce qui est très efficace en théorie. Il a été démontré que si les décalages sont réduits à zéro d'une manière contrôlée, l'ensemble du système peut converger vers l'optimum global. L'optimisation semi-globale a été réalisée en débutant la méthode avec un paramètre de décalage assez grand entre les entrées et puis le réduire de façon monotone à une valeur petite « ε ». Par cette technique, il est possible de converger vers l'optimum global d'un système statique et non linéaire si le paramètre scalaire initial de l'algorithme a été sélectionné correctement.

La dynamique de chaque unité est formulée de manière à ce qu'elle absorbe le mouvement des autres unités à un meilleur point de fonctionnement local sur le plan non linéaire. L'algorithme passe sur les optima locaux et converge vers l'optimum global. La nature déterministe de cette approche garantit la convergence de l'algorithme à l'optimum global. L'approche est d'abord présentée pour l'optimisation sans contrainte, suivie d'une extension à des problèmes avec

contraintes, pour laquelle une logique de « Switching » est introduite. Plusieurs exemples académiques (scalaires) sont présentés pour illustrer cette approche.

Le chapitre 2 s'attarde à l'idée de base d'une telle extension et à la preuve de convergence vers l'optimum global de la courbe statique comportant deux unités identiques. Selon l'hypothèse où les courbures non-convexes des unités statiques sont les mêmes, il a été démontré que la convergence globale de l'algorithme peut être assurée par un choix de perturbation variable introduite entre les points de fonctionnement de chaque unité (Δ). Comme mentionné ci-dessus, la principale contribution de notre approche est que l'optimum global peut être atteint au coût d'un calcul relativement simple par une décroissance monotone du décalage à zéro. L'algorithme déterministe converge à proximité de l'optimum global par le choix approprié de ce décalage. Choisir un décalage initial (Δ_0) suffisamment petit minimise le nombre d'évaluations de la fonction et se termine par une convergence plus rapide.

D'autre part, si la valeur de la perturbation initiale choisie est trop petite, le système converge vers un état d'équilibre différent de l'optimum global réel. Toutefois, il est garanti que le point d'équilibre obtenu est toujours l'optimum global dans l'intervalle d'exploitation qui est imposé par la décalage initial (Δ_0) à multi-unités. Il a été démontré que, selon la caractéristique de la constante de Lipschitz des systèmes statiques, les lois d'adaptation peuvent introduire une rigidité («stiffness») dans le processus d'intégration. Une modification à la conception originale de l'algorithme a été mise en place pour minimiser cet effet. Une autre solution pour surmonter cette rigidité a été de remplacer la fonction « Signe » par la fonction « Tangente Hyperbolique » dans la loi principale d'adaptation. Il a été démontré que le nombre d'évaluations de fonctions et la vitesse de convergence du système dépendent de la relaxation du gain d'adaptation pour l'atteinte des points de fonctionnement. En outre, l'adaptation en ligne simultanée de multi-unités vers l'optimum global ne nécessite aucune interruption.

Un autre avantage de cet algorithme est l'absence de l'hypothèse de différentiabilité au long de ce travail. L'utilisation de la méthode présentée ne requiert pas nécessairement l'hypothèse de la différentiabilité du système. L'exemple 2.3.4 présenté au chapitre 2, introduit une caractéristique

non-différentiable statique à l'optimum global. La preuve de convergence pour cet algorithme est fournie en utilisant le formalisme mathématique contradictionnel. Les résultats des simulations ont confirmé les développements théoriques sur les nombreux exemples de référence pour l'optimisation globale. L'efficacité de la méthode et le nombre requis des évaluations de la fonction objectif ont été comparés aux algorithmes « DIRECT », « Genetic Algorithm » et « Simulated Annealing ». Les concepts théoriques sont illustrés par l'optimisation globale de plusieurs exemples. Les résultats de la comparaison avec d'autres méthodes ont confirmé la meilleure performance de la nouvelle technique en termes du nombre d'évaluations de la fonction. Il a été démontré que la méthode proposée permet de résoudre ce problème d'optimisation scalaire de manière très efficace.

Pour le cas scalaire, l'algorithme a été étendu au problème d'optimisation avec contraintes où une loi d'adaptation de « Switching » a été utilisée pour gérer les contraintes. Il a été démontré qu'une telle adaptation conduit à l'optimal global sous contrainte. La preuve de la convergence de cette méthode en utilisant cette logique a également été établie. Bien que cette méthode ne soit pas une commande extrémale en temps réel, c'est une stratégie d'optimisation « boîte noire », car elle utilise une méthode de commande extrémale comme un outil. L'algorithme utilise une procédure récursive comme un optimiseur en temps réel afin de converger vers l'optimum. Toutefois, cette récursivité s'arrête après la convergence vers l'optimum global puisque le décalage (Δ) entre les entrées multi-unités converge vers zéro à la fin de l'optimisation. Afin de maintenir le processus sur ce point ou de suivre l'optimal global qui est variable, d'autres stratégies de contrôle doivent être développées. Toutefois, il a été observé que l'algorithme proposé peut osciller (« chattering») lorsque la solution est sur la limite de la région réalisable. Des méthodes utilisant la projection pourrait en principe remédier à cette difficulté. L'extension de l'optimisation semi-globale pour des cas multivariables et sans contrainte constitue la prochaine étape de ce travail de recherche.

Dans le cas multivariable, le premier schéma a été étendu aux systèmes à deux entrées. Au chapitre 3, le développement de l'algorithme d'optimisation globale au long de la circonférence d'un cercle contracté est une autre contribution de cette thèse. L'idée de base de l'optimisation des systèmes monovariables est utilisée pour l'extension de l'algorithme à plus d'une variable. Cela a

été fait en répétant l'optimisation globale monovariable sur la circonférence d'un cercle de rayon réduit. La séparation de temps nécessaire entre la dynamique de l'adaptation itérative vers l'optimum global sur le cercle et la dynamique de la contraction du même cercle est requise pour cette méthode. Ce fait a été démontré à l'aide d'une preuve typique par contradiction pour le cas limite. Une méthode d'optimisation globale sans contrainte a été proposée en contrôlant le centre d'un cercle contracté sur lequel le gradient est estimé à partir des différences finies entre les entrées décalées de deux unités identiques. Le paramètre de décalage entre les entrées des deux unités est réduit à zéro de façon monotone et itérative lorsque le rayon du cercle est diminué en parallèle d'une manière monotone. Par cette démarche, il a été démontré qu'il est possible de converger vers l'optimum global d'une fonction objectif non-linéaire statique à deux variables, mais à condition que l'optimum global soit présent dans le premier cercle composé par le centre des entrées initiales et la valeur initiale du rayon.

L'efficacité de l'algorithme proposé a été démontrée à l'aide de trois exemples de référence. En outre, l'algorithme proposé a été comparé avec d'autres algorithmes déterministes et stochastiques afin de démontrer qu'il est efficace en termes du nombre d'évaluations de fonctions. Trois couches de l'algorithme itératif sont considérées :

Couche 1: Optimisation globale au long de la circonférence d'un cercle

Couche 2: Optimisation globale récursive

Couche 3: Réduction du rayon du cercle

Le développement de l'algorithme proposé pour les systèmes avec plus de deux degrés de liberté est considéré comme la prochaine étape de cette recherche.

Dans le chapitre 4, une nouvelle méthode est proposée afin de résoudre les problèmes d'optimisation globale sans contrainte avec trois variables. Cette technique est l'extension à trois dimensions de l'optimisation globale des systèmes multi-unités avec deux entrées. Le contrôle adaptatif de rotation du cercle de rayon variable sur un espace sphérique rétréci est la contribution principale du chapitre 4. En tournant le cercle variable sur la sphère, il couvre la région réalisable lorsque l'optimisation multi-unités a lieu. Cette réalisation a été développée sur la base des

concepts de rotation de cercle de rayon réduit sur un plan, tel que décrit dans le chapitre 3. Dans les systèmes à trois entrées, l'intégration des entrées est contrôlée de manière dynamique par une nouvelle formulation. Cette loi d'adaptation fait en sorte que le système multi-unités se déplace en direction d'une trajectoire circulaire rotative dans un espace sphérique qui rétrécit vers l'optimum global. Le mouvement de rotation sur le plan non linéaire est basé sur la différence de gradient entre les sorties du système multi-unités. Le rayon du mouvement circulaire est changé avec un taux spécifique et le rayon de l'espace sphérique est contracté à zéro. Une séparation d'échelle de temps assez grande doit être appliquée à la dynamique des lois d'adaptation dans les différentes couches de l'optimisation. Le principal défi dans le cadre de l'optimisation globale des systèmes multivariables est l'évolutivité de l'algorithme qui ne doit pas mettre en péril l'efficacité de son exécution pour atteindre une bonne performance. Cela signifie que plus le nombre de degrés de liberté d'un système augmente, plus la convergence vers l'optimum global est coûteux en calcul. Une discussion sur l'extension de l'algorithme à plus de trois variables est également présentée.

Dans le chapitre 4, l'algorithme d'optimisation globale avec l'adaptation multi-unités a été étendu aux systèmes avec trois variables. Le contrôle adaptatif de rotation du cercle de rayon variable sur un rétrécissement de l'espace sphérique est la principale contribution de ce chapitre. Le cercle du rayon variable en rotation couvre la région réalisable lorsque l'optimisation multi-unité a lieu. L'idée de base de l'optimisation des systèmes à deux entrées est utilisée pour l'extension de l'algorithme à des problèmes d'optimisation à trois variables. Dans les systèmes à trois entrées, l'intégration des entrées du système est contrôlée de manière dynamique par une nouvelle formulation. Cette modification des lois d'adaptation permet au système multi-unités de se déplacer le long d'une trajectoire circulaire rotative marquée par rétrécissement de l'espace sphérique vers l'optimum global. Le mouvement de rotation dans l'espace non-linéaire est basé sur la différence de gradient entre les sorties du système multi-unités. Le rayon du mouvement circulaire est changé avec un taux spécifique et le rayon de l'espace sphérique est réduit à zéro de manière simultanée. Une séparation suffisante de l'échelle de temps doit être appliquée à la dynamique des lois d'adaptation dans les différentes couches de l'optimisation. La dynamique de chaque unité est formulée de manière à ce qu'elle absorbe le mouvement des autres unités à un meilleur point de fonctionnement local sur le plan non linéaire. La nature déterministe de cette

approche garantit la convergence de l'algorithme à l'optimum global. Il a été démontré que la méthode proposée est efficace en termes de convergence précise à l'optimum. Les comparaisons avec les trois techniques d'optimisation globale dans cette classe (DIRECT, GA et SA), illustrent la performance compétitive de l'approche proposée. Cinq couches de l'algorithme itératif sont considérées pour l'optimisation des systèmes avec trois variables :

Couche 1: Optimisation globale le long de la circonférence d'un cercle en rotation sur une sphère à trois dimensions

Couche 2: Optimisation globale récursive le long du cercle

Couche 3: Expansion et la contraction du rayon du cercle

Couche 4: Optimisation globale récursive sur la sphère

Couche 5: Réduction du rayon de la sphère

Une discussion sur l'extension de la technique proposée à un plus grand nombre de variables est présentée. Les résultats de la comparaison pour l'optimisation multi-unité des systèmes en deux et trois entrées montrent que lorsque le nombre de variables est petit, la méthode multi-unités de l'optimisation globale est acceptable et parfois même supérieure aux autres méthodes concurrentes. Toutefois, comme le nombre de variables augmente, cette méthode peut rapidement devenir inefficace. La généralisation de l'algorithme à des dimensions supérieures est discutée. Les résultats montrent que l'évolutivité de la méthode est le principal défi dans le cas multivariable. L'impact de cet inconvénient devient de plus important avec l'augmentation de la taille du problème en termes de degrés de liberté. Cela pourrait réduire l'efficacité de l'algorithme. Cette situation découle du fait que beaucoup d'itérations sont faites de façon répétée et systématique sur les sous-espaces limités des cercles. Cela peut empêcher l'algorithme de converger sur l'optimum global selon les valeurs d'entrée initiales. Ainsi, l'algorithme de multi-unités ne peut pas sauter à l'optimum global (déterministe ou stochastique) comme les autres méthodes concurrentielles. Cela provoque des progrès limités systématiques vers le minimum global. Bien que cela puisse être considéré comme un inconvénient de cette méthode, la convergence vers un optimum global de cette technique est toutefois garantie. Certaines modifications ont été introduites pour permettre une mise à l'échelle de l'algorithme plus efficace. Une façon de minimiser est l'utilisation de plusieurs unités en cadre d'optimisation multi-unités

au lieu de seulement deux d'entre elles. La considération d'une unité supplémentaire flottant dans le centre du cercle contracté pourrait améliorer considérablement l'évolutivité de l'algorithme.

Enfin, cette thèse ouvre un nouveau domaine de recherche et expose plusieurs sujets relatifs à l'optimisation globale par la commande extrémale multi-unités. Sur la base des premiers résultats obtenus dans cette thèse, quelques idées originales sur la structure préliminaire de l'optimisation globale à l'aide de l'adaptation multi-unité ont été établies. Avec le travail effectué sur cette technique, la recherche future peut maintenant se concentrer sur les futurs progrès à réaliser avec cette méthode. Le dernier chapitre suggère des travaux qui sont liés aux résultats et les contributions présentées dans cette thèse. L'application expérimentale potentielle de cette méthode pour optimiser certains problèmes techniques et industriels est également discutée dans le dernier chapitre. Les contributions principales de cette thèse ont été identifiées comme suit:

1. La méthode classique d'optimisation multi-unité locale est étendue à la classe des techniques d'optimisation globale. Une méthode d'optimisation déterministe globale et indépendante des conditions initiales a été établie dans le cadre de commande extrémale multi-unités.
2. Un algorithme itératif d'adaptation à l'aide de commande extrémale multi-unités le long de la circonférence d'un cercle rétrécissant permet la convergence d'un système à deux entrées à l'optimum global.
3. Un contrôle de rotation adaptatif du cercle avec un rayon variable sur l'espace hyper-sphérique rétrécissant effectue l'optimisation globale des systèmes multivariables en utilisant la commande extrémale multi-unités.

Structure de la thèse :

Le chapitre 1 présente la revue de la littérature scientifique pour les méthodes d'optimisation globale « boîte noire », la commande extrémale en temps réel et l'optimisation multi-unités. Le chapitre 2 présente le nouvel algorithme pour les systèmes scalaires sans et avec contraintes et fournit les résultats principaux de convergence. Les effets de différents paramètres sur la convergence de la méthode sont également décrits dans le chapitre 2. De plus, la méthode d'optimisation globale établie est appliquée sur plusieurs exemples et enfin, une comparaison

avec trois autres méthodes d'optimisation globale est présentée. Le chapitre 3 porte sur l'extension de la méthode obtenue à l'optimisation globale des systèmes statiques avec deux entrées. La preuve mathématique de convergence pour l'optimisation globale le long de la circonférence d'un cercle réduit est présentée. Le chapitre 4 contient l'optimisation globale des systèmes avec trois variables d'entrée à l'aide d'adaptation multi-unités. Ces résultats sont également comparés avec les résultats obtenus en utilisant certaines des autres méthodes d'optimisation globale. Une discussion sur la généralisation de l'algorithme à des dimensions plus élevées est également présentée. Enfin, les conclusions de cette thèse, et quelques recommandations pour les travaux futurs sont abordées dans le dernier chapitre.

TABLE OF CONTENTS

DEDICATION III

ACKNOWLEDGEMENTS IV

RÉSUMÉ VI

ABSTRACT VIII

CONDENSÉ EN FRANÇAIS X

TABLE OF CONTENTS XXIV

LIST OF TABLES XXVII

LIST OF FIGURES XXVIII

LIST OF SYMBOLS AND ABREVIATIONS XXXI

LIST OF APPENDICES XXXIV

INTRODUCTION 1

CHAPTER 1 RESEARCH REVIEW 8

1.1 Black-box global optimization methods 8

1.1.1 Deterministic and Stochastic global optimization 10

1.1.2 Simulated annealing 11

1.1.3 Genetic algorithm 16

1.1.4 Lipschitzian optimization 18

1.1.5 DIRECT algorithm 20

1.2 Extremum seeking control 28

1.2.1 Local extremum seeking based on perturbations 30

1.2.2 Global extremum seeking based on perturbations 34

1.2.3 Local extremum seeking using multiple units 37

1.3 Summary 38

CHAPTER 2 GLOBAL OPTIMIZATION OF SCALAR SYSTEMS USING MULTI-UNIT EXTREMUM SEEKING 39

2.1 Unconstrained global optimization using multi-units 39

2.1.1 Schematic diagram 39

2.1.2 Convergence 41

2.2 Constrained global optimization using multi-units 43

2.2.1 Schematic diagram 43

2.2.2 Convergence 45

2.3 Illustrative examples 46

2.3.1 Application of global multi-unit optimization method 46

2.4 Comparison with other global optimization methods 55

2.5 Conclusion 59

CHAPTER 3 GLOBAL OPTIMIZATION OF TWO-INPUT SYSTEMS USING MULTI-UNIT ADAPTATION 61

3.1 Construction of the algorithm 61

3.1.1 Layer 1: Global optimization along the circumference of a circle 62

3.1.2 Layer 2: Recursive global optimization 63

3.1.3 Layer 3: Reducing the radius of the circle 64

3.2 Convergence 67

3.3 Illustrative Examples 69

3.3.1 Test problems 69

3.3.2 Application of global multi-unit optimization method 71

3.4 Comparison with Other Global Optimization Methods 75

3.5 Conclusion 81

CHAPTER 4 GLOBAL OPTIMIZATION OF THREE-INPUT SYSTEMS USING MULTI-UNIT ADAPTATION 82

4.1 Construction of the algorithm 82

4.1.1 Layer 1: Global optimization along the circumference of a rotating circle on a three-dimensional sphere 83

4.1.2 Layer 2: Recursive global optimization along the circle 84

4.1.3 Layer 3: Expansion and contraction of the radius of the circle 85

4.1.4 Layer 4: Recursive global optimization along the sphere 90

4.1.5 Layer 5: Reducing the radius of the sphere 90

4.2 Convergence 96

4.3 Illustrative examples 96

4.3.1 Test problems 96

4.3.2 Application of global multi-unit optimization method 98

4.4 Comparison with other global optimization methods 102

4.5 Discussion on extension of the algorithm to higher dimensions 109

4.6 Conclusion 113

CONCLUSIONS AND RECOMENDATIONS 115

BIBLIOGRAPHY 122

APPENDICES 129

LIST OF TABLES

Table 2.1 Comparison between global multi-unit optimization, DIRECT, genetic algorithm and simulated annealing 57

Table 2.2 Comparison between global multi-unit optimization with different η 59

Table 3.1 Comparison between MU and DIRECT in terms of percent error from the global optimum 77

Table 3.2 Comparison of GA and SA in terms of successful global convergence ($E\% < 0.01$) 77

Table 3.3 Comparison between global MU, DIRECT, GA and SA in terms of average number of function evaluations 78

Table 4.1 Comparison between MU and DIRECT in terms of percent error from the global optimum 105

Table 4.2 Comparison between GA and SA in terms of successful global conv. ($E\% < 0.01$) ... 105

Table 4.3 Comparison between global MU, DIRECT, GA and SA in terms of average number of function evaluations 106

Table 4.4 Comparison between MU and DIRECT in terms of percent error from the global optimum for the problems listed in appendix II 108

Table 4.5 Comparison between global MU and DIRECT in terms of average number of function evaluations for the problems listed in appendix II 108

Table II.1 Initial values and parameters chosen for tests problems in section 4.4 137

Table IV.1 Default values of the tuning parameters of Simulated Annealing in the optimization Toolbox 4.0 of MATLAB version 7.6.0.347 (R2009a) 140

Table IV.2 Default values of the tuning parameters of Genetic Algorithm in the optimization Toolbox 4.0 of MATLAB version 7.6.0.347 (R2009a) 141

LIST OF FIGURES

Figure 1.1 Flow chart of the basic simulated annealing algorithm 15

Figure 1.2 Shubert's algorithm.....20

Figure 1.3 Divisions of DIRECT algorithm in 2D.....22

Figure 1.4 Divisions of DIRECT algorithm in 3D.....23

Figure 1.5 Flow chart of the basic DIRECT algorithm.....26

Figure 1.6 Real time optimization procedure.....29

Figure 1.7 Perturbation-based local extremum seeking control (after Krstic and Wang, (2000)) .30

Figure 1.8 An example for static and continuous map.....32

Figure 1.9 Perturbation-based global extremum seeking control 35

Figure 1.10 Extremum seeking control with multiple units.....37

Figure 2.1 Global extremum-seeking control with multiple units 40

Figure 2.2 Constrained global extremum seeking control with multiple units 44

Figure 2.3 Static nonlinear map for example 2.3.1 46

Figure 2.4 Evolution of u_1, u_2, u and exponential Δ for example 2.3.1.....47

Figure 2.5 Evolution of u_1, $u2$, u and linear Δ for example 2.3.1.....47

Figure 2.6 Influence of η on integration time 48

Figure 2.7 Static nonlinear map for example 2.3.2 49

Figure 2.8 Static nonlinear map of example 2.3.3 50

Figure 2.9 Evolution of $f(u_1)$ and $f(u_2)$ in example 2.3.3.....50

Figure 2.10 Static nonlinear map of example 4 51

Figure 2.11 Evolution of $f(u_1)$ and $f(u_2)$ in example 2.3.4.....51

Figure 2.12 Static nonlinear map for example 2.3.5 52

Figure 2.13 Evolution of u_1, u_2, u and Δ for example 2.3.5 53

Figure 2.14 Nonlinear map and constraints for example 2.3.6 54

Figure 2.15 Evolution of u_1 and u_2 for example 2.3.6.....55

Figure 2.16 Zoom of figure 2.15 after convergence 55

Figure 3.1 Global optimization along the circumference of a circle.....63

Figure 3.2 Contraction of the circle toward the global optimum 65

Figure 3.3 Flow chart of the global optimization of two-input systems using multi-units 66

Figure 3.4 Ackley's function for example 3.3.1 70

Figure 3.5 Double Summation function with ω=1 (DS1) (inverted plot) 71

Figure 3.6 Double Summation function with ω=2 (DS2) (inverted plot) 71

Figure 3.7 Evolution of the inputs and Δ for example AC.....72

Figure 3.8 Evolution of the circles for example AC 72

Figure 3.9 Evolution of the centre with $T_{tot}/T=102$ (red line) and with $T_{tot}/T=10$ (blue line) for example DS1 73

Figure 3.10 Evolution of θ_m for example DS1 74

Figure 3.11 Evolution of the centre for exponential (red line) and linear (blue line) decreasing Δ for example DS2 74

Figure 3.12 Sampling points by DIRECT algorithm for AC 80

Figure 3.13 Sampling points by multi-unit optimization for AC 80

Figure 4.1 Global optimization along the circumference of a rotating circle on the sphere.....87

Figure 4.2 Contraction of the sphere towards the global optimum (top view) 93

Figure 4.3 Contraction of the sphere towards the global optimum (side view) 93

Figure 4.4 Illustration of repetitive expansion and contraction of the rotating circles on the sphere toward the global optimum ($\mathbf{u}_{mi}$) 94

Figure 4.5 Flow chart of the global optimization of three-input systems using multi-units 95

Figure 4.6 Evolution of the inputs and Δ for example Levy function 99

Figure 4.7 Evolution of the spheres for example Levy function 99

Figure4.8 Evolution of the centre of sphere with $T_{tot}/T=66$ (red line) and with $T_{tot}/T=10$ (blue line) for example Hartman function 100

Figure 4.9 Evolution of θ_m and φ_m for example Perm function 101

Figure 4.10 Evolution of the centre of sphere for $\boldsymbol{M}_{step\ size} = 1$ (red line) and $\boldsymbol{M}_{step\ size} = 5$ (blue line) for example Rosenbrock function 102

Figure I.1 Rotation of vector $\mathbf{v}_1$ about axis $\boldsymbol{\Gamma}$ 129

Figure I.2 Illustration of the basis vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ 130

Figure I.3 Illustration of the angle of rotation γ about the axis of rotation $\boldsymbol{\Gamma}$ 131

LIST OF SYMBOLS AND ABREVIATIONS

a :	The amplitude of dither in extremum seeking by perturbations
$C_i(.)$:	The ith constraint set
$E\%$:	Percent error
$f(.)$:	Objective function of an optimization problem
$f_{av}(.)$:	Average function in extremum seeking by perturbations
$f_{\min}$:	The best function value found by the global optimization algorithm
f_{global} :	The known global optimum of the test function
GA:	Genetic Algorithm
$g(.)$:	Positive locally Lipschitz function in extremum seeking by perturbations
k :	The adaptation gain that determines the rate at which the offset Δ is reduced
k_s :	The adaptation gain that determines the rate at which the radius Δ is reduced
k_θ :	The adaptation gain that determines the rate at which Δ_θ is reduced
k_φ :	The adaptation gain that determines the rate at which φ is reduced
$l(.)$:	The isolated root in extremum seeking by perturbations
$\mathbf{M}$:	Rodrigues' rotation matrix
MU:	Multi-Unit optimization
$M_{step\text{-}size}$:	Maximum step size of integration in multi-unit optimization
N_{pop}:	Number of population in genetic algorithm
ODE:	Ordinary Differential Equation
RTO:	Real-Time Optimization

SA:	Simulated Annealing
S :	Switching logic in constraint multi-unit system
$sign(.)$:	Sign function
$\tanh(.)$:	Tangent hyperbolic function
$T_{initial}$:	Initial temperature in simulated annealing
T :	One period of iteration
T_{θ}:	One period of iteration corresponding to the evolution of azimuth angle
T_{φ} :	One period of iteration corresponding to the evolution of elevation angle
T_s :	Required time for shrinking of the sphere
T_{tot} :	Total integration time
u_{ini} :	The initial input value of a static system
u_{ij} :	The ith input of the unit j
u_{im} :	The ith input of the global optimum (m)
u_k^* :	The kth local optimum input of a static system
u^{**} :	The unique global optimum input of a static system
$\mathbf{u}$:	The input vector of a static system
$\mathbf{v}$:	Axis of rotation
$\mathbf{y}$:	The output vector of a static system
$\dot{()}$:	Derivative
γ :	Angle of rotation
γ_{mi} :	Angle of rotation corresponding to the global optimum in i-1 iteration

$\mathbf{\Gamma}$: Axis of rotation

$\mathbf{\Gamma}_{mi}$: Axis of rotation corresponding to the global optimum in *i-1* iteration

δ : Strictly positive parameter in extremum seeking by perturbations

$\Delta(t)$: The variant perturbation parameter of multi-unit system

Δ_{ini} : The initial value of the variant perturbation parameter of multi-unit system

Δ_{θ} : The variant perturbation parameter (offset value) between to angles

ε : Small positive value

ε_{θ} : Small positive value

η : The tuning parameter in multi-unit system

θ_j : The azimuth angle of unit *j* in the three-dimensional polar system

θ_{mi} : The azimuth angle corresponding to the global optimum in *i-1* iteration

ρ : Time instant

τ : Time instant

φ_j : The elevation angle of unit *j* in the three-dimensional polar system

φ_{mi} : The elevation angle corresponding to the global optimum in *i-1* iteration

ω : The frequency of dither in extremum seeking by perturbations

LIST OF APPENDICES

Appendix I Rodrigues' rotation matrix 129

Appendix II Global optimization test problems 135

Appendix III Coefficients of test problems 138

Appendix IV Default values of the tuning parameters for stochastic algorithms 140

INTRODUCTION

Context

The economic profit of an industrial process (an objective function to be maximized) is typically a nonlinear function of different parameters such as operating conditions, the prices of products and raw materials. In general, this objective function may exhibit several local optima (maxima, minima and saddle points) among which the global maximum is typically sought. Herein, real-time optimization strategies bring and maintain a process at its optimal operating point. In this regard, extremum-seeking schemes are real-time optimization methods that control the gradient to zero. Most of these methods can converge only to the closest local optimum, though recently, some schemes have been proposed for global optimization of a restrictive class of nonlinear maps.

Finding the real time global optimal value of the control variable(s) of an industrial process which yields the best performance of the objective function has been always attractive in many engineering applications. The adaptation of the manipulated variables to their optima could be done based on mathematical models or by using experimental methods. Herein, many deterministic (such as Branch and Bounds and Lipschitzian) or probabilistic (such as random search and clustering) global optimization approaches have been significantly developed during the last decades (Floudas et al., 2008). The computational complexity, cost and the accuracy of these methods differ from one scheme to another. These classes of numerical methods are based on fundamental models and will not reach the global optimum of the true physical system as there is a model mismatch between the mathematical model and the reality. The experimental methods on the other hand are slower but accurate (Srinivasan, 2003).

When the only available information about the process to be optimized is the online input/output values, the system is so-called a “black-box”. In this case, without having an overall model of the system, it becomes impossible to use the above mentioned offline optimization methods in an effective manner. In a black-box scenario, the mapping function $f: U \rightarrow Y$ is treated as a black-box

like an executable procedure in a computer for which the programming code is not known or not accessible. In this case, it is difficult to determine whether a global optimal solution has been found, unless the entire decision space has been sampled. Black-box scenarios arise whenever the objective functions (1) are not given in closed form, i.e., if the objective function values are determined via complex computations, simulations, or experiments; or (2) are highly complex and/or poorly understood (Zitler, 2003). Herein, stochastic global optimization methods would have been used but the convergence to the real global optimum is not always guaranteed (Schneider, 2006).

On the other hand, model-free optimization methods have been studied under the name of extremum-seeking control, where the basic concept is to reformulate the unconstrained optimization problem as a problem of controlling the gradient of the objective function to zero. Though this method is quite old (Leblanc, 1922), it has received renewed interest recently (Ariyur & Kristic, 2003; Guay et al., 2004; Srinivasan, 2007). Also, many recent publications have reported successful applications (Ariyur & Kristic, 2003; Propović et al., 2003).

Extremum-seeking methods vary in their gradient estimation strategies. Two main gradient estimation methods will be discussed in this dissertation. The first is the traditional one (Leblanc, 1922; Ariyur & Kristic, 2003) where a temporal variation, i.e., a dither signal with constant, pre-fixed amplitude is added to input. The gradient is obtained as a correlation between the inputs and the outputs. As an alternative, Srinivasan (2007) proposed the multi-unit optimization method, where the gradient is computed based on the finite difference between a set of parallel units which operate with input values differing by a constant, pre-fixed offset. Herein, it was shown that time-scale separation is not necessary and that the multi-unit optimization could result in faster convergence (Woodward et al., 2009). The two above mentioned extremum-seeking strategies lead to the local optimum, since they are gradient-based. The local optimum that is reached depends on the initial conditions from where the optimization starts. So, the algorithm could be trapped in a local optimum instead of reaching the global one. Moreover, an optimum which is currently global can eventually become local or not even an optimum, if process parameters change (Lacks, 2003). These situations lead to inferior process performance and

provide strong motivation to develop global extremum seeking strategies. A global extremum seeking method based on perturbations has been recently analyzed by Tan et al. (2006 a, 2006 b). This has been an extension of their previous work on studying the semi-global stability properties of the extreme seeking controllers (Tan et al., 2005). The core idea is to reduce the amplitude of the dither signal asymptotically to zero. It was shown that although this strategy was successfully tested on a collection of nonlinear scalar problems, it is only applicable for a restrictive class of static maps.

Real-time constrained extremum seeking deals with online optimization of nonlinear functions under inequality constraints. Herein, barrier or penalty functions can be used to convert a constrained optimization problem into the unconstrained one (Dehaan and Guay, 2005). Also, projection of the gradient on the active constraints can be used to get to the constrained optimum (Woodward et al., 2007). A switching logic is required to determine the set of active constraints. However, both these methods only get to the local optimum.

The current research proposes a deterministic global optimization method that uses the spirit of real-time extremum seeking strategies in terms of controlling the gradient. However, it is not a real-time optimization method which follows the variation of global optimum permanently. In other words, real-time optimization in the sense of tracking the variable global optimum is a question that is not addressed in this study. From this point of view, this achievement can be sought as a "global optimization" strategy for the black-box systems. Towards this end, the illustrative examples are presented by some mathematical models of nonlinear systems as the objective functions. These benchmarks simulate the real processes to get the input/output measurement data and the gradient is exclusively estimated through the online input/output values. This is in contrast with the model-based optimization methods which use some mathematical properties of a virtual model in the gradient estimation procedure. In the presented algorithm, after data acquisition from the measured outputs, there is no need to estimate the unknown parameters of a model or to update the coefficients of a model as the classical real-time model-based optimizers. The acquired data will be directly used to optimize an objective function in order to find the new optimal set points which will be transferred to the control system in order

to implement in the process. That is the reason for classifying the algorithm as a model-free optimization strategy. It is shown that the presented algorithm converges to a very small vicinity of the global optimum of the static nonlinear continuous scalar maps. It is based on the multi-unit optimization approach and uses the concept of reducing the offset asymptotically to a small positive value. It is shown that with a minor modification of the adaptation algorithm, the algorithm converges to a small neighborhood of the global optimum without any preconditions on the nonlinear function. Also, relaxations of the algorithm are presented to make it numerically efficient.

Problem Statement

The general definition of the problem under question in this thesis is as follows:

Running a process with a local optimization method may cause the process to operate at a lower profit. This is because that the model-based local optimization is not always able to find the best operating set points. This fact is not only a problem of model mismatching, because it occurs even if the real process is modeled perfectly. Moreover, sometimes there is no appropriate model of the process and the only measurable data are input/output of the system (black-box optimization). Therefore a gradient based model-free global optimization strategy is needed in order to adjust the process on its best operating point. These issues form the main motivation of this research.

The specific definition of the problem is as follows:

The optimization problem considered in this study is the global optimization of a static and continuous system, where the objective function is non-convex. The manipulated variables can be estimated online based on available measurements. Moreover, the differentiability of the system is not necessary. The optimization problem may or may not contain inequality constraints and the appropriate measures of the constraints are also available. Finally, a priori knowledge of mathematical characteristics of the process is very limited, such that using a basic model in the optimization method is considered impossible. In this context, the global optimization using perturbation method can be applied (Tan et al. 2006 a). This method superimposes an

asymptotically decreasing periodic disturbance on process (known as the dither signal) in order to observe its effect on the objective function. A correlation between the measures of objective function and this disturbance can estimate the gradient in the steady state. The decreasing perturbation is temporary and a good estimation of the gradient in steady state requires different time scales between the frequency of disturbance, cutoff filters and adaptation. The speed of convergence in this method is commonly slow. As a result, using an alternative strategy that can eliminate the separation of time scales and accelerate the convergence to the optimum in this context would be very effective. Considering the above problem definition, the multi-unit optimization method (Srinivasan, 2007) - where the estimated gradient through the measurements are controlled to zero - provides the framework of this thesis. This method requires the presence of identical units to optimize the process. In the local based schematic of this method, the constant disturbance between the input values of the identical units is not a temporal offset. The gradient is then estimated by finite differences between the measures of objective function of these units. Modifications of this method in order to make it convergent to the global optimum form the main idea to solve the mentioned problem.

Main Objective

There has always been a debate on how efficient gradient estimation techniques of the continuous processes can be used for global optimization purposes without the intermediary of a model. The main objective of this thesis is the global optimization of the static and continuous nonlinear systems using tools from real-time local optimization. In this framework, the extension of the classical local-based multi-unit extremum seeking controllers to a model-free deterministic global optimization technique is considered.

Specific Objectives

The other specific objectives of this work are:

1. To diminish the restriction of the global extremum seeking controllers using perturbation methods in convergence to the false optimum.

2. To prove the convergence of the global optimization algorithm using multiple units.

3. To compare the results of this study with other global optimization algorithms in order to make more comprehensible the pros and cons of the presented algorithm.

4. To extend the algorithm to constrained optimization problems (in scalar case).

5. To determine the scalability impact of the extended method in multivariable optimization problems for higher dimensions.

6. To develop a gradient-based global optimization algorithm that is independent from the initial conditions.

Structure of the Thesis

The outline of this research dissertation is as follows:

Chapter 1 provides the literature review in the black-box global optimization methods, real-time extremum seeking controllers and multi-unit optimization. Chapter 2 presents the new algorithm for the unconstrained and constrained scalar systems and provides the main convergence results. The effects of different parameters on the convergence of the method are outlined. The established global optimization method is applied on several illustrative examples and finally a comparison with three other global optimization methods is presented. Chapter 3 deals with the extension of the obtained method to the global optimization of static systems with two inputs. The mathematical convergence proof has been provided for global optimization along the circumference of a shrinking circle. Chapter 4 contains the global optimization of the three variable systems using multi-unit adaptation. A discussion on generalization of the algorithm to higher dimensions has been also presented. These results are also compared with results obtained using some of the other global optimization methods. Finally, conclusions of this thesis and some recommendations for future work are addressed in the last chapter.

Contributions

The main contributions of this dissertation are as follows:

1. The local multi-unit optimization method is extended to a global optimization technique by reducing the offset parameter to zero.

2. In the two-input case, the same algorithm is used along the circumference of a shrinking circle in order to converge to the global optimum.

3. The method is extended to the optimization of three-input systems by performing two-dimensional optimization on the surface of a shrinking sphere.

CHAPTER 1 RESEARCH REVIEW

1.1 Black-box global optimization methods

The main purpose of optimization is to improve the profit or reduce the operating cost, which is typically expressed as a nonlinear function of different decision variables. Finding the global optimum of an industrial process has always been attractive in many engineering applications. Most of these problems are intrinsically multivariable. There are a growing number of problems in which optimization methods can be applied. Optimization algorithms have been developed in every discipline of engineering and science (both in theory and practice) for several decades. The common pattern in optimization research would be to design a specific algorithm given a very specific class of problems with some known properties. There is a large variety of different optimization algorithms which are applied to different problems. However, many of these optimization algorithms make assumptions about the properties of the objective functions which restrict their application. It is also important to note that the class of real-world optimization problems is often not easy to identify in order to apply the appropriate optimization algorithm.

On the other hand, optimization algorithms which use no further knowledge than the value of the objective function can be classified under the so-called "black-box optimization". In these approaches, the objective function may not be easily differentiable, may not have a closed functional form, and may require extensive computations in order to obtain them. With growing computational capabilities, designing such algorithms with little prior knowledge on the characteristics of an optimization problem is becoming more common.

For global optimization, two classes of methods can be distinguished, i.e., model-based methods and black-box methods. In model-based approaches (such as tight convex αBB underestimators for C^2-continous functions) have been significantly developed during the last decades (Floudas et al., 2008), where structural properties present in the nonlinear functions are exploited. Contrarily, "black-box" algorithms can handle the optimization of objective functions which may not be available as an explicit analytical expression. Given the set of manipulated variables, a "black-

box" provides the objective function to be optimized. Black-box scenarios arise whenever (1) objective functions are not given in closed from, i.e., if the objective function values are determined implicitly via complex computations, or simulations or (2) the model is highly complex and/or poorly understood (Laguna et al., 2010). The "black-box" optimization framework will be discussed further in this thesis.

Black-box optimization is addressed when there is a little domain of knowledge about the process. While some random enumerative search methods try to solve these problems, it is important to note that the sampling in black-box optimization is an inductive procedure (Kargupta, 1997). In the absence of any knowledge about the relation among the members of search space, induction is interpreted as a matter of guessing, based on what is known. A common ground to approach the black-box optimization is to make a framework which relates the members of the search space. The performance and scope of many black-box optimization algorithms like genetic algorithm (Holland, 1973), simulated annealing (Kirpatrick, Gelatt and Vecchi, 1983) and tabu search (Glover, 1989), differ from one another on many aspects. In black-box optimization, the objective function is available as a black-box meaning that for a given x in a search domain it returns the function value $f(x)$. The general black-box optimization can be formally defined as follows (Kargupta and Goldberg, 1997),

$$f : X \rightarrow Y \tag{1.1}$$

where X denotes the set of finite-dimensional input space and Y being the real line. For a given input x, the black-box $f(x)$ is computed. The main purpose of a minimization problem is to find some $x^* \in X$ such that $f(x^*) \leq f(x)+\varepsilon \;\; \forall x \in X$ and $\varepsilon>0$ is a small arbitrary constant. The next sections deal with problems in the following form,

$$\begin{aligned} &Min \qquad f(x) \\ &s.t. \quad x_L \leq x \leq x_U \end{aligned} \tag{1.2}$$

where $f \in \mathbb{R}$ is a "Lipschitz" continuous function (Jones et al., 1993) and $x, x_L, x_U \in \mathbb{R}^n$. Here x_L, x_U indicate the lower and upper bounds of the feasible region for the variable x respectively.

The global optimization has a rich volume of literature. Many of these methods are restrictive because of their assumptions about the properties of the objective function (Schoen, 1991). The global optimization methods which take into account prior information about the characteristics and structure of the objective function cannot be considered as black-box optimization. On the other hand, two kinds of algorithms can be considered for black-box global optimization techniques: deterministic and non-deterministic (stochastic) methods.

1.1.1 Deterministic and Stochastic global optimization

Many earlier efforts in global optimization suggest to classifying these methods based on their deterministic or non-deterministic (stochastic) nature (Archetti and schoen, 1984; Dixon and Szegö, 1978). Global optimization methods with black-box objective functions can also be categorized based on this approach. However, there are other ways to classify the optimization algorithms from different points of view (Törn and Žilinskas, 1989; Vavasis, 1991).

The stochastic methods try to solve the problem by introducing some random elements in their algorithm. The main reason for the popularity of the stochastic algorithms is that they scale well with dimension, i.e., for higher dimensions deterministic algorithms become prohibitively expensive. However, stochastic methods do not always guarantee convergence to the global optimum. The Monte Carlo and multi-start algorithms are two typical types of blind random search methods (Schoen, 1991; Törn and Žilinskas, 1989). Bayesian algorithms (Betrò, 1983), clustering methods (Rinnooy and Timmer, 1987), simulated annealing (Kirpatrik et al., 1983), genetic algorithms (Holland, 1973; Goldberg, 1989) and evolutionary strategies (Rechenberg, 1973) are examples of the adaptive sampling search techniques which have been extensively used for solving black-box optimization problems. These algorithms try to exploit the information gathered from samples from search space. A good survey and critical evaluation of these methods can be found in Kargupta and Goldberg, 1996. The Monte Carlo algorithm simply compares the random samples generated from the feasible space with a fixed distribution. Multi-start global optimization method combines the local optimization techniques with Monte Carlo sample

generation method. Bayesian algorithms exploit the information needed for optimization from sampling points by developing a statistical model of an objective function. In this method an implicit objective function is constructed based on a random variable. The expected value of this random variable minimizes the expected deviation of the estimated global optimum from the real one. Bayesian algorithms are fairly complicated and need expensive computations which involve the inverse of the covariance matrix. Clustering methods combine the Monte Carlo sample generation technique with the cluster analysis algorithms to identify local minima which is followed by a local search for each of them. This algorithm performs poorly in multivariable systems or in optimization of a function with many local optima.

On the other hand, deterministic algorithms systematically search the feasibility region based on information gathered about the objective function. DIRECT is a deterministic algorithm which uses "Lipschitz" constant arguments (from zero to infinity) to decide which regions of the search space are worth to explore at each iteration (Jones et al., 1993). This way, the algorithm explores the search region efficiently by focusing extra function evaluations only in regions which seem "potentially optimal". Some reports indicate that in some particular problems, the deterministic algorithm DIRECT is more reliable than other competing stochastic methods (Bartholomew-Biggs et al., 2003)

In the following sections a brief review of some previous efforts to solve the black-box global optimization problems are presented. Their pros and cons along with the similarities and differences between these competing methods will be discussed. This gives a common idea to relate and investigate these kinds of algorithms from different points of view.

1.1.2 Simulated annealing

Classical local optimization methods start from an initial point and move around this point such that the objective function value is always improved. This procedure stops when the optimization algorithm cannot move to a better point. This is the reason for convergence of the algorithm to the nearest local optimum to the initial point. Different optimal solutions can be found depending

on the starting point. In order to find the global optimum of a function, the algorithm must have the capability to occasionally move in a direction which makes worse the objective function value. This is essential to move over the local optima and discover the other areas of the search space. In other words, during the movement in the search space the algorithm should choose the worse points with a particular probability.

Simulated annealing (SA) is an optimization technique which has been significantly discussed in black-box optimization framework. This algorithm was motivated based on the mathematical simulation of the statistical behavior of molecules during the crystallization process in annealing of solids. The molecules of a solid metal have a certain potential energy among themselves. Heating the metal increases the kinetic energy of the molecules. As a result, they can move around and locate to the new different states with respect to the other molecules. The total set of molecules have a tendency to settle down in a position (state) in which the system has the lowest energy. In the beginning of the annealing process, a solid material is heated to its melting point. Then, the solid is allowed to cool gradually enough such that thermal equilibrium is maintained. The cooling process with a slow rate arranges the molecules in the solid material as the minimum energy state is attained. In other words, cooling process gives the ability of experimenting different states to the system (even the states which increase the potential energy). This way, the system would reach a state which has a lower potential energy compared to its initial state. The system eventually converges to a stable state by decreasing the kinetic energy and movement of the molecules. The important point to note is that the final crystallized state of the molecules is completely independent from their initial states.

This natural phenomenon first was applied to numerical calculations by Metropolis et al. (1953) and then it is applied to engineering (optimization of VLSI circuit design) by Kirpatrick et al. (1983). The states of the molecules and the potential energy in annealing process are corresponding to the sampling points of the search space and the penalty function in the simulated algorithm respectively. The ability of the algorithm to move towards the worse solutions is attributed as the kinetic energy of the molecules. The main task of such an algorithm would be move the state of a system with a certain initial potential energy and arbitrary kinetic

energy to a new state in which the potential energy is the lowest. In the final state, the kinetic energy would be zero too and the system reaches the stable equilibrium.

In simulated annealing an arbitrary initial point is chosen and the penalty function is calculated at this point. The final result of the algorithm should not be influenced by the starting point since it is a global optimization method. An initial temperature (T_0) is considered corresponding to the kinetic energy. The choice of initial temperature is arbitrary, but it can be chosen based on the behavior of the function at the starting point. For instance, if the function has less variation it would be better to choose a small T_0 in order to limit the ability of the extra variation in the algorithm. Otherwise, a larger T_0 is chosen to increase the ability of movement from the pitfalls of local optima. A new point is then generated using the sampling points in the neighborhood of the initial point. The state of the algorithm is represented by one design at a time. During the next generations, if the value of the objective function corresponding to the new design is better than the old one, it is accepted and the algorithm moves to this state. Otherwise, the algorithm moves to the new point by the probability of (P). As a result, the worse point may or may not be accepted. The new point is accepted or failed as the new state of the algorithm based on the following probabilistic comparison (Kirpatrick et al., 1983),

$$P = \begin{cases} \exp(-\dfrac{U - U_0}{T}) & if \quad U > U_0 \\ 1 & if \quad U \le U_0 \end{cases} \tag{1.3}$$

P is the probability of the movement to the new state, U_0 and U correspond to the potential energy at the old and new points respectively and T is the temperature proportional to the kinetic energy. The above probability function for decision making to move to the new state has been inspired from the Boltzmann probability function ($P(E) = exp(-E/K_BT)$) which indicates the probability of the specified state with energy E and temperature T (Metropolis et al., 1953). K_B is the Boltzmann constant.

If the Boltzmann probability (P) is more than a randomly generated number between zero and one, the worse design will be accepted. ΔU represents the change in the objective function and

the temperature parameter (T) dynamically sets the Boltzmann probability (P). This function is defined such that the probability of the movement to a higher potential energy (worse state) is proportional to the temperature value, i.e., the larger is T, the smaller would be ($\Delta U/T$) and P will be larger accordingly. On the other hand, the slope of the function or the rate of the change of the function in the movement direction (ΔU) is another parameter which affects P. In other words, the larger is the difference between the potential energies of the original and destination states, the smaller would be the probability of this movement. After each movement, temperature T is reduced a little bit. (The kinetic energy is reduced a little bit). The choice of cooling schedules (the rate of reduction for T) is not completely arbitrary and it must be chosen such that the global convergence is guaranteed. This choice may vary from one problem to another. After several iterations and reducing temperature T to zero (i.e., losing the kinetic energy completely) it is supposed that this algorithm converges to the lowest penalty function (the lowest potential energy) which is corresponding to the global minimum of the function.

Choosing a low temperature with a very fast cooling rate in SA, would lead the algorithm to the closest local minimum from the starting point. Therefore, the initial value of the temperature parameter must be chosen high enough while it gradually decreases according to a cooling schedule. Furthermore, in order to reach thermal equilibrium, it is suggested that the temperature parameter T is held constant at the starting temperature (T_0) for a few number of first iterations. For example, the number of initial iterations in which T is held constant is set at [15×number of variables] as in Vanderbit and Louie (1984). The basic structure of the simulated annealing algorithm is presented in the following flowchart,

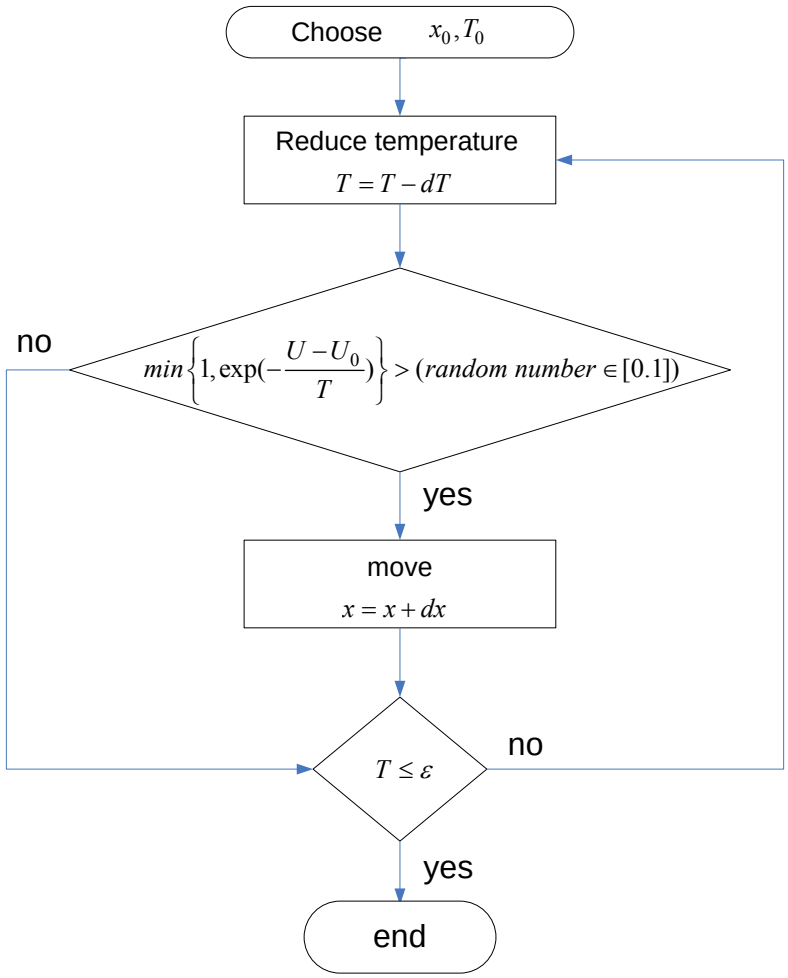

Figure 1.1 Flow chart of the basic simulated annealing algorithm

Kirpatrick et al. established a proof for asymptotic convergence of this algorithm to the optimal solution. Simulated annealing can be also used to optimize non-continuous systems and functions with discrete inputs. Although simulated annealing has introduced a promising contribution in global optimization, some negative results have been reported in literature (Ferreira and Žerovnik, 1993). This algorithm needs a large amount of computational load and several iterations in order to ascertain the convergence to the global minimum. As the number of variables grows, simulated annealing becomes very slow and the time taken to ascertain the global convergence increases drastically. Another disadvantage of simulated annealing is that many attempts to speed up the algorithm (e.g., parallel versions of the algorithm) are highly problem-dependent because of the nature of the algorithm (Ram et al., 1996).

Simulated annealing has been widely applied to a variety of problems such as: optimizing the placement of the elements on integrated circuits, the famous traveling salesman problem, time scheduling, resource allocation problems and data mining.

1.1.3 Genetic algorithm

Genetic algorithms (GA) have received significant attraction in the black-box global optimization research area. The simple genetic algorithm was motivated by natural selection process in the evolution of living organisms. The core idea in this algorithm is the survival of the fittest members in a sampling population with the greatest probability (Holland, 1973, De Jong, 1975 and Goldberg, 1989). Holland's work (1975) has been the foundation of many new developed algorithms in this framework. These algorithms, along with evolutionary strategies (Rechenberg, 1973) and evolutionary programming (Fogel, Owens and Walsh, 1996) are the probabilistic global optimization methods dealing with black-box problems.

Genetic algorithms work with a population of samples instead of a single point at a time. Initial population members are chosen randomly. Each sample is represented as some sequences. In other words, the population members are generally represented by binary strings in the written code. These binary strings are called "Chromosomes". The characteristics of the optimization variables (members) are coded in these binary strings which are known as the "Genes". A fitness function which is based on the objective function evaluates each member of the population. "Mutation" and "Crossover" operators generate the new samples and members for the next generation are chosen based on their fitness values. The probability of being chosen for the next generation is proportional with the fitness value corresponding to that member. The exchange of bits of the binary strings of parents to create a new child is called "Crossover". Genetic algorithms treat combination of two existing solutions (parents) as a new candidate (child) which is assumed to be a good solution with more probably found near already found good solutions. This assumption seems to be consistent since such combinations (children) logically inherit the properties of their parents. This is the key factor which makes a big difference between genetic

algorithms with random search. A child of two good solutions is more probably good than a random solution. “Mutation” is the random change of a bit in a binary string from 0 to 1 or 1 to 0. The occurrence of these operators in the algorithm depends on a given probability.

The simple GA has some restrictive disadvantages such as noisy evolution of schemata. The similarities and differences between genetic algorithms and Bayesian approaches have been studied by Jones et al. (1992) and a hybrid algorithm has been produced by their combination. A parallel genetic version of simulated annealing that tries to employ the strengths of both GA and SA is called parallel recombinative simulated annealing (Mahfoud and Goldberg, 1992). The global convergence behavior of evolutionary algorithms and simulated annealing has been studied by Hart (1996).

The fundamental similarities and differences between the genetic algorithm and simulated annealing has been studied by Goldberg (1990) and Kohonen (1999). Although both approaches are closely related, however, their terminology is quite different. SA deals with *solutions, costs, neighbors* and *moves* while in GA there is a different mathematical indication such as *chromosomes, fitness function, crossover* and *mutation*. These techniques do not require the evolution of the gradient information in order to solve the problem. Instead, both of these methods share the idea that good solutions are more probably near the already known good solutions rather than randomly selected samples. If the population size in GA is considered as only one, this algorithm can somehow be thought as SA. On the other hand, while SA deals with one solution at a time, GA creates multiple solutions by combining different parents.

Some attempts have been reported to compare these algorithms. Manikas and Cain (1996) compare 20 trials with each algorithm on a circuit partitioning problem without taking account on the execution time used. Their results show that GA produces solutions equal or better than SA. Mann and Smith (1996) reported that the execution times of SA were always shorter than of the GA (10 to 24 times) on solving a particular traffic routing problem. It was indicated that SA is a good “quick starter” that is able to converge to good solutions quickly, but it is not able to improve it given more time. GA is a good “slow starter” which can improve the solution given

more execution time. In other words, SA usually gives better solution than GA given the same amount of execution time. In general, the answer to the question that whether SA is better or worse than GA is not straightforward and depends on the characteristic of the working problem.

1.1.4 Lipschitzian optimization

In this thesis, a new deterministic algorithm for two-dimensional global optimization will be studied. The most appropriate algorithm with which it can be compared is the method termed "DIRECT" which is a deterministic sampling approach based on the space-partitioning scheme (Jones et al., 1993). No derivative information is used in this algorithm. DIRECT is the shortening form of two words "Dividing RECTangles" which indicates the way this algorithm moves toward the global optimum. The algorithm divides large bounded hyper-rectangles into a collection of smaller ones where the center of each hyper-rectangle is evaluated via the objective function. A set of potentially optimal hyper-rectangles are selected for further partitioning and divisions in the next iterations. In this section, this algorithm is explained in detail.

DIRECT is a modification of a standard "Lipschitzian" approach that eliminates the need to know the Lipschitz constant (Jones et al., 1993). This method was created for cases with bounded domains and real-value objective functions. In the rest of this section first a brief introduction to Lipschitzian algorithms and their shortcomings is introduced. Then the DIRECT algorithm which is motivated to overcome the drawbacks of the Lipschitzian optimization is discussed. The core ideas of the DIRECT algorithm are summarized here. The proof of convergence was presented in Jones et al., (1993). Detailed and good surveys of this algorithm can be found in Finkel, (2005) and Björkman and Holmström, (1999).

The Lipschitzian algorithms are the iterative algorithms that use the Lipschitz constant information (K) of a continuous function to seek its minimum. A function $f : X \rightarrow Y$ is called Lipschitz-continuous on X if there exists a positive constant K such that,

$$\begin{gathered} \left| f(x) - f(x') \right| \le K \left\| x - x' \right\|, \quad \forall x, x' \in X \\ f(x), f(x') \in Y \\ K \in \mathbb{R}^{+} \\ X \subseteq \mathbb{R}^{n} \quad , \quad Y \subseteq \mathbb{R} \end{gathered} \tag{1.4}$$

In standard Lipschitzian methods the maximum rate of change of the objective function must be always equal to (or less than) the known Lipschitz constant of the function. This constant is usually large which emphasizes the effort of the algorithm on global search. This makes these algorithms converge slowly. If the Lipschitz constant K of a function is known, this information can be used to construct Lipschitzian global optimization methods. One of the most straightforward type of these methods is the Shubert algorithm (Shubert, 1972). If we assume that $x \in [x_L, x_U]$ and we substitute x_L and x_U from (1.2) for x' into the definition of the Lipschitz continuity (1.4), the following two conditions for $f(x)$ are obtained,

$$f(x) \ge f(x_L) - K(x - x_L) \tag{1.5}$$

$$f(x) \ge f(x_U) + K(x - x_U) \tag{1.6}$$

The lines corresponding to these two inequality conditions form a V-shape below $f(x)$ as figure 1.2. The intersection point of two lines which provides the first estimate of the minimum f is then easily calculated as follows,

$$x_1(x_L, x_U, f, K) = \frac{f(x_L) - f(x_U)}{2K} + \frac{x_L + x_U}{2} \tag{1.7}$$

$$f_1(x_L, x_U, f, K) = \frac{f(x_L) + f(x_U)}{2} + \frac{K(x_L - x_U)}{2} \tag{1.8}$$

The algorithm continuous performing the same operation of subdividing on the new regions $[x_L, x_1]$ and $[x_1, x_U]$. The subsequent subdivisions are performed in the regions with intersection point corresponding to the lower function value. Figure 1.2 shows this process for two iterations.

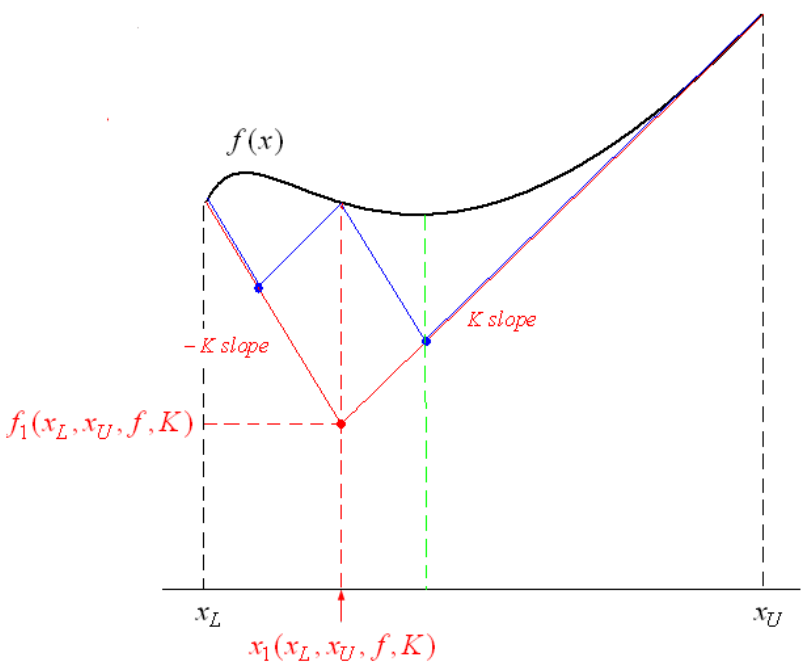

Figure 1.2 Shubert's algorithm

One of the drawbacks of shubert's algorithm is that the Lipschitz constant must be known to perform this method. K might not be easily accessible. Not only for the Lipschitz-continuous functions this constant cannot be estimated properly but also many applications may not be even Lipschitz-continuous on their space. Moreover, a poor estimation of K can lead to a poor result of the algorithm. If the estimation is too small, the final result may not be the global optimum of f and if the choice is too large the convergence of this algorithm will be very slow (Finkel, 2005). The other shortcoming of this method is that since the idea of end-points cannot be translated well into higher dimensions, generalization of the algorithm for multivariable optimization is intuitively impossible.

1.1.5 DIRECT algorithm

DIRECT needs no prior knowledge of the Lipschitz constant of the problem. This algorithm even does not require the objective function to be Lipschitz continuous. DIRECT solves the

optimization problem by carrying out simultaneous searches using all possible constants from zero to infinity. This method uses all possible Lipschitz constants, during each iteration. As a result, DIRECT operates at both the global and local search levels simultaneously. This makes the convergence of the algorithm very fast. In the DIRECT algorithm the unknown "Lipschitz" constant is viewed as a weighting parameter. This weighting parameter balances the emphasis which must take place on global versus local search. In other words, K implicitly serves as a trade-off between global and local search. As will be discussed further, the DIRCET algorithm repeatedly samples at the midpoints of the search space. This way the confusion of the generalization of the Lipschitzian algorithms to higher dimensions is removed.

The DIRECT algorithm begins with a given n-dimensional "hyper-rectangle" as the initial search space. First of all the search space is normalized by transforming the search domain of the problem into a unit imaginary domain called a hyper-rectangle,

$$\bar{X} = \left\{ x \in \mathbb{R}^n : 0 \le x_i \le 1 \right\} \tag{1.9}$$

In DIRECT algorithm, the sampling is done at the central point of the intervals (hyper-rectangles). This way the computational complexity arising in higher dimensions is removed. Similar to the scalar case, subdivision procedure of the algorithm consists of trisecting (shrinking) the unique longest edge (side) of an existing hyper-rectangle j characterized by (c_j, f_j) to three equal sections. As a result, DIRECT replaces each hyper-rectangle by three new smaller hyper-rectangles. Each hyper-rectangle is defined by its center point "c_0" where the objective function is $f_0 \equiv f(c_0)$. In other words, the objective function is evaluated at the center of the hyper-rectangle (c_0) and then at points $c_0 \pm \delta e_i$, where "δ" is one third of the side length of hyper rectangle, and "e_i" is a unit vector with a one in the ith position and zeros elsewhere ($i=1,\ldots,n$ represents each dimension). The initial normalized hyper-rectangle is then repeatedly split into smaller hyper-rectangles by subsequent subdivisions. The fact that all divisions are restricted to only being done along the longest dimension(s) of the hyper rectangle ensures that the rectangles will shrink on every dimension. The hierarchy of such a division pattern is explained here. The subdivision is a trisecting process in each hyper-rectangle. This division is done in a manner that

the best objective function value evaluated in the centers of these hyper-rectangles is left in the largest hyper-rectangle. In other words, by defining the following criteria,

$$w_i = \min(f(c_0 + \delta e_i), f(c_0 - \delta e_i)) \qquad i = 1,\ldots,n \tag{1.10}$$

the division is first done on the dimension with the smallest w_i into third parts. Therefore, $c_0 \pm \delta e_i$ are the centers of the new divided hyper-rectangles in this dimension. After the first division, the above mentioned pattern is repeated for all dimensions on the "center hyper-rectangle" hierarchically. The next smaller w_i determines the next dimension to be divided. In other words, DIRECT subdivides along directions with best function values first. This way the largest rectangles contain the best function values. This process of division is illustrated for the imaginary objective function values at the center of divided hyper-rectangles for a 2D domain in figure 1.3.

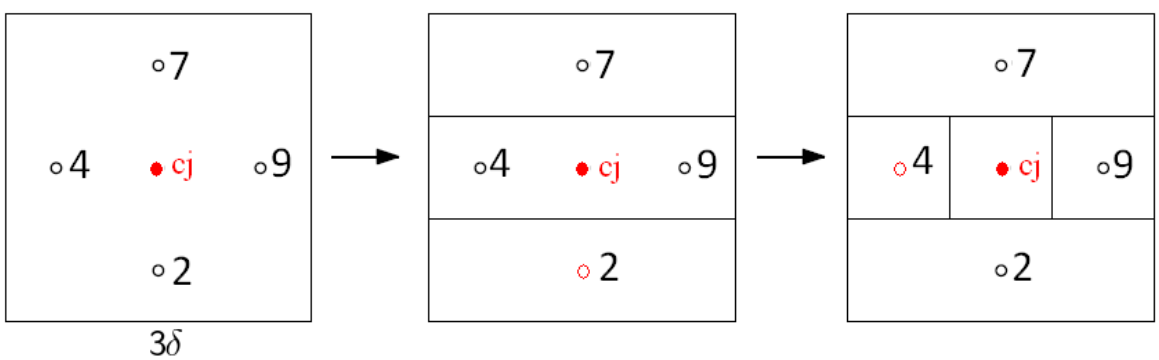

Figure 1.3 Divisions of DIRECT algorithm in 2D

This way, the rectangles are subdivided along their longest dimension. In more than two dimensions the rectangles become hyper-cubes and DIRECT similarly divide them along the set of longest sides for all dimensions (Figure 1.4).

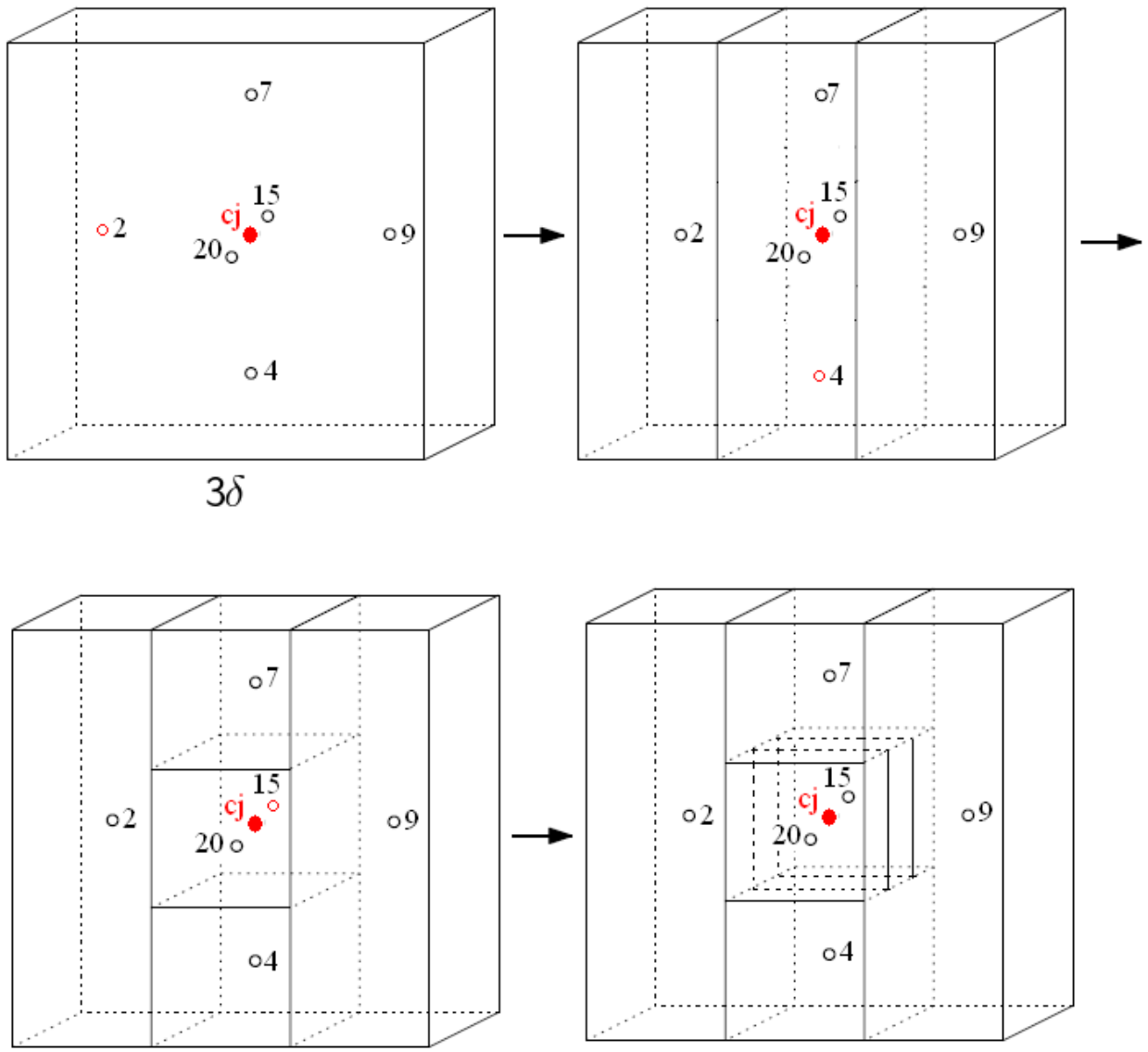

Figure 1.4 Divisions of DIRECT algorithm in 3D

The key point in DIRECT algorithm is that instead of using a Lipschitz constant for determining the candidate rectangles to sample next, it identifies a set of potentially optimal hyper-rectangles in each iteration. All potentially optimal hyper-rectangles are further split to smaller hyper-rectangles and the objective function value f is evaluated in their center-points. In other words, rectangles (cubes) which have the potential to contain the global optimum are identified within the iteration and make the set of next candidates (S) for further subdivision into smaller

rectangles (cubes). S can have "m" different members during different iterations depending on the objective function. Within each iteration, the objective function can be evaluated several times. A hyper-rectangle $s \in S$ is called "potentially optimal" if there exist a rate of change constant ($\tilde{K} > 0$) such that the following conditions are satisfied,

$$1) \quad f(c_j) - \tilde{K}d_j \leq f(c_s) - \tilde{K}d_s \qquad \forall s \in S \tag{1.11}$$

$$2) \quad f(c_j) - \tilde{K}d_j \leq f_{\min} - \varepsilon \left| f_{\min} \right| \tag{1.12}$$

Here, c_j and d_j denote the centre point and "size" of the jth hyper-rectangle respectively where $j \in J$ and J is the set of all of the hyper-rectangles. d_j is simply the distance from the centre point to the vertices of hyper-rectangles. In other words, the hyper-rectangles are grouped according to the distance from their center to their corner. This measuring criterion of the hyper-rectangles is known as their size. ε is a positive constant. f_{min} is the global optimum found in all iterations so far.

For each hyper-rectangle j, the algorithm checks whether there exists any "Lipschitz" constant such that it *could* contain a lower objective function value than other rectangles. This is interpreted as the local search of the algorithm. If there exists such a Lipschitz constant, that particular rectangle is considered to further subdivisions. Otherwise, rectangle j is not worth to being split more on the current iteration. In other words, the algorithm predicts the regions that must be avoided to further search and this way focuses on the worthy regions as long as the algorithm proceeds. This can be interpreted as the global searching aspect of the algorithm. While the first inequality condition emphasizes the effort of the algorithm on local search in hyper-rectangles with the same size, the second inequality condition prevents the algorithm from becoming too local by comparing the optimality criteria with the best optimum found so far. In other words, the second condition prevents the algorithm from doing numerous function evaluations in local level which yield only small improvements. As a result, some smaller rectangles are not selected for further division during consecutive iterations. In general, there may be more than one potentially optimal hyper-rectangle found during an iteration. Once the set of potentially optimal rectangles (S) is chosen, DIRECT divides them into smaller units according to the explained pattern. The algorithm stops when a predefined budget of number of iterations or divisions is completed.

The drawback of this algorithm is that the global convergence may cost a large number of function evaluations in the feasible space. If there are only a few local minima, DIRECT uses a lot of unnecessary function evaluations to explore unvisited territory. Usually the algorithm is performed for a predefined number of iterations. This algorithm generally requires a few pre-defined parameters to run such as the maximum number of iterations and the maximum number of hyper-rectangle divisions. The basic structure of the DIRECT algorithm is depicted in the following flowchart,

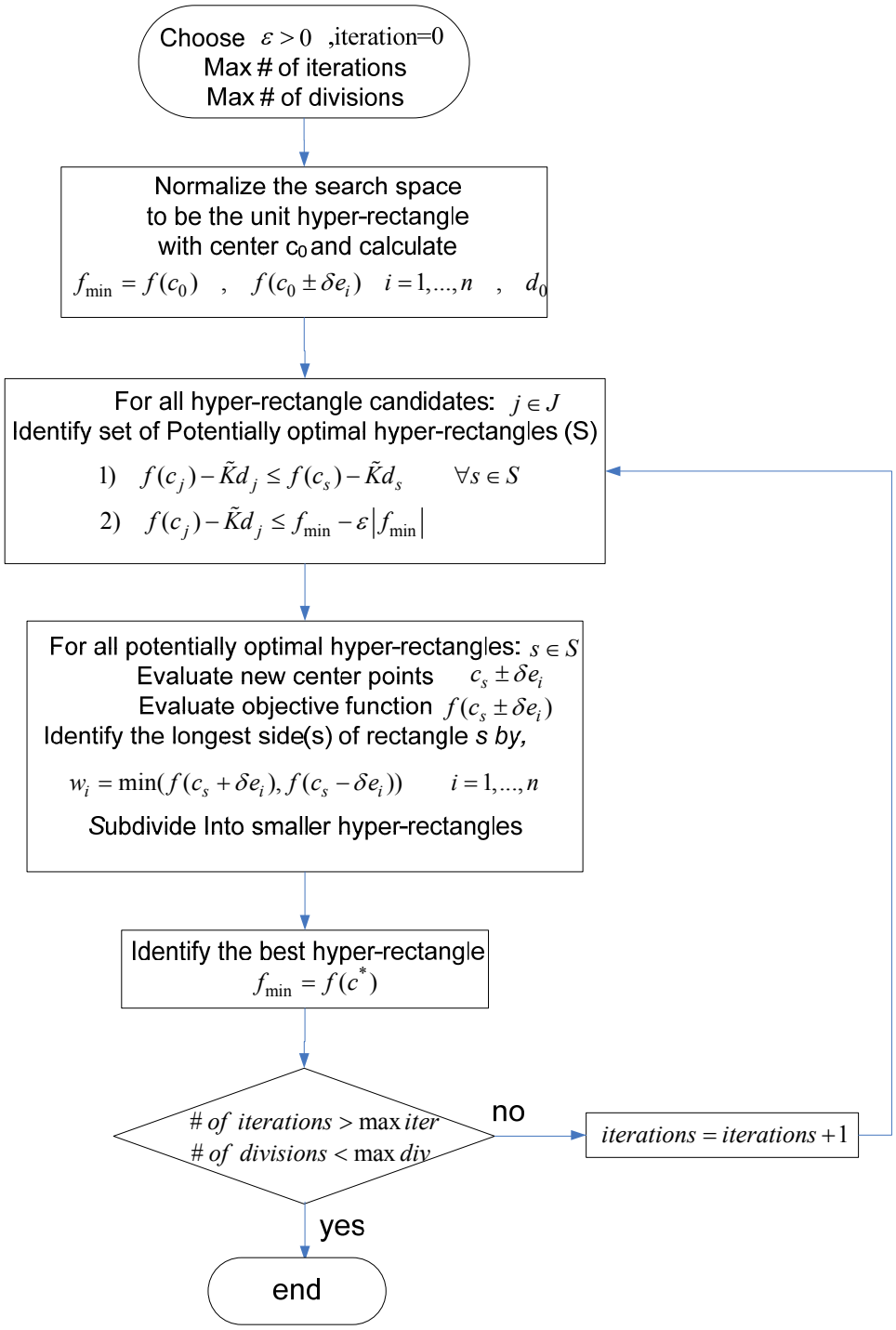

Figure 1.5 Flow chart of the basic DIRECT algorithm

Since no gradient information is used, there is no natural way for defining the global convergence in this algorithm. However, the convergence of the DIRECT algorithm to the global optimum is

guaranteed for continuous objective functions (or at least continuous in the neighborhood of the global optimum) if the number of iterations goes to infinity (Jones et al., 1993). This is consistent since as the number of iterations goes to infinity, a very dense subset of the hyper-rectangles is formed by the sampling points. Finkel (2005) developed the rate of convergence analysis for this algorithm.

A number of other versions of this algorithm have been suggested to modify the basic one. DIRECT-1 is another version of this algorithm which uses *aggressive searches* in which the potential optimality test is omitted on some iterations and all K_j candidate hyper-rectangles are subdivided (Gablonsky, 2001). In DIRECT-1 if $50n$ hyper-rectangle subdivisions are done without any significant reduction in the best value of the objective function an aggressive search occurs. DIRECT-2 is another version of the algorithm in which the trisection process takes place along only one edge thereby being less expensive per iteration. However, it may need more iteration to ascertain the global convergence. In DIRECT-1 a lower limit on the subdivided hyper-rectangle sizes has been considered ($\delta_j < 10^{-3}$) as the stopping criteria for the algorithm. DIRECT can be performed in restart mode, i.e., instead of running "*i*" consecutive iterations, we stop the algorithm after "*i /2*" or "*i /4*" iterations and restart it in a new hyper-rectangle centered on the best point found so far. It was reported that sometimes DIRECT leads to a better point with a lower computational cost in restart mode (Bartholomew-Biggs et al., 2003). Re-centering the initial search on a good estimate of solution not only reduces the number of candidate hyper-rectangles that have to be considered over the "*i*" consecutive iterations, but also it reduces the number of potentially optimal hyper-rectangles that are identified after subsequent restarts.

Many applications of this algorithm can be found in literature. Bartholomew-Biggs et al. (2003) applied DIRECT algorithm to solve an aircraft routing problem. Another application of this method was to optimize the slider air-bearing surface in design of hard-drive heads (Zhu and Bogy, 2002). Finkel (2005) applied this method to minimize the installation and operational cost in a groundwater optimization problem. DIRECT was also employed to optimize the gas pipeline networks by Gablonsky (2001). Rapid and robust phase behavior stability was analyzed using DIRECT by Saber and Shaw (2008).

1.2 Extremum seeking control

Real-time optimization with "black-box" objective functions has been addressed by using extremum-seeking techniques. In these methods, the unconstrained optimization problem is cast as a problem of controlling the gradient of the objective function to zero. For estimating the gradient, many techniques have been used: perturbations (Leblanc, 1922; Kristic, 2000), model-based (Guay et al., 2004) and multi-unit optimization (Srinivasan, 2007). These strategies lead to the closest local optimum depending on where the optimization starts. The framework used in this thesis is the multi-unit optimization method, where the gradient is approximated based on the finite difference between a set of parallel units which operate with input values differing by a constant, pre-fixed offset (Srinivasan, 2007).

Real time optimization (RTO) is a feed back control system that in an on-line manner, maximizes or minimizes a particular objective function. A schema presented by Marlin and Hrymak (1997) shows the place of the real-time optimization in an assumed plant operation. In a general description, process optimization is between the planning and control levels and it gets the constraints and objective function from the higher planning level and provides a set of optimized set-points for the control level. In other words, the planning step deals with the inventory management and production rate which is a function of the market needs and costs. Next, this information is transferred to the real time optimization system which handles the most profitable operating conditions. The operating conditions generally are revised after a few hours or a few days during the plant production. Finally, the desired conditions are transferred to the control system which in turn, maintains the process states to their desired set points, despite of the perturbation. It means that the task of moving the plant process towards this obtained optimality is by control system. Several elements are necessary in order to realize the real time optimization of a process. Marlin and Hrymak (1997) presented the pre-requirements of a RTO problem from a theoretical point of view. As a brief conclusion, the main role of the real-time optimization is to follow the displacement of the optimum points in the process. There are different approaches to solve an optimization problem. Certain parameters are fixed, while other parameters like primary material costs, environmental norms, utility and energy expenses, product specifications and its

price and marketing demand are dynamically variable. Therefore, an objective function which is a combined relationship of all the previously mentioned data and constraints can be determined in order to achieve the best operating conditions and set-points which lead the plant to produce with the minimum possible cost. The performance of an optimization problem depends on the quality of the reduced and simplified models which represent the process states, constraints and cost function. Real-time optimization is a recursive procedure. The first step is data acquisition from the available measured outputs (y) and to evaluate their precision. Then these data will be used in order to estimate the unknown parameters ($\hat{\theta}$) and the model coefficients. The new model which is updated by the estimated parameters will be used to optimize an objective function in order to find the new optimal set points (y^*) which will be transferred to the control system in order to implement the process. The mentioned procedure will be repeated indefinitely according to a fixed delay time by a proper algorithm. It is ideally awaited that the previous calculated set points are implemented into the system and the process attaints a new steady state before activating a new optimization cycle.

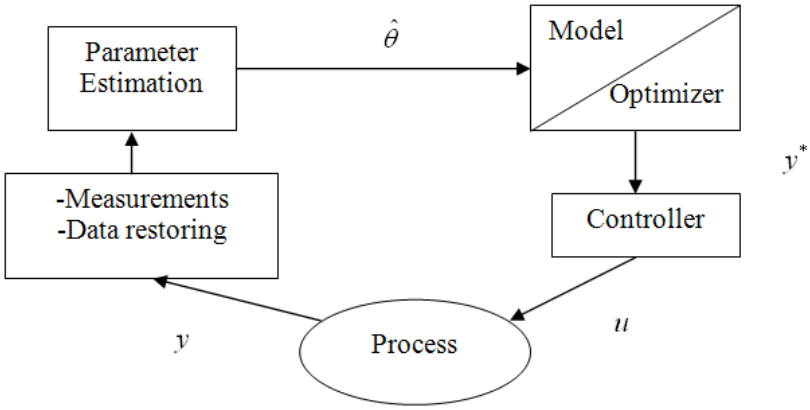

Figure 1.6 Real time optimization procedure

One of the interesting methods for implementing real-time optimization algorithms in a control loop is the extremum-seeking controllers. This is demanding when the objective function of the system should be maximized or minimized and the set-points of the system are unknown or are a complex function of different conditions. The basic concept of the extremum seeking control which is a non-model based method, is to control the gradient of the output (instead of the output)

towards a zero set-point and in this way to find the local maximum or minimum of the objective function. The objective function depends on the system to be controlled. The principal idea of the gradient-based extremum seekers is the fact that an extremum has a gradient with the magnitude of zero.

1.2.1 Local extremum seeking based on perturbations

In extremum-seeking, which is designed to take a system to one of its local maxima, the optimization problem is recast into a problem of controlling the gradient to zero. The difference between different extremum-seeking methods lies in the way the gradient is estimated. In the method using perturbations, a temporal perturbation (termed the dither) is injected along the input and the gradient is estimated by using the correlation between the input and the output. An integral controller is used to control the gradient at zero. A schematic of the standard local extremum seeking controller is provided in figure 1.7.

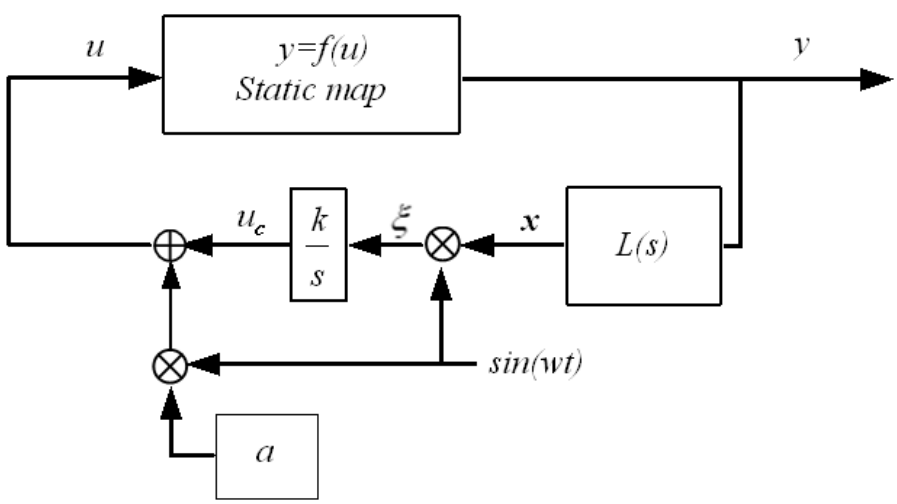

Figure 1.7 Perturbation-based local extremum seeking control (after Krstic and Wang, (2000))

The update equation is given by

$$
\begin{aligned}
&\dot{u}_c = k\xi \qquad , u_c(0) = u_{c,ini} \\
&\xi = x\sin(\omega t) \\
&X(s) = L(s)Y(s) \\
&y = f(u_c + a\sin(\omega t)).
\end{aligned}
\tag{1.12}
$$

Back to the history of the extremum seeker controllers, for the first time in 1922 Leblanc proposed a perturbation based method in order to estimate the gradient. Despite some implementation problems, the research in extremum seeking controllers was continued in the 1950s and 60s. Although it was not possible to guarantee their stability, in some practical applications they performed quite well. The 1970s and 80s were the dormant period for the research in this area. Krstic constructed a milestone in this area of research by starting to publish some articles about the stability criterion for a perturbation scheme. During the year 1999, Miroslav Krstic presented the first related article for the 38th Conference on Decision & Control concerning, a proof for stability of an extremum controller based on a perturbation scheme which was followed by several other articles related to his theories with different approaches and applications (Krstic, 1999).

The extremum seeking algorithms have to be able to control the sign and gain variation in the vicinity of the maximum or minimum. In the maximization case, if the estimated gradient is positive the search direction is kept and if it is negative the sign of the step size is switched. The size of the parameter step in each time interval can be adapted. The gain can be regarded as the step length. Once a gradient is obtained, the algorithm can determine how close the inputs of desired objective function are to the extremum. The basic idea by assuming a unique extremum was to keep the gradient at zero. This is accomplished by using an integrator (*1/S*), a gain multiplier (*k*) and a high pass filter (*L(S)*).

In order to stay at the top of the objective function (in the case of maximization with a positive gain *k*) or at the bottom (in case of minimization with a negative gain *k*), extremum seeking

algorithms determine in which direction the gradient is pointing given a state of the process. In extremum seeking controllers, the input signal is slightly distorted by some kind of periodic signal (e.g., a sinusoid). The scheme presented by Krstic in figure 1.7 estimates the gradient using a sinusoidal wave which is added before the plant. The correlation between the output and input then determines whether the operating point is located at the "right" or "left" side of the extremum (i.e., gradient estimation using correlation analysis). Suppose that we want to find the local minimum of a static and continuous map which is approximately quadratic (at least locally about the current input). In fact, there is no real requirement that the static map be quadratic and this assumption is for convenience in the following illustration. The extremum of f occurs at unknown u^* and assume that there is only one extremum in the input-output domain of interest, namely,

$$f(u) \cong f_{\min} + \lambda(u - u^*)^2 \tag{1.13}$$

Assume that the current input u_c is less than the local optimum u^*. We start perturbing this input by the following small sinusoidal wave of frequency ω:

$$u = u_c + a\sin(\omega t) \tag{1.14}$$

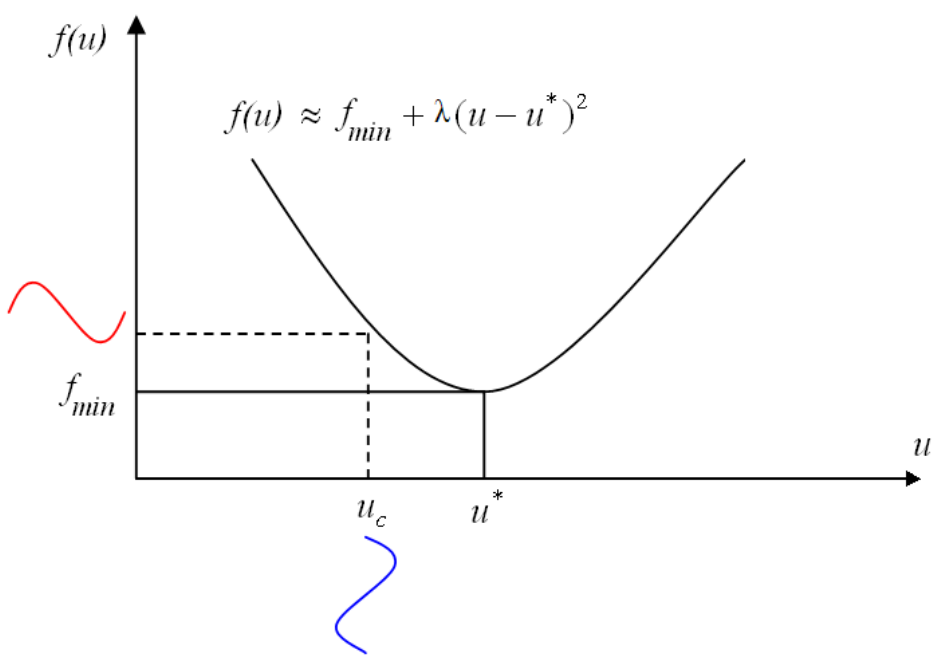

Figure 1.8 An example for static and continuous map

The sign of the output sinusoid is now opposite to the sign of the input perturbation. Then, it is desired to isolate this information and use it to update the current input u_c in order to converge to the local extremum.

$$\begin{aligned} f(u) &\approx f_{\min} + \lambda(u - u^*)^2 \\ &= f_{\min} + \lambda(a\sin\omega t - \tilde{u})^2 \quad where \quad \tilde{u} = u^* - u_c \\ &= f_{\min} + \lambda(\tilde{u}^2 - 2\tilde{u}a\sin\omega t + a^2\sin^2\omega t) \\ &= f_{\min} + \lambda(\tilde{u}^2 \underline{-2\tilde{u}a\sin\omega t} + \frac{a^2}{2} - \frac{a^2}{2}\cos 2\omega t) \end{aligned} \tag{1.15}$$

The sign of the underlined term is the important information that is desired to be isolated (since $\tilde{u}$ indicates whether the input is above or below the extremum). Then the high pass filter starts to remove DC terms i.e.,

$$\frac{X}{Y} = \frac{s}{s+h} \qquad \rightarrow \qquad x \approx \lambda(\tilde{u}^2 \underline{-2\tilde{u}a\sin\omega t} - \frac{a^2}{2}\cos 2\omega t) \tag{1.16}$$

Consequently, the sign of the second term is isolated through "demodulation",

$$\begin{aligned} \xi &= x\sin\omega t \\ &\approx \lambda\tilde{u}^2\sin\omega t \underline{-2\lambda\tilde{u}a\sin^2\omega t} - \frac{a^2\lambda}{2}\underline{\underline{(\cos 2\omega t\sin\omega t)}} \\ &\approx \lambda\tilde{u}^2\sin\omega t \underline{-\lambda\tilde{u}a + \lambda\tilde{u}a\sin 2\omega t} - \frac{a^2\lambda}{4}\underline{\underline{(\sin 3\omega t - \sin\omega t)}} \end{aligned} \tag{1.17}$$

In the above equation everything except the term ($-\lambda\tilde{u}a$) is a function of relatively high frequency. Therefore using a low pass filter integrator (*k/s*) with cut-off at 0 *Hz* after one complete period of the sinusoidal wave, we identify the desired gradient information as follows,

$$\begin{aligned} \dot{u}_c &= k\xi \\ \frac{du_c}{dt} &= -k\lambda\tilde{u}a \end{aligned} \tag{1.18}$$

Considering the definition $\tilde{u} = u^*-u_c$ and $k < 0$ for minimization case, if u_c is below u^* then du_c/dt will be positive, and if u_c is above u^* then du_c/dt will be negative. In other words, u_c always moves towards u^* and it can be shown that it converges to within a ε-neighborhood of u^*. This mathematical illustration can be similarly developed for the maximization case ($k > 0$). The scheme in figure 1.7 causes convergence to the extremum point in a static map and is analyzed by an averaging approach. Krstic developed some stability results using the averaging analysis. This method can be extended further to multiple inputs and slope seeking rather than extremum seeking. When there exist several inputs, it is possible to obtain the required information by applying the perturbation signals with different frequencies to each input. Early multivariable extremum seeking schemes developed by Rotea (2000) and Walsh (2000) followed by a systematic design procedure provided by Ariyur et al. (2003). Teel and Popović (2001) studied sufficient conditions for the asymptotic stability of the smooth and non-smooth multivariable extremum-seeking controllers.

Several articles of gradient-based extremum seeking control with adaptive design in chemical processes and reactor engineering have been published by Perrier, Dochain, Guay, Dehaan and their co-workers. Some new developments of numerical extremum seeking algorithms and their applications to Antilock Braking Systems (ABS) and swarm source seeking control were published by Zhang in 2006. Numerical extremum seeking methods have been developed in two categories (line search method and trust region method) by Zhang.

1.2.2 Global extremum seeking based on perturbations

A global extremum-seeking strategy based on perturbations (Tan et al., 2005, 2006a, b) was proposed, where the amplitude of perturbation was reduced to zero. However, it has been shown that such a technique works for a restricted class of univariate nonlinear functions. In the standard perturbation method (local optimization scheme), a, the amplitude of the dither signal is kept constant. It was shown that decreasing the amplitude of the dither can lead to the global extremum in certain cases (Tan et al., 2006 a). A schematic is provided in figure 1.9.

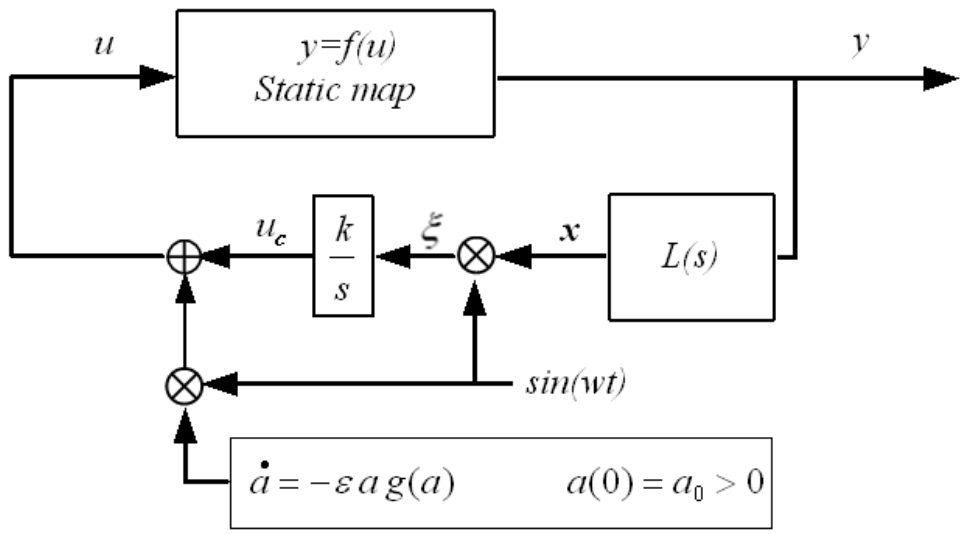

Figure 1.9 Perturbation-based global extremum seeking control

The update equation is given by,

$$\begin{aligned} &\dot{u}_c = k\xi \qquad , u_c(0) = u_{cini} \\ &\xi = x\sin(\omega t) \\ &X(s) = L(s)Y(s) \\ &y = f(u_c + \dot{a}\sin(\omega t)). \end{aligned} \tag{1.19}$$

The amplitude of the dither is constant in the local extremum-seeking controller. The difference between the local and global extremum seeking controllers is that in global optimization the amplitude of the dither is reduced in the following manner,

$$\dot{a} = -\varepsilon a g(a) \qquad a(0) = a_0 > 0 \tag{1.20}$$

where *g(.)* is a positive locally "Lipschitz" function and the parameters ε, a_0 are strictly positive. However, convergence to the global minimum requires the following conditions. For ease of the analysis consider that *L(s)*=1 (Tan et al., 2006 a), which results the following update equation,

$$\dot{u}_c = k\, f(u_c + \dot{a}\sin(\omega t))\sin(\omega t) \quad , \quad u_c(0) = u_{cini} \tag{1.21}$$

Consider the averaged function,

$$f_{av}(u,a) := \frac{\omega}{2\pi}\int_0^{\frac{2\pi}{\omega}} f(u + \dot{a}\sin(\omega t))\sin(\omega t)dt \tag{1.22}$$

Let *l(a)* be one of the isolated roots of $f_{av}(u,a) = 0$ with the following properties:

a) l is defined for all $a > 0$, $l(a)$ is continuous and $\frac{\partial f_{av}}{\partial u}(l(a)) < 0$

b) There exists $a^* > 0$, such that for all $a \geq a^*$, $l(a)$ is the unique root of $f_{av}(u,a) = 0$.

c) When $a \to 0$ then $l(0) = u^{**}$.

It was shown that if the above conditions hold, convergence of the algorithm to an arbitrarily small neighborhood of the global extremum from an arbitrarily large set of initial conditions is guaranteed. Though, these conditions are trivially satisfied in the case of fourth-order polynomials, they are difficult to satisfy in the general case and the authors present several counter-examples. Two of them will be considered below. These examples will be discussed in detail in chapter 2.

Counter-example 1: 6th-order polynomial (Tan et al., 2006 a):

$$f(u) = -u^6 + \frac{1}{10}u^5 + \frac{623}{400}u^4 - \frac{659}{4000}u^3 - \frac{11287}{20000}u^2 + \frac{259}{4000}u + \frac{637}{20000} \tag{1.23}$$

This polynomial has 3 maxima at u = -0.8985, u = 0.5, u = 0.8951, with u = -0.8985 being the global maximum. The above-mentioned algorithm converges to u = 0.8951.

Counter-example 2: Sum of exponentials (Tan et al., 2006 a):

$$f(u) = e^{\frac{1}{1+0.2u^2}} + e^{\frac{1}{1+5(u-15)^2}} \tag{1.24}$$

This counterexample has two local maxima at u = 0 and u = 15. The global maximum is at u = 15, while the algorithm converges to u = 0.

1.2.3 Local extremum seeking using multiple units

The multi-unit optimization technique is based on an alternate gradient estimation method recently proposed in Srinivasan (2007). The idea is to have two identical units operating at two different operating points differing by an offset Δ. The gradient is estimated by the finite difference and an integral controller forces the gradient to zero. The schematic is presented in figure 1.10.

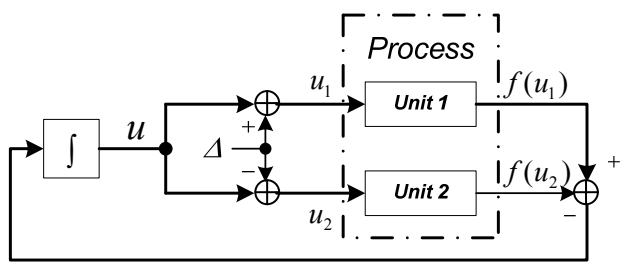

Figure 1.10 Extremum seeking control with multiple units

The update equations are given by,

$$u_1 = u + \Delta, \quad u_2 = u - \Delta \tag{1.25}$$

$$\dot{u} = k\left(\frac{f(u+\Delta) - f(u-\Delta)}{2\Delta}\right) \tag{1.26}$$

where k > 0 is the adjustable gain. The main advantage of this method is that the dynamics of the two units cancel out and the adaptation does not require time-scale separations necessary in methods based on temporal perturbations. Thereby, faster convergence to the optimum can be achieved.

1.3 Summary

In this chapter a general description of different optimization strategies including black-box optimization methods, global optimization algorithms and real-time optimization techniques has been presented. Black-box algorithms only use input and output of the system to find the optimum operation conditions. Furthermore, these methods are divided in two main categories; deterministic and stochastic. General properties of these methods have been described and pros and cons of each one have been analyzed. The genetic algorithm and simulated annealing as stochastic methods and the Lipschitzian optimization and DIRECT algorithm as deterministic methods have been analyzed in details.

One of the criteria for evaluating a global optimization algorithm is the number of functional evaluations used to ascertain global convergence. For lower dimensional systems, stochastic algorithms require prohibitively large number of functional evaluations and further more do not always guarantee global convergence. So, the goal of this work is to provide an alternate deterministic algorithm for global optimization that would require lesser number of functional evaluations than the Lipscitzian and DIRECT optimization for lower dimensional problems.

In the second part of this chapter, extremum-seeking controllers as the real-time optimization techniques have been introduced. The perturbation-based extremum seeking and multi-unit extremum seeking control are two main categories presented in this class. These methods recast the optimization problem into a control problem. Moreover, these strategies are very appropriate to perform black-box optimization. A brief introduction to the local and global extremum seeking based on perturbation has been presented. The perturbation-based global extremum seeking is applicable only to scalar systems and furthermore to only a restricted class of polynomials. So, the goal of this thesis is to check whether one could extend the class of systems for which the global optimum is found by using multi-unit extremum seeking instead of perturbation based extremum seeking. Also, we wish to extend the methodology to global optimization of two- and three-input systems.

CHAPTER 2 GLOBAL OPTIMIZATION OF SCALAR SYSTEMS USING MULTI-UNIT EXTREMUM SEEKING

The goal of chapter 2 is to develop a deterministic global optimization method that can handle the black-box univariate objective functions. It is shown that in extremum seeking control with multi-units when the offset between inputs is reduced to zero, the scheme reaches the global optimum for all continuous nonlinear scalar functions. This technique is also extended to include inequality constraints.

2.1 Unconstrained global optimization using multi-units

Consider the problem of maximizing, $y = f(u)$, where $f: R \rightarrow R$, is scalar, static, non-convex continuous, nonlinear function. The problem may have multiple local maxima, u_k^*, $k = 1,2, \ldots, n$, but a unique global maximum, u^{**}. In the rest of the chapter, it is assumed that the global maximum is unique.

2.1.1 Schematic diagram

In this thesis, an idea similar to that proposed by Tan et al. (2006 a) is used in the multi-unit framework to achieve global optimization. The offset parameter is reduced to zero in a predefined fashion. But, the core contribution lies in the fact that the length of step taken by the input is determined not by the gradient but by the variation of the offset parameter. The adaptation laws are given by,

$$\dot{u} = g(\Delta).sign(f(u+\Delta) - f(u-\Delta)) \qquad u(0) = u_{ini} \tag{2.1}$$

$$\dot{\Delta} = -g(\Delta) \qquad \Delta(0) = \Delta_{ini} > 0 \tag{2.2}$$

g(.) can be any positive bounded function which is zero only when *Δ = 0*. Depending on *g(.)*, the offset *Δ* decreases to zero linearly, exponentially or in any other manner. For linear and exponential decrease, *g(Δ)* would be as represented follows,

$$g_{lin}(\Delta) = \begin{cases} k & if \quad \Delta > 0 \\ 0 & if \quad \Delta = 0 \end{cases} \tag{2.3}$$

$$g_{\exp}(\Delta) = k\Delta \tag{2.4}$$

k > 0 is a parameter that determines the rate at which *Δ* is reduced to zero. For example, if the exponential convergence for dynamics of *Δ* is chosen, the dynamics of *Δ* would be calculated by $\Delta = \Delta_{ini} e^{-kt}$. The time constant of this system is $\frac{1}{k}$. The larger the adaptation gain *k*, the smaller will be the time constant of the system. The time is artificial in this method and any value can be chosen as long as the integration time, *T*, is chosen accordingly. However, the precision of the algorithm is determined by "*kT*" i.e., as long as the value of "*kT*" remains unchanged, the precision of the algorithm won't be affected. Since the value of *k* does not affect the performance of integration algorithm, *k = 1* is chosen in the rest of the section with a fixed execution (simulation) time. This way, the precision of the algorithm is fixed for all examples.

The modified schematic is presented in figure 2.1.

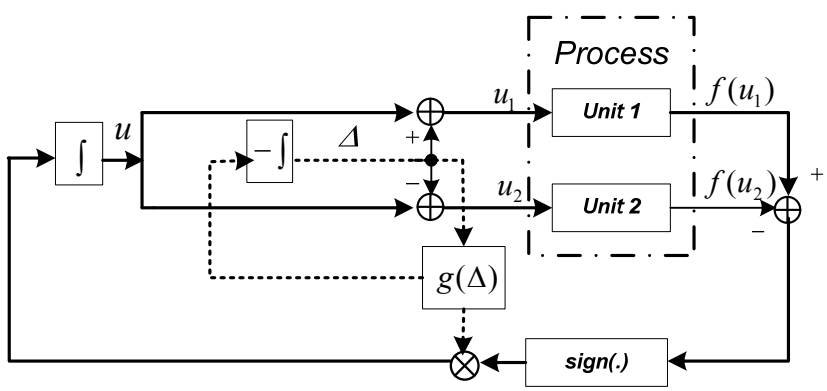

Figure 2.1 Global extremum-seeking control with multiple units

2.1.2 Convergence

With these modifications, it will be shown below that no preconditions on the nonlinear function are required to achieve the global maximum. However, the initial condition and the initial value of the offset parameter should be chosen such that the global maximum lies in the initial interval $[u_1(0), u_2(0)]$. Using the equations (1.25) this interval is equivalent to $[u(0)-\Delta(0), u(0)+\Delta(0)]$ or $[u_{ini}-\Delta_{ini}, u_{ini}+\Delta_{ini}]$. It will be shown that this algorithm is capable of avoiding being stuck in any of the local optima and always converges to the global one.

Theorem 2.1.1 *Consider the multi-unit optimization scheme with the adaptation laws (2.1)-(2.2) where the offset parameter Δ is monotonically decreased to zero. If (a) f(.) has a unique global maximum and (b) $|u_{ini}-u^{**}|\leq\Delta_{ini}$, then* $\lim_{t\to\infty} u(t) = u^{**}$.

Proof First, it will be shown that $|u(t)-u^{**}| \leq \Delta(t), \forall t$. This fact will be proved by contradiction. Suppose at time instant t, $|u(t)-u^{**}| > \Delta(t)$. Also, from the hypothesis, $|u_{ini}-u^{**}| \leq \Delta_{ini}$, i.e., $|u(0)-u^{**}| \leq \Delta(0)$. So, there should exist a time instant $\tau < t$, such that $|u(\tau)-u^{**}| = \Delta(\tau)$. This means that either $u_1(\tau) = u(\tau)+\Delta(\tau) = u^{**}$ or $u_2(\tau)= u(\tau)-\Delta(\tau) = u^{**}$. Without loss of generality assume that $u_1(\tau) = u^{**}$.

From equations (2.1) and (2.2) it is clear that $|\dot{u}| = |\dot{\Delta}|$. With $u_1(\tau) = u^{**}$, due to the uniqueness of the global maximum, $f(u_1) > f(u_2)$, and from the same equations it can be concluded $\dot{u} = -\dot{\Delta}$. On the other hand, $\dot{u}_1 = \dot{u} + \dot{\Delta}$, so, $\dot{u}_1 = 0$.

So, $u_1(t) = u^{**}$, $\forall t > \tau$. In other words, $u_1(t) = u(t)+ \Delta(t) = u^{**}$, i.e., $|u(t)-u^{**}| = \Delta(t), \forall t > \tau$. This is in contradiction with the supposition $|u(t)-u^{**}| > \Delta(t)$. This shows that $|u(t)-u^{**}| \leq \Delta(t) \ \forall t$.

Since $\dot{\Delta} = -g(\Delta)$ is bounded, $\Delta(t)$ is a continuous function of t. Also, since $\Delta(0) = \Delta_{ini} > 0$, if $\Delta(t)$ has to become negative at some point of time, it cannot jump but pass through $\Delta = 0$. However when $\Delta = 0$ at $t = \rho$, since $g(0) = 0$, $\dot{\Delta}$ will be 0, which implies $\Delta = 0$ from that time onwards, i.e., $t \geq \rho$. So $\Delta(t)$ can never change sign and, $\Delta(t) \geq 0$ for all t. Also $g(\Delta) > 0$ means that Δ decreases in a strictly monotonic fashion for $\Delta > 0$ and is bounded from below by zero. This concludes that $\lim_{t\to\infty} \Delta(t) = 0$.

Since $|u(t) - u^{**}| \leq \Delta(t)$ $\forall t$, it can be said that asymptotically $\left|\lim_{t\to\infty} u(t) - u^{**}\right| \leq \lim_{t\to\infty} \Delta(t) = 0$, i.e., $\lim_{t\to\infty} u(t) = u^{**}$. ■

Remark 1 The sign function in the adaptation law (2.1) corresponds to a very high gain and induces a non-Lipschitz nature in the system. This might cause stiffness in integration. A simple solution is to replace the sign function by the hyperbolic tangent as shown below.

$$\dot{u} = g(\Delta).\tanh\left(\eta\left(\frac{f(u+\Delta) - f(u-\Delta)}{2\Delta}\right)\right) \tag{2.5}$$

where η is a tuning parameter. The lower the value of η, the faster will be the integration. However, a low value of η might lead to a situation where the global optimum is missed and the algorithm converges to a local maximum. Thus, the value of η should be chosen as compromise between accuracy and integration time.

Remark 2 The only condition for this algorithm to converge to the global optimum is $|u_{ini} - u^{**}| \leq \Delta_{ini}$. Since the location of u^{**} is not known a priori, the above condition will be satisfied by choosing a large enough initial value for Δ_{ini}. The downside of such a choice is that the algorithm requires more time to get to the optimum.

Remark 3 The final convergent point of the algorithm depends on the choice of Δ_{ini}. If Δ_{ini} is chosen large enough such that the real global optimum of the objective function sits in the interval $[u_{ini} - \Delta_{ini}, u_{ini} + \Delta_{ini}]$, the algorithm will converge to the real global optimum of the system. However, if Δ_{ini} is not chosen large enough, the algorithm still converges to the best optimum corresponding to the interval $[u_{ini} - \Delta_{ini}, u_{ini} + \Delta_{ini}]$. As a result, this algorithm always guarantees convergence to the best optimum which resides in the initial interval $[u_1(0), u_2(0)]$.

Remark 4 The performance of the "Sign" function in the adaptation law of the global extremum seeking method makes the structure of this control law similar to sliding mode control (Khalil, 2002). Sliding mode control is a nonlinear control method that switches from one continuous control structure to another by means of a high-frequency switching strategy. The similarities of these methods deserve more research. One similarity would be the robustness of these controllers to parameter variations (that may enter into the control channel) because of their particular On/Off switching structure.

2.2 Constrained global optimization using multi-units

2.2.1 Schematic diagram

Consider the problem of maximizing the constrained function $y = f(u)$, where $f: R \rightarrow R$, is scalar, static, non-convex, nonlinear and bounded function. This problem can be expressed as follows:

$$\begin{aligned} &\underset{u \in R}{Max} \quad y = f(u) \\ &s.t. \quad C_i(u) \leq 0 \qquad i = 1 \cdots m \end{aligned} \tag{2.6}$$

The problem may have multiple local maxima, u_k^*, $k = 1,2,\ldots,n$, but a unique global maximum, u^{**}. It is also assumed that the constrained global maximum is unique. The input constraint sets $C_i(u) \leq 0$ $(i=1,\ldots,m)$ define the feasibility region for input u. These constraints represent the critical or physical bounds of the input. The main question that is addressed in this section is how these constraints, which could lead to a non-convex feasible set, can be incorporated in the algorithm. The proposed algorithm uses the spirit of the unconstrained global optimization one. It is assumed that the feasible global maximum lies within the initial interval $[u_1(0), u_2(0)]$. If inputs of both the units are feasible, then the same adaptation law as in (2.1) is used. However, when one of the inputs is infeasible, then the feasible input is kept unchanged, while only the infeasible input is moved as the offset is reduced to zero. If both the inputs are infeasible, both of them are moved towards each other in search of a feasible point. The inputs will eventually become

feasible, since the feasible global maximum is assumed to be within that interval. This switching logic is expressed through the following adaptation laws,

$$\dot{u} = g(\Delta).S \qquad u(0) = u_{ini}$$

$$S = \begin{cases} sign(f(u+\Delta) - f(u-\Delta)) & \text{if } u_1 \text{ and } u_2 \text{ are feasible} \\ 1 & \text{if } u_1 \text{ is feasible and } u_2 \text{ is infeasible} \\ -1 & \text{if } u_1 \text{ is infeasible and } u_2 \text{ is feasible} \\ 0 & \text{if } u_1 \text{ and } u_2 \text{ are infeasible} \end{cases} \tag{2.7}$$

$$\dot{\Delta} = -g(\Delta) \qquad \Delta(0) = \Delta_{ini} > 0 \tag{2.8}$$

where *g(.)* is a positive bounded function as (2.3) and (2.4). The schematic of this switching control strategy is presented in figure 2.2. As can be seen, the only difference from figure 2.1 is that the sign function is replaced by a switching logic decided by the constraints of the problem.

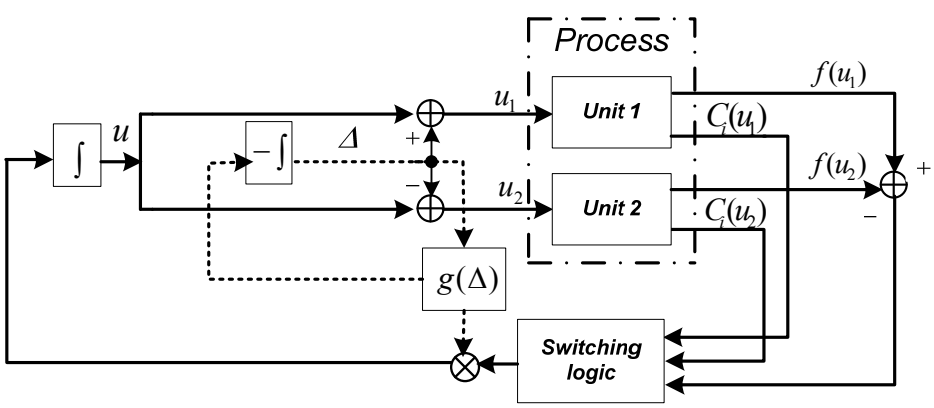

Figure 2.2 Constrained global extremum seeking control with multiple units

2.2.2 Convergence

With these modifications, it will be shown below that without any preconditions on the nonlinear function and the constraints, the algorithm always converges to the feasible constrained global maximum. However, the initial condition and the initial value of the offset parameter should be chosen such that the global maximum lies in the initial interval $[u_1(0), u_2(0)]$.

Theorem 2.2.1 *Consider the multi-unit constrained optimization problem (2.6) with the adaptation laws (2.7)-(2.8). If (a) f(.) has a unique constrained global maximum and (b)* $|u_{ini} - u^{**}| \leq \Delta_{ini}$, *then* $\lim_{t \to \infty} u(t) = u^{**}$.

Proof Note that the proof of reaching the unconstrained global optimum is based on the fact that the global maximum always lies within the interval $[u_1(t), u_2(t)]$ for all t (Theorem 2.1.1). The same idea will be used here. It is shown next that global optimum cannot be removed from the interval $[u_1(t), u_2(t)]$ in any of the cases by the adaptation law.

- If both inputs are feasible, the arguments from unconstrained global optimization can be evoked to show that feasible global optimum still lies within the interval $[u_1(t), u_2(t)]$.
- If one of the inputs is infeasible, the feasible input is left unchanged and only the infeasible one moves. So, as the interval shrinks, the part that is removed by this adaptation is only an infeasible sub-interval. The global optimum being feasible, the adaptation proposed in this case cannot remove it from the interval.
- If both inputs are infeasible, both inputs are moved towards each other till one of them is feasible. This way only the infeasible sub-intervals are removed by the adaptation. On similar lines, the feasible global optimum cannot be removed from the interval $[u_1(t), u_2(t)]$.

As the interval shrinks to zero, since the feasible constrained global optimum is trapped in the interval, both units reach the global optimum. ■

2.3 Illustrative examples

2.3.1 Application of global multi-unit optimization method

Example 2.3.1 Consider the following nonlinear static map (shown in figure 2.3) with a unique global maximum at $u^{**} = 1.68$ and several local optima at other points.

$$f(u) = -3u^4 + 64\sin^2(u^3) + 12u^2 + 4u - 80 \tag{2.9}$$

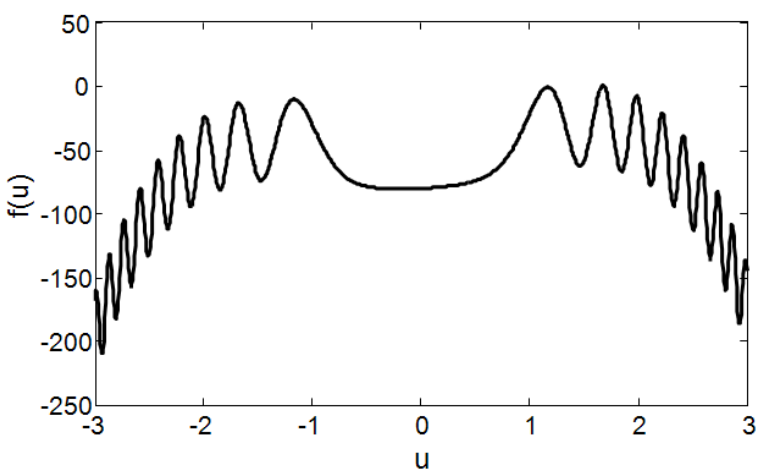

Figure 2.3 Static nonlinear map for example 2.3.1

The global optimization algorithm using two identical units is applied to optimize this nonlinear system. The initial input u_{ini} = -1 and Δ_{ini} = 4 were considered such that the global maximum along with several other local ones lie in the interval. If the exponential decreasing is considered for Δ by choosing $g_{exp}(\Delta)$ as equation (2.4) (with k =1), the time evolution of the inputs and Δ would be as figure 2.4.

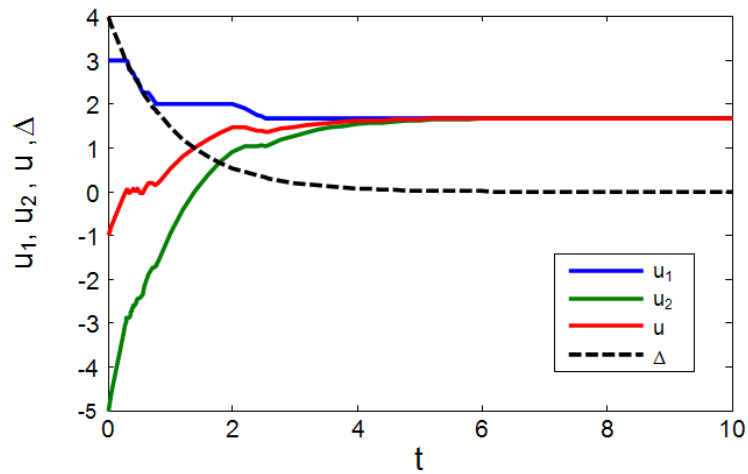

Figure 2.4 Evolution of u_1, u_2, u and exponential Δ for example 2.3.1

It is equally interesting to see the evolution of the inputs to the two units, u_1 and u_2. Most of the time only one of them evolves and the other (which has a higher objective function value) is kept constant. Also, there are time regions where both units have the same objective function and the inputs evolve together.

If the linear adaptation is considered for Δ by choosing $g_{lin}(\Delta)$ as equation (2.3) (with k=1), the time evolution of the inputs and Δ would be as figure 2.5.

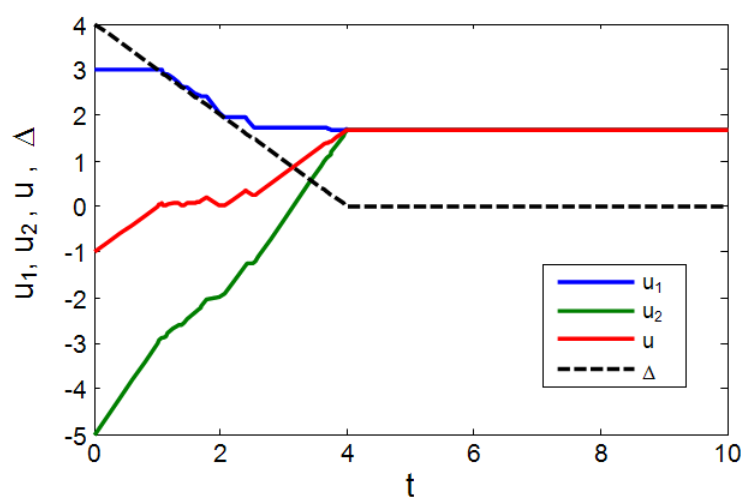

Figure 2.5 Evolution of u_1, $u2$, u and linear Δ for example 2.3.1

The exponential and linear adaptation of Δ only affects the number of function evaluations needed for convergence, which is 82 in the exponential case and 91 in linear case. Otherwise, both schemes converge to the global optimum. Using the adaptation as in (2.1) instead of (2.5) leads to a longer execution time of 13.2 sec in contrast to 1.31 sec in the later case with $\eta = 1$. The increase in execution time can be attributed to the increase in stiffness. To illustrate this idea, figure 2.6 shows the increase of integration time with increasing η. Of course, if too small value of η is chosen ($\eta < 0.2$), then the global optimum will be missed.

Note that the integration time is the time horizon over which the algorithm is integrated. Execution time (simulation time) can be calculated by "tic and toc" command in Matlab. The number of function evaluations is the number of times that the objective function is called.

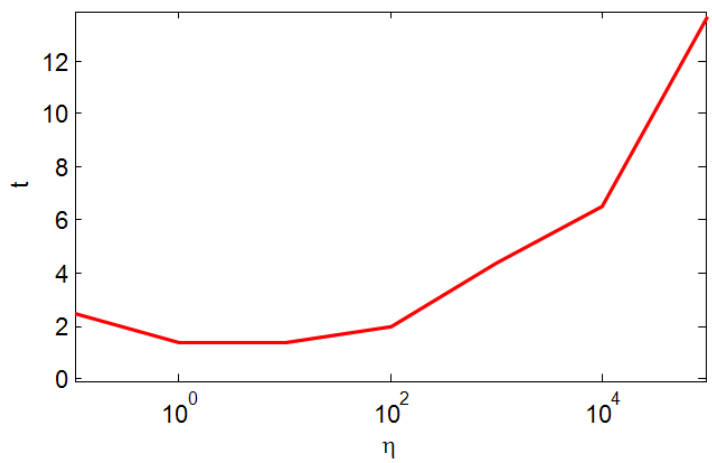

Figure 2.6 Influence of η on integration time

Example 2.3.2 Consider counter-example 1 presented in Section 1.2.2 where the global extremum-seeking controller based on the sinusoidal perturbation method was unable to find the global optimum. The static map is depicted in figure 2.7.

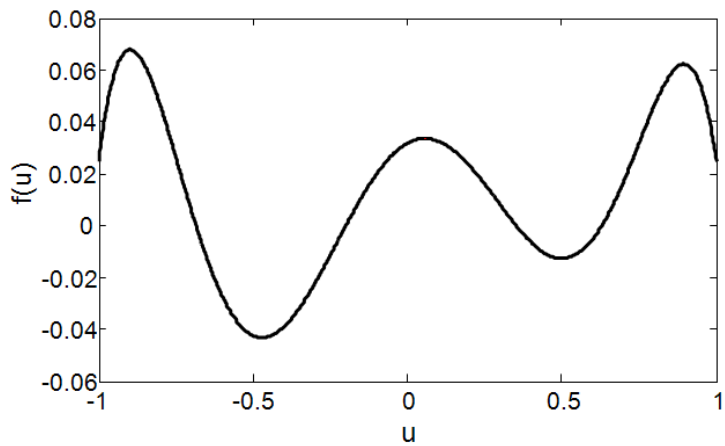

Figure 2.7 Static nonlinear map for example 2.3.2

Applying the global optimization algorithm using multiple units makes the system input converge to the global maximum. $g_{exp}(\Delta) = \Delta$ has been chosen in this example. The key condition to satisfy is the inequality $|u_{ini} - u^{**}| \leq \Delta_{ini}$ which is in fact verified by choosing $u_{ini} = 2$ and $\Delta_{ini} = 3$. The initial condition of u is chosen on purpose so as to be as closer to the local maxima and further away from the global one. The algorithm converges to the global maximum at $u^{**} = -0.8985$. Choosing $u_{ini} = 2$ and $\Delta_{ini} = 2$ causes the algorithm to converge to the local maximum $u = 0.8951$ instead of the global one. This is because the initial interval $[u_1(0), u_2(0)] = [0, 4]$ does not include the global maximum. In fact, the solution corresponds to the global maximum of this initial interval.

Example 2.3.3 Consider the second counter-example of section 1.2.2 where the global perturbation-based extremum-seeking controller was unable to find the global maximum. The static map is given in Figure 2.8. The proposed algorithm successfully converges to the global maximum with the following parameters: $u_{ini} = -10$, $\Delta_{ini} = 60$ and $g_{exp}(\Delta) = \Delta$.

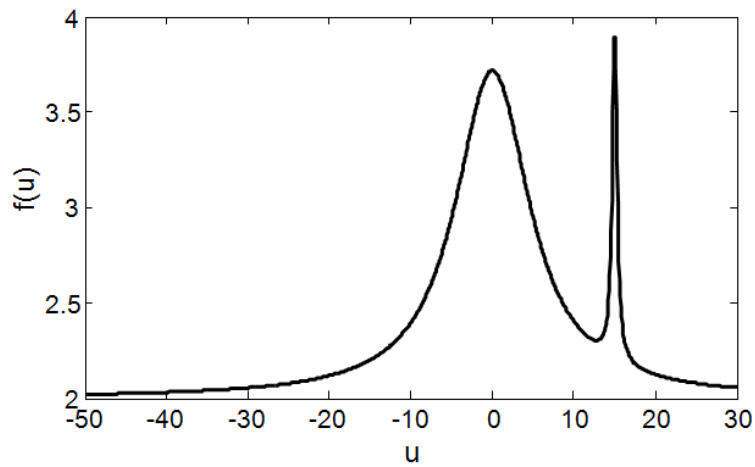

Figure 2.8 Static nonlinear map of example 2.3.3

In this example, it can be seen from figure 2.9 that the maximum of $f(u_1)$ and $f(u_2)$ (which in this case corresponds to $f(u_1))$ is always non-decreasing. Also, it can be seen that $f(u_1)$ remains constant while the other tries to catch up with it. Also, there are phases where both improve together ($t < 2$ and $t > 6$). Although the maximum (of $f(u_1)$ and $f(u_2)$) being non-decreasing is a good property but it is not always satisfied by all problems (example 2.3.4).

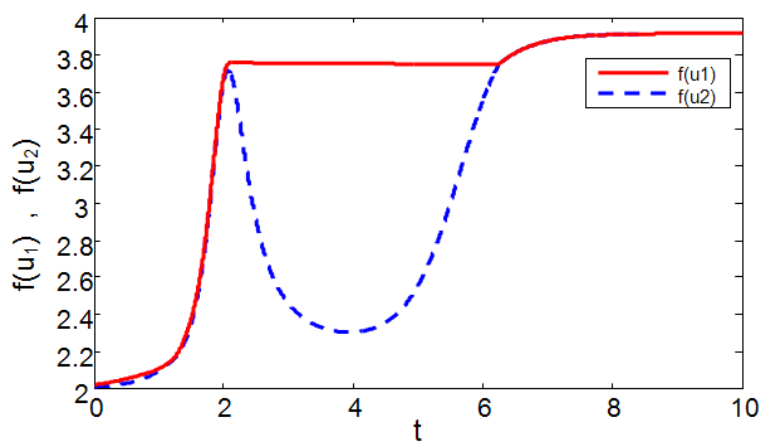

Figure 2.9 Evolution of $f(u_1)$ and $f(u_2)$ in example 2.3.3

Example 2.3.4 This example considers a nonlinear system which has several local optima, but with the global maximum that occurs at $u^{**} = 0$. Two important aspects of this example are the

discontinuous derivative at the global optimum and the presence of equal valued and symmetric local optima.

$$f(u) = -\left|u + 3\sin(u)\right| \tag{2.10}$$

The optimization and control parameters were: u_{ini} = -10, Δ_{ini} = 15 and $g_{exp}(\Delta) = \Delta$. The algorithm converges to the global optimum despite discontinuous derivatives.

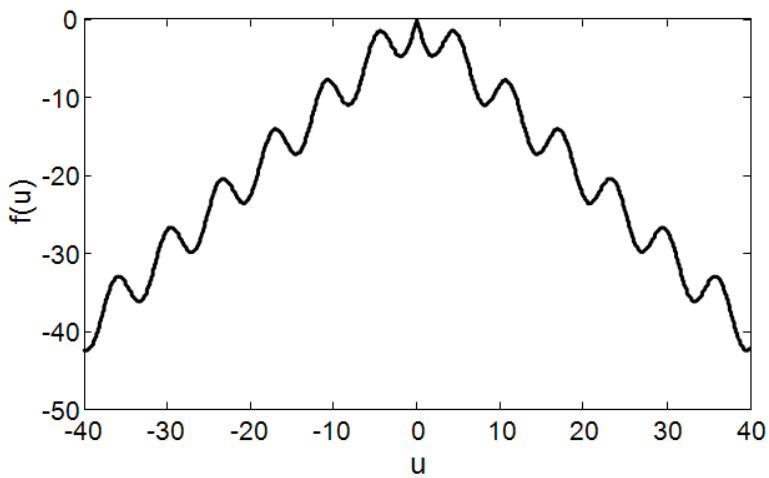

Figure 2.10 Static nonlinear map of example 4

Due to the symmetric nature of the problem, the maximum of the objective function of the two units follows the ups and downs of the given static map. In this case, it can be seen that maximum of $f(u_1)$ and $f(u_2)$ does not monotonically increase as shown in figure 2.11.

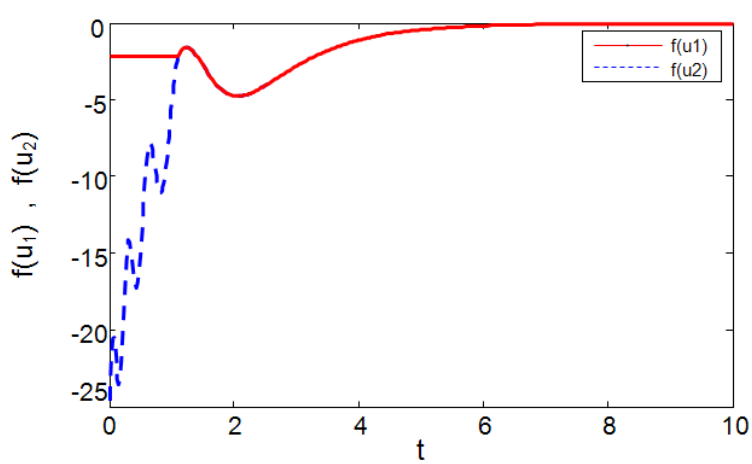

Figure 2.11 Evolution of $f(u_1)$ and $f(u_2)$ in example 2.3.4

Example 2.3.5 Consider the following constrained nonlinear optimization problem.

$$\begin{aligned} &\underset{u \in R}{Max} \quad f(u) = -3u^4 + 64\sin^2(u^3) + 12u^2 + 4u - 80 \\ &s.t. \quad (u+1)(u-2) \geq 0 \end{aligned} \tag{2.11}$$

The global constrained optimization algorithm using two identical units is applied to optimize this nonlinear system. The initial parameters used were u_{ini} = -2, Δ_{ini} = 5 and $g_{exp}(\Delta) = \Delta$ has been chosen for adaptation laws. The feasible region for the unit inputs is highlighted and the infeasible region has been shown by dashed lines on the nonlinear map (figure 2.12). The static map *f(u)* has a unique feasible global maximum at u^{**} = -1.166 and several local optima at other points.

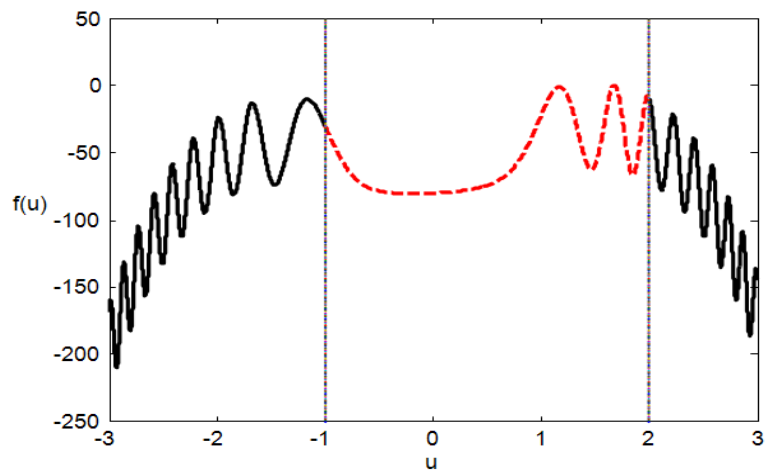

Figure 2.12 Static nonlinear map for example 2.3.5

The time evolution of the units' inputs and Δ are shown in figure 2.13.

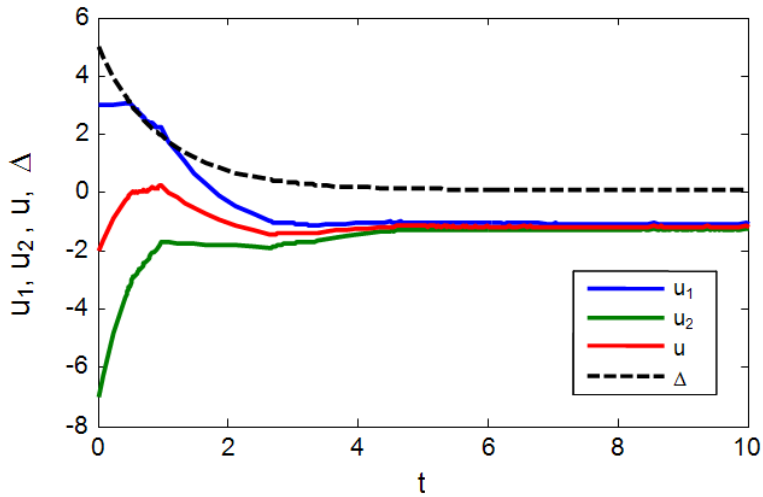

Figure 2.13 Evolution of u_1, u_2, u and Δ for example 2.3.5

As it is seen from Figure 2.13 the inputs of both units converge to u^{**}= -1.166 which is the feasible global maximum of the objective function. The value of the offset is reduced to zero. It can be seen from this figure that the input trajectory of the unit 1 violates the constraints when it takes the value between -1 and 2. However, infeasibility is temporary and the operating points of the units are never stuck in the infeasible region.

Example 2.3.6 The last example considers the constrained optimization of the following nonlinear system:

$$\begin{aligned} &\underset{u \in R}{Max} \quad f(u) = -2u - u\sin(u) \\ &s.t. \quad (u-20)^2 + (-2u - u\sin(u) + 25)^2 \le 144 \end{aligned} \tag{2.12}$$

As it is seen from figure 2.14, the feasible set consists of several discontinuous intervals (the objective function in the feasible intervals is represented by a solid line, the dotted line is used to represent the objective function outside the feasible region). The method proposed in this chapter is applied with initial values u_{ini} = 20, Δ_{ini} = 12 and $g_{exp}(\Delta)$ = Δ has been considered for adaptation laws.

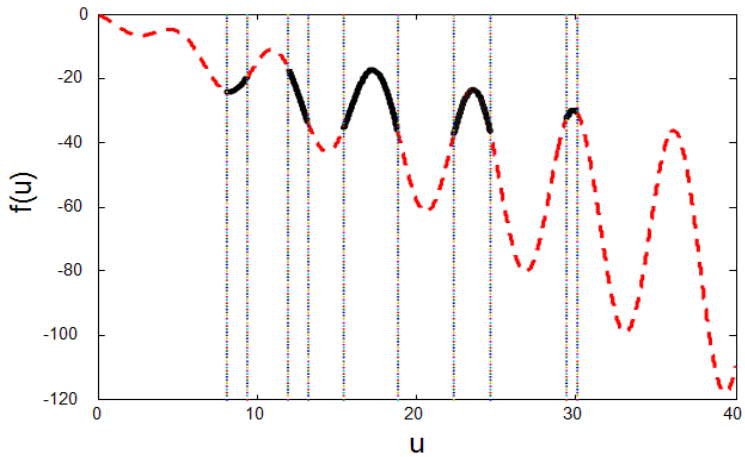

Figure 2.14 Nonlinear map and constraints for example 2.3.6

The initial conditions of u_{ini} and Δ_{ini} are chosen such that the initial interval covers the entire feasible region. The algorithm converges to the feasible global maximum u^{**}= 11.92, as seen in figure 2.15. Though the feasible region is non-convex in examples 5 and 6, the key difference is that the solution is at the boundary here. This means that the gradient is not equal to zero at the optimum. The non-zero gradient then tries to seek an operating point with a higher cost by pushing the system into the infeasible region. However, the feasibility part of the algorithm pushes the system back into the feasible region. Thus, the solution chatters around the optimal point as can be seen in figure 2.16. The frequency of chattering depends on the final value of Δ which is determined by ε. Gradient projection methods (Woodward et al., 2007) can be used to eliminate this chattering. The very small value ε is chosen to avoid numerical problems. This way Δ asymptotically goes to ε. In other words, ε is a parameter that determines the arbitrary neighborhood of the global optimum at the end of convergence.

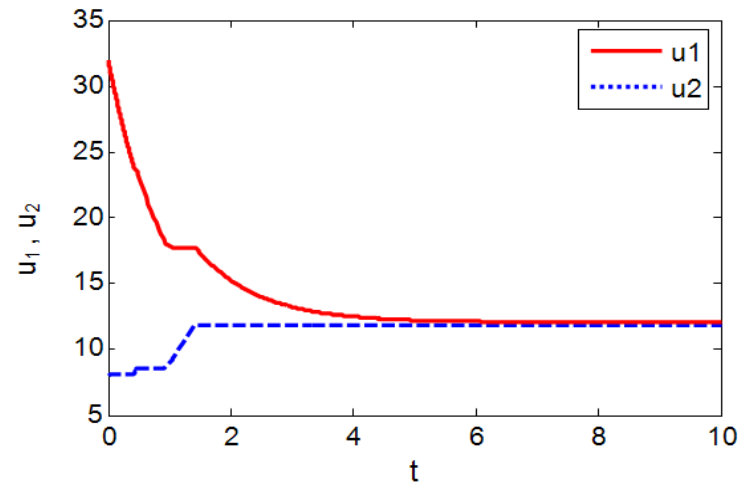

Figure 2.15 Evolution of u_1 and u_2 for example 2.3.6

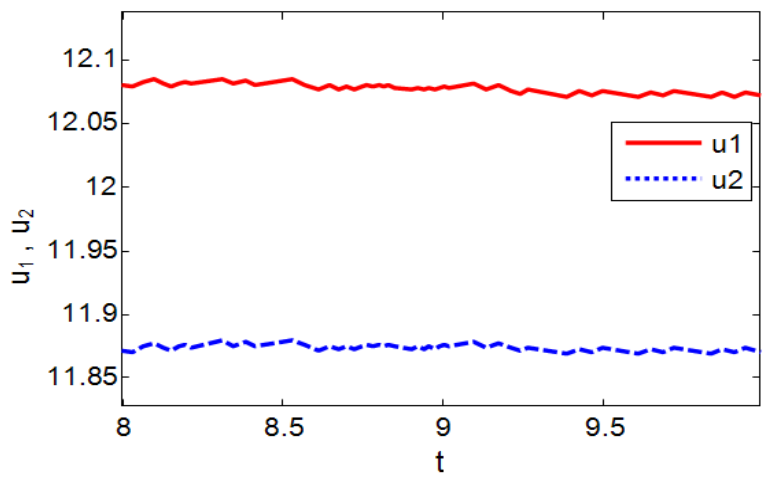

Figure 2.16 Zoom of figure 2.15 after convergence

2.4 Comparison with other global optimization methods

In order to compare the performance of the global optimization by Multi-Units (MU) with other global optimization methods; DIRECT algorithm, Genetic Algorithm (GA) and Simulated Annealing algorithm (SA) were considered to optimize the nonlinear map of Example 2.3.1.

It is important to note that the multi-unit optimization can be visualized as multi-model optimization where two mathematical models are used in the place of units. In fact, MU uses two individual units (models) to perform its algorithm whereas the other methods, except GA, use only one unit in their procedure. In GA, as many units (models), as there are members in the population, are used.

Global multi-unit optimization generates a single point at each iteration (function evaluation) and selects the next point by a deterministic computation. The sequence of points approaches an optimal solution. DIRECT converges to global optimum by systematic gridding the search area. On the other hand, the Genetic Algorithm generates a population of points at each iteration and selects the next population by a computation which uses random number generators. The best point in the population approaches an optimal solution. Similarly, the simulated annealing algorithm begins by randomly generating a new point and the next point of each iteration is determined by a probabilistic computation. The optimal solution is gradually approached by systematically decreasing a control parameter (temperature) which determines the probability of accepting a worse solution at any step (Schneider & Kirkpatrick, 2006).

The computational results of these algorithms were obtained by using the Optimization Toolbox 4.0 of MATLAB version 7.6.0.347(R2009a) and SIMULINK version 6(R14) which are registered trademark of the MathWorks. For deterministic algorithms two sets of tests and for probabilistic algorithms three sets of tests with different tuning parameters were performed. For probabilistic algorithms, each set consists of 100 individual implementations of the considered algorithm. In order to show a comparative result, one of the key factors of each algorithm ($g(\Delta)$ in MU, the iteration number in DIRECT ($N_{iteration}$), the population size (N_{pop}) in GA and the initial temperature ($T_{initial}$) in simulated annealing) were changed during each set of these tests. The other initial parameters for global multi-unit optimization were u_{ini} = -1, Δ_{ini} = 3. The tuning factors for Genetic algorithm such as (fitness scaling function, crossover function, mutation function, stopping criteria, …) and for simulated annealing algorithm such as (annealing parameters, temperature update function, stopping criteria, …) were set as their default values (Appendix IV). Since the searching area for MU was between $(u_{ini} - \Delta_{ini})$= - 4 and $(u_{ini} + \Delta_{ini})$= 2, these bounds were considered for scalar optimization by DIRECT, GA and SA. This way, all algorithms could operate within the same bounded area [-4, 2].

The results were compared in terms of the percentage of successful convergence to the global optimum and the average number of function evaluations (table 2.1). The percentage of the successful convergence to the global optimum after 100 individual implementation of an algorithm is referred to as the reliability of the global optimization method.

Table 2.1 Comparison between global multi-unit optimization, DIRECT, genetic algorithm and simulated annealing

Global optimization method		Successful global convergence %	Average number of function evaluations
Deterministic	**MU1** g_{exp}	100	82
	MU2 g_{lin}	100	91
	DIR1 $N_{iteration}=50$	100	593
	DIR2 $N_{iteration}=20$	100	153
Probabilistic	**GA1** $N_{pop}=200$	100	10400
	GA2 $N_{pop}=50$	60	2556
	GA3 $N_{pop}=20$	38	1012
	SA1 $T_{initial}=200$	100	825
	SA2 $T_{initial}=20$	97	781
	SA3 $T_{initial}=2$	84	865

As it is shown in table 2.1, the global optimization method with multiple units always converges to the global optimum using a strictly positive decreasing function in the adaptation laws. The DIRECT algorithm also converges to the global optimum after a certain number of function evaluations. On the other hand, these results show a significant percentage of convergence to a false optimum when the population size of Genetic algorithm and the value of initial temperature for Simulated annealing algorithm are chosen very small. Increasing the population size of GA

and the value of initial temperature in SA, enhance the percentage of the global convergence. As a result, both of the algorithms converge to the global optimum when the mentioned parameters are tuned sufficiently large.

The precision of the converged global optimum was 0.001 for all of the algorithms. Also, in optimization by multiple units, this precision is controllable by choosing ε (the final value of Δ). While the optimization by multi-units converges to the same optimum during all tests of each set, the other algorithms show a probabilistic nature in their final convergence. The rate of successful convergence for MU is always 100% if Δ_{ini} is chosen properly (big enough). The global convergence of DIRECT algorithm with an acceptable precision is also guaranteed after limited number of function evaluations. However for GA and SA, the rate of successful convergence is directly proportional to the population size and the initial temperature value respectively. It is also important to note that the number of function evaluations via global optimization by multi-units and DIRECT algorithm were constant in each individual set of these tests. This fact is because of the deterministic nature of these algorithms. However, the number of function evaluations is also dependent on the integrator type (*ode15s, ode23, ode45,…*) which is applied in *MATLAB* or *SIMULINK* (*ode15s* was applied in this study). It also depends on the tolerances used in the integration. In short the accuracy of the method depends very much on the accuracy of the integrator. It is also seen that the number of function evaluations for MU depends on the decreasing manner of Δ.

In Genetic Algorithm when the population size is made large, the number of function evaluation significantly increases. However, there is no correlation between the number of function evaluations and the value of initial temperature in simulated annealing algorithm.

As is clear from table 2.1, the average numbers of function evaluations for global optimization by multi-units were less than the one for Genetic Algorithm and simulated annealing in all of the tests. Also, MU usually takes less number of function evaluations to converge in univariate case comparing to DIRECT algorithm. This shows that for this case, a significant improvement in terms of number of function evaluations performed by global optimization via multiple units comparing to the other global optimization algorithms.

If the optimization algorithm is directly linked with real units, the global optimization using multi-units would be advantageous to other methods (DIRECT, GA and SA), particularly due to

the continuity of system inputs in multi-unit optimization. The inputs of the multi-unit system continuously and smoothly converge to the global optimum whereas in other methods the inputs are essentially discontinuous.

Table 2.2 Comparison between global multi-unit optimization with different η

Global optimization method		Successful global convergence %	Average number of function evaluations
Deterministic	**MU1** *η=200*	100	195
	MU2 *η=20*	100	170
	MU3 *η=0.2*	0	343

As it is shown in table 2.2, the global optimization method by multiple units always converge to the global optimum when the value of parameter η is selected sufficiently large (η>0.2). Similarly, this method never converges to the global optimum if η is selected too small. It is also seen that the number of function evaluations for MU (when η>1) is proportional to the value of tuning parameter η. This means that for (η > 1), the lower the value of η, the lower will be the number of function evaluations. This was similarly mentioned in (remark 1) and (figure 2.6). A very low value of η (η < 0.2) leads to a situation where the global optimum is missed.

2.5 Conclusion

Chapter 2 describes the global optimization of a scalar noise-free nonlinear function. The approach is cast in the framework of extremum-seeking control. However, instead of inferring the gradient through the addition of a dither signal and the computation of the correlation between the input and the output, it proposes to position two similar units in parallel and feed them

slightly different inputs, thereby allowing the computation of the gradient. In this framework, a deterministic global optimization method using multi-units for nonlinear static scalar maps is proposed. The presented method is a model-free gradient-based algorithm which uses the measurement data of the objective function for estimation of the gradient. This technique overcomes the classical shortcoming of the real-time local extremum seekers which drop in the pitfall of the local optima depending on the initial conditions.

The approach is first presented for unconstrained optimization, followed by an extension to constrained problems, for which a switching logic is introduced. Several simple (scalar) academic examples are presented that illustrate the approach. The working idea consists in reducing the amplitude of the dither signal asymptotically to zero, which is quite appealing theoretically. In this technique, the excitation is the difference between the inputs sent to the two plants.

The presented method uses the multi-unit optimization structure, where some predefined offsets are introduced between the inputs of identical units and the gradient is estimated by finite difference. The method utilizes some interesting properties of finite differences unexplored previously. It was shown that if the offsets are reduced to zero in a controlled manner, the whole system can be made to converge to the global optimum for nonlinear continuous static maps. The global optimization was achieved by initially starting off with a large offset parameter between the inputs and then reducing it monotonically to a small value ε. With this, it was shown that it is possible to converge to the global optimum of nonlinear static and scalar objective function if the algorithm's initial parameter (Δ_{ini}, k, η) have been chosen properly. The algorithm was extended to the constraint optimization problem where a switching adaptation law was used to handle the constraints. It was shown that such an adaptation would lead to the global constrained optimum.

However, it was seen that the proposed algorithm can chatter when the solution is on the boundary of the feasible region. Methods using projection will be investigated in the future to alleviate this difficulty. Extensions of the global optimization framework to the multivariable case, with and without constraints form the next steps of this research work. It has been demonstrated that the proposed method solves that univariate optimization problem rather effectively, at least compared to stochastic approaches such as genetic algorithms (GA) and simulated annealing (SA).

CHAPTER 3 GLOBAL OPTIMIZATION OF TWO-INPUT SYSTEMS USING MULTI-UNIT ADAPTATION

The goal of this chapter is to develop a deterministic global optimization method that can handle the black-box objective functions with two-inputs. Herein, an extremum-seeking strategy that converges to the global optimum of the static nonlinear continuous systems with two decision variables is proposed. The core idea is to iteratively perform univariate global optimization on the circumference of a circle of reducing radius. The radius of the circle is asymptotically reduced to zero.

The outline of this chapter is as follows. Section 3.1 and 3.2 present the new algorithm and establishes its convergence. The proposed methodology is numerically simulated on some illustrative examples in Section 3.3 and compared with other global optimization algorithms in Section 3.4.

3.1 Construction of the algorithm

The main question that is addressed in this chapter is how the global optimum of a two-dimensional map can be found in the multi-unit optimization framework.

Consider the problem of minimizing, $y = f(u_1,u_2)$, where $f: R^2 \rightarrow R$, is a non-convex continuous, nonlinear function. The problem may have multiple local optima, (u_{1k}^*, u_{2k}^*), $k = 1,2, \ldots, n$, but a unique global minimum, (u_1^{**}, u_2^{**}). In the rest of the chapter, it is assumed that the global minimum is unique. The proposed algorithm uses the spirit of the unconstrained scalar global optimization. Here, we need two units referred to as "*a*", "*b*". Let (u_{1a} and u_{2a}) represent the first and second inputs of unit "*a*" and (u_{1b} and u_{2b}) represent the inputs of unit "*b*". The core idea of

this algorithm is to perform global optimization on the circumference of a circle of reducing radius.

It is assumed that the feasible global optimum lies within the initial circle. The radius of this circle is reduced to zero in a predefined fashion. If the centre of the circle is so adapted as to keep the best optimum at the circumference, the algorithm converges to the global optimum of the nonlinear map when the radius goes to zero. In order to mathematically formulate the above mentioned methodology three iterative layers for the new optimization algorithm are considered:

3.1.1 Layer 1: Global optimization along the circumference of a circle

Consider a circle centered at the input values (u_1, u_2) and a radius of Δ (figure 3.1). The multi-unit optimization along the circumference of the circle of reducing radius is repeated iteratively. Let θ_a , θ_b be the angles of the two units. Then the input values of the two units are given by:

$$\begin{cases} u_{1a} = u_1 + \Delta\cos(\theta_a) \\ u_{2a} = u_2 + \Delta\sin(\theta_a) \end{cases} \tag{3.1}$$

$$\begin{cases} u_{1b} = u_1 + \Delta\cos(\theta_b) \\ u_{2b} = u_2 + \Delta\sin(\theta_b) \end{cases} \tag{3.2}$$

The adaptation laws (for minimization) along the circumference of the circle are given by,

$$\theta_a = \theta + \Delta_\theta \quad , \quad \theta_b = \theta - \Delta_\theta \tag{3.3}$$

$$\dot{\theta} = -g_\theta(\Delta_\theta)\, .sign\,(f_a - f_b) \quad , \quad \theta(iT_+) = \pi + \theta_{mi} \tag{3.4}$$

$$\dot{\Delta}_\theta = -g_\theta(\Delta_\theta) \quad , \quad \Delta_\theta(iT_+) = \pi \tag{3.5}$$

where Δ_θ is the offset between two angles θ_a and θ_b. Here $g_\theta(.\ )$ can also be any positive bounded function which is zero only when $\Delta_\theta = 0$. This corresponds to univariate global optimization along the circle of radius Δ using the angle θ. As will be discussed in the next section, the initial

conditions of the equations (3.4) and (3.5) would be reinitialized periodically. The period of each iteration (T) on the circumference of the circle is so chosen that Δ_θ reduces to ε_θ, i.e.,

$$T = \int_{\varepsilon_\theta}^{2\pi} \frac{d\Delta_\theta}{g_\theta(\Delta_\theta)} \tag{3.6}$$

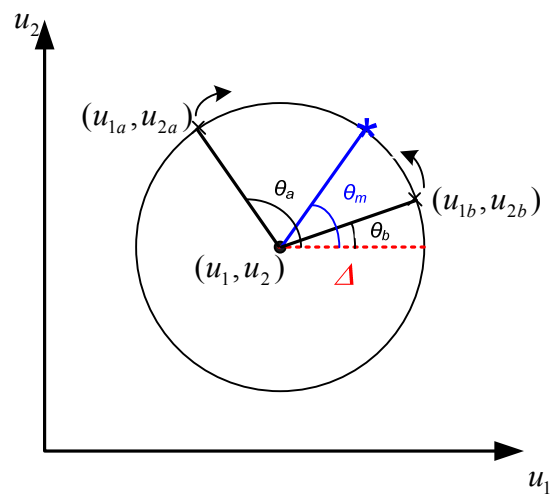

Figure 3.1 Global optimization along the circumference of a circle

3.1.2 Layer 2: Recursive global optimization

Let "i" denote the number of iteration ($i = 0,1,2,\ldots$). At the beginning of each iteration, Δ_θ is initialized to π in order to cover the entire circle. The initial value of θ is so chosen to be the global optimum of the previous iteration. At the beginning of first iteration (i.e., i=0), the initial value of θ_{m0} is arbitrarily set at zero. In the next iterations, θ_{mi} is computed from the values of θ_a , θ_b at the end of the previous iteration as follows,

$$\theta_{mi} = \begin{cases} \theta_a(iT_-) & if \quad f_a < f_b \\ \theta_b(iT_-) & if \quad f_a \geq f_b \end{cases} \tag{3.7}$$

The optimization along the circumference is repeated every T time units. θ_{mi} corresponds to the converged value and would represent the global optimum along the circumference of the circle of iteration "i-1" if $\varepsilon_\theta = 0$.

3.1.3 Layer 3: Reducing the radius of the circle

It is assumed that the feasible global minimum lies within the initial circle (centered at the initial inputs $(u_1(0),u_2(0))$ with the radius of $\Delta(0)$). This radius is monotonically reduced to ε i.e.,

$$\dot{\Delta} = -g(\Delta) \qquad \Delta(0) = \Delta_0 > 0 \tag{3.8}$$

$g(.)$ is a positive bounded function as (2.3 or 2.4) and it is zero only when $\Delta = 0$. The algorithm stops when Δ is reduced to a prefixed value ε. This means the time of integration of the algorithm is given by,

$$T_{tot} = \int_{\varepsilon}^{\Delta_0} \frac{d\Delta}{g(\Delta)} \tag{3.9}$$

This way, the total number of iterations for convergence to the global optimum is fixed to T_{tot} / T. T is defined in equation (3.6). The coordinates which correspond to the global optimum of each iteration are presented as follows (figure 3.2),

$$\begin{aligned} u_{1m} &= u_1 + \Delta\cos(\theta_{mi}) \\ u_{2m} &= u_2 + \Delta\sin(\theta_{mi}) \end{aligned} \tag{3.10}$$

where θ_{mi} corresponds to the global optimum of the previous iteration. The adaptation laws of the centre of the circle are so chosen to keep the global optimum found. In other words, the circle with the radius Δ and centre (u_1,u_2) is contracted in such a manner as to keep (u_{1m},u_{2m}) at the same point i.e., $(\dot{u}_{1m},\dot{u}_{2m}) = (0,0)$. So, the adaptation laws are given by,

$$\begin{aligned} \dot{u}_1 &= -\dot{\Delta}\cos(\theta_{mi}) \qquad , u_1(0) = u_{10} \\ \dot{u}_2 &= -\dot{\Delta}\sin(\theta_{mi}) \qquad , u_2(0) = u_{20} \end{aligned} \tag{3.11}$$

This contraction is depicted in figure 3.2. The centre of the circle is expected to converge to the global optimum of the non-linear map when Δ reaches zero.

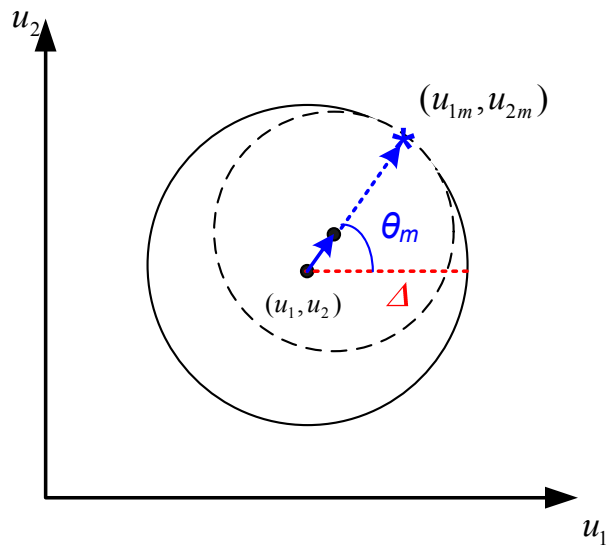

Figure 3.2 Contraction of the circle toward the global optimum

The structure of the above mentioned algorithm is presented in the following flowchart,

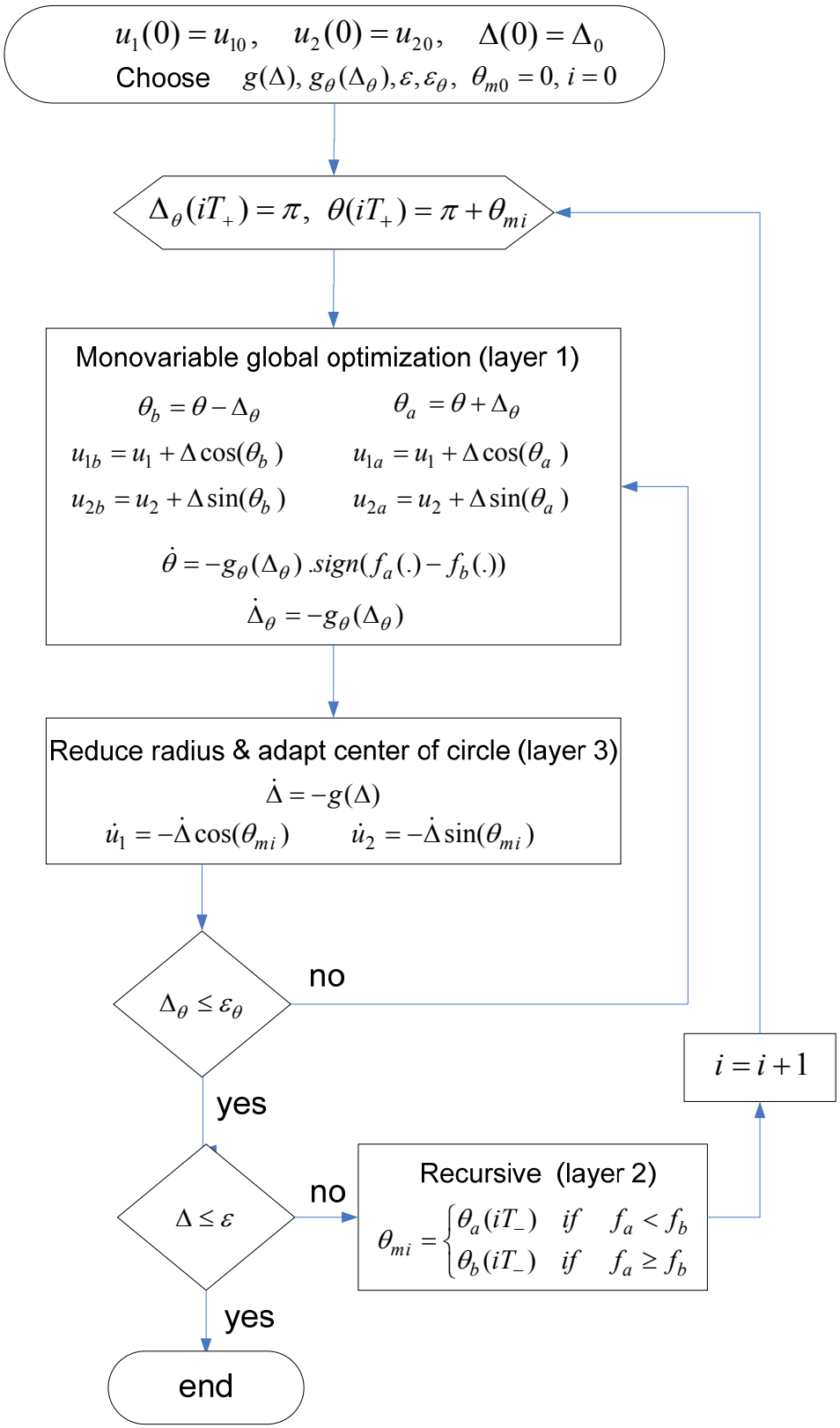

Figure 3.3 Flow chart of the global optimization of two-input systems using multi-units

3.2 Convergence

For the general case, i.e., non-zero values of ε and ε_θ, it cannot be guaranteed that the above scheme is indeed global. However, in the limiting case, it can be shown that this algorithm is capable of avoiding the local optima and converging to the global one.

Theorem 3.2.1 *Consider the multi-unit optimization scheme with the adaptation laws (3.4), (3.5), (3.8) and (3.11), with* $\varepsilon = 0$ *and* $\varepsilon_\theta = 0$ *If (a) f(.) has a unique global minimum, (b)* $T << T_{tot}$ *and (c)* $(u_1^{**} - u_{10})^2 + (u_2^{**} - u_{20})^2 \leq \Delta_0^2$ *, then* $\lim_{t\to\infty} u_1(t) = u_1^{**}$ *and* $\lim_{t\to\infty} u_2(t) = u_2^{**}$.

Proof The proof of this result is based on the fact that there are two different time scales in the algorithm. The fast time scale is that of θ and Δ_θ, while the slow one consists of Δ, u_1 and u_2. Initially it will be shown that the fast time scale keeps θ_{mi} at the global optimum along the circumference of the circle. Secondly, assuming this fact, it will be shown that the shrinking of the circle leads to the global optimum of the problem. In fact, for θ_{mi} to always correspond to the global optimum, it is very important to have a good time scale separation between the two dynamics and the assumption $T << T_{tot}$ is made towards this end.

To prove that θ_{mi} is in fact the global optimum along the circle, the proof follows the lines of proof of theorem 2.1.1 in chapter 2. At each iteration ($i = 1,2,3,\ldots$) the univariate global optimization along the circumference would lead to, $|\theta(t) - \theta_i^{**}| \leq \Delta_\theta(t),\ \forall t \in [iT_+, (i+1)T_-]$, where θ_i^{**} is the angle corresponding to the global optimum along the circle at iteration "i". So, when $\Delta_\theta \to 0$, then $|\theta(t) - \theta i^{**}| = 0$, $\theta_a(iT_-) = \theta_b(iT_-) = \theta_i^{**}$ i.e., at the end of the iteration "i", $\theta_{mi} = \theta_i^{**}$.

Now, in a slower time scale, note that θ_{mi} is the global optimum along the circumference of the shrinking circle at any time. It will be shown by contradiction that $(u_1^{**} - u_1(t))^2 + (u_2^{**} - u_2(t))^2 \leq \Delta^2(t),\ \forall t$. Suppose that at time instant t, $(u_1^{**} - u_1(t))^2 + (u_2^{**} - u_2(t))^2 > \Delta^2(t)$. From the hypothesis,

$(u_1^{**} - u_{10})^2 + (u_2^{**} - u_{20})^2 \leq \Delta_0^2$, there exists a time instant $t = \tau$, such that $(u_1^{**} - u_1(\tau))^2 + (u_2^{**} - u_2(\tau))^2 = \Delta^2(\tau)$. This means that the global optimum of the map (u_1^{**}, u_2^{**}) is on the circle with centre $(u_1(\tau), u_2(\tau))$ and radius $\Delta(\tau)$. So, the angle search θ_{mi} would indeed latch on to this point (since the global optimum of the map is indeed the global optimum along the circumference of the circle). Also, in the next iteration $\theta_b(iT_+) = \theta_{mi}$, $u_{1b} = u_1^{**}$, $u_{2b} = u_2^{**}$ and,

$$\dot{\theta}_b = \dot{\theta} - \dot{\Delta}_\theta = -g_\theta(\Delta_\theta)\, sign(f_a - f_b) + g_\theta(\Delta_\theta) \quad = 0 \tag{3.12}$$

Note that within the iteration, $\dot{\theta}_b = 0$, since $f_b < f_a$ is guaranteed by the uniqueness of the global minimum. Also, at the end of the iteration, θ_b will be retained as θ_{mi} since it has a better function value. Thus, once $(u_1^{**} - u_1(\tau))^2 + (u_2^{**} - u_2(\tau))^2 = \Delta^2(\tau)$, from there on for all $t > \tau$, $u_{1b} = u_1^{**}$ and $u_{2b} = u_2^{**}$.

Since $(u_{1b}, u_{2b}) = (u_1^{**}, u_2^{**})$ is on the circumference of the circle of radius $\Delta(\tau)$, it can be seen that $(u_1^{**} - u_1(t))^2 + (u_2^{**} - u_2(t))^2 = \Delta^2(t)$ for all $t > \tau$, which is a contradiction to the assumption $(u_1^{**} - u_1(t))^2 + (u_2^{**} - u_2(t))^2 > \Delta^2(t)$. So, it is deduced that $(u_1^{**} - u_1(t))^2 + (u_2^{**} - u_2(t))^2 \leq \Delta^2(t), \ \forall t$.

On the other hand, since $\dot{\Delta} = -g(\Delta)$ is bounded, $\Delta(t)$ is a continuous function of t. Also, since $\Delta(0) = \Delta_{ini} > 0$, if $\Delta(t)$ has to become negative at some point of time, it cannot jump but pass through $\Delta = 0$. However when $\Delta = 0$ at $t = \rho$, since $g(0) = 0$, $\dot{\Delta}$ will be 0, which implies $\Delta = 0$ from that time onwards, i.e., $t \geq \rho$. So $\Delta(t)$ can never change sign and, $\Delta(t) \geq 0$ for all t. Also $g(\Delta) > 0$ means that Δ decreases in a strictly monotonic fashion for $\Delta > 0$ and is bounded from below by zero. This concludes that $\lim_{t \to \infty} \Delta(t) = 0$.

Since $(u_1^{**} - u_1(t))^2 + (u_2^{**} - u_2(t))^2 \leq \Delta^2(t), \ \forall t$, it can be said that when $\Delta \to 0$, asymptotically $(u_1^{**} - \lim_{t \to \infty} u_1(t))^2 + (u_2^{**} - \lim_{t \to \infty} u_2(t))^2 \leq \lim_{t \to \infty} \Delta^2(t) = 0$, i.e., $\lim_{t \to \infty} u_1(t) = u_1^{**}$ *and* $\lim_{t \to \infty} u_2(t) = u_2^{**}$. ■

Remark 1 Similar to what is stated in chapter 2 (2.5) , the sign function in the adaptation law (3.4) corresponds to a very high gain and induces a non-Lipschitz nature in the system. This might cause stiffness in integration. A simple solution is to replace the sign function by the hyperbolic tangent as shown below.

$$\dot{\theta} = -g(\Delta_\theta)\tanh\left(\eta\left(\frac{f_a - f_b}{2\Delta_\theta}\right)\right) \tag{3.13}$$

Remark 2 The basic condition for this algorithm to converge to the global optimum is $(u_1^{**} - u_{10})^2 + (u_2^{**} - u_{20})^2 \leq \Delta_0^2$. Since the location of u^{**} is not known a priori, the above condition will be satisfied by choosing a large enough the initial value for Δ_0. The downside of such a choice is that the algorithm requires more time to get to the optimum.

3.3 Illustrative Examples

3.3.1 Test problems

The algorithm is tested on three global optimization problems. The three problems are first presented followed by the results with the proposed algorithm. The performance of the proposed algorithm is compared to the other deterministic and stochastic methods in the next section.

Example 3.3.1 Ackley's function (AC) (Ackley, 1987)

Consider the following nonlinear static map (figure 3.4) with a unique global minimum at $(u_1^{**}, u_2^{**}) = (0, 0)$ and several other local optima.

$$f(u_1, u_2) = -20\exp\left(-0.2\sqrt{\frac{1}{2}\sum_{i=1}^{2} u_i^2}\right) - \exp\left(\frac{1}{2}\sum_{i=1}^{2}\cos(2\pi u_i)\right) + 20 + \exp(1) \tag{3.14}$$

The key aspect of this example is the presence of equal valued and symmetric local optima.

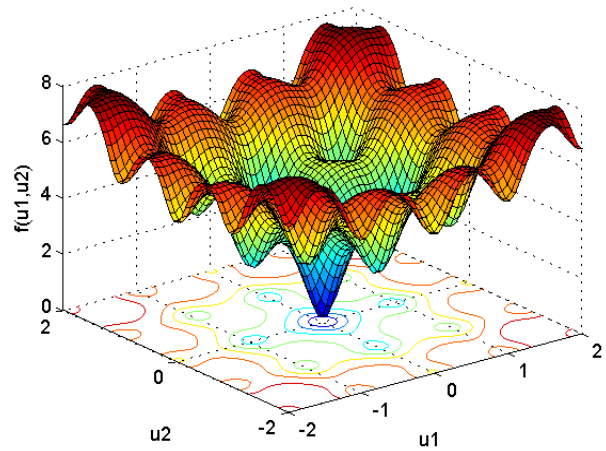

Figure 3.4 Ackley's function for example 3.3.1

Example 3.3.2 Double Summation (DS1) (Strongin and Sergeyev,1992)

$$Min \quad f(u_1,u_2) = -\{(\sum_{i=1}^{7}\sum_{j=1}^{7}[A_{ij}a_{ij}(u_1,u_2)+B_{ij}b_{ij}(u_1,u_2)])^2$$

$$+(\sum_{i=1}^{7}\sum_{j=1}^{7}[C_{ij}c_{ij}(u_1,u_2)+D_{ij}d_{ij}(u_1,u_2)])^2\}^{\frac{1}{2}}$$

$$s.t. \quad 0 \le u_1 \le 1$$

$$0 \le u_2 \le 1$$

$$a_{ij}(u_1,u_2) = \sin(\omega\, i\pi u_1)\sin(\omega\, j\pi u_2) \tag{3.15}$$

$$b_{ij}(u_1,u_2) = \cos(\omega\, i\pi u_1)\cos(\omega\, j\pi u_2)$$

$$f_{global} = f(0.563, 0.25) = -8.2885$$

A_{ij}, B_{ij}, C_{ij}, D_{ij} are random coefficients from interval [-1,1]. ω is a parameter that controls the density of the local optima. The bigger the ω, the number of up and downs increases, thereby increasing the complexity of the test function. The coefficients chosen for A_{ij}, B_{ij}, C_{ij}, D_{ij} are presented in the Appendix III (A). This function with ω =1 is shown in figure 3.5 (in order to better presentation of minima, the function is plotted in inverted form).

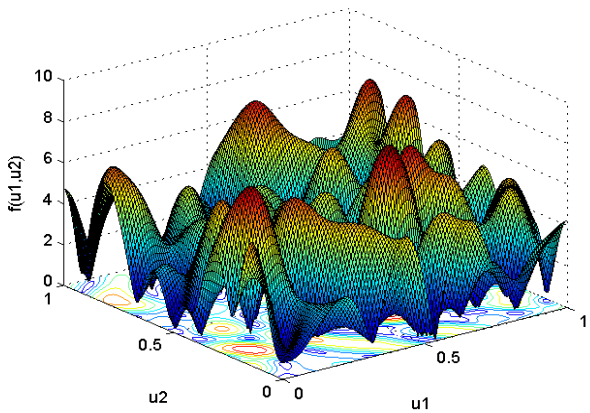

Figure 3.5 Double Summation function with ω=1 (DS1) (inverted plot)

Example 3.3.3 Double Summation (DS2)

The optimization test function in (3.15) with ω = 2 and a new set of coefficients (provided in Appendix III (B)) is depicted in Figure 3.6. The global optimum in this case is f_{global}(0.2349,0.3714) = -9.3633 and it is comparatively more difficult to find.

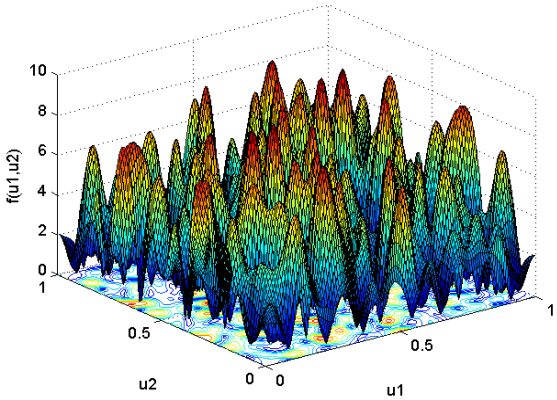

Figure 3.6 Double Summation function with ω=2 (DS2) (inverted plot)

3.3.2 Application of global multi-unit optimization method

For example 3.3.1 (AC), the initial settings u_{10} = -1, u_{20} = -2, and Δ_0 = 5 were considered such that the global minimum among the several other local ones lie in the circle composed by the centre of (u_{10},u_{20}) and the radius of Δ_0. The key condition to satisfy is the inequality $(u_1^{**} - u_{10})^2 +$

$(u_2^{**} - u_{20})^2 \leq \Delta_0^2$ which is in fact verified by choosing Δ_0 big enough. The other parameters used were $k = 0.01$, $k_\theta = 1$, $\varepsilon = 0.007$, $\varepsilon_\theta = 0.01$. As will be discussed later, the maximum variable step size of the integration routine is a key variable for convergence. Here $M_{step\text{-}size}=1$. Applying the global optimization algorithm using multiple units makes the system inputs to converge to the global minimum. The time evolution of the inputs and Δ are shown in figure 3.7.

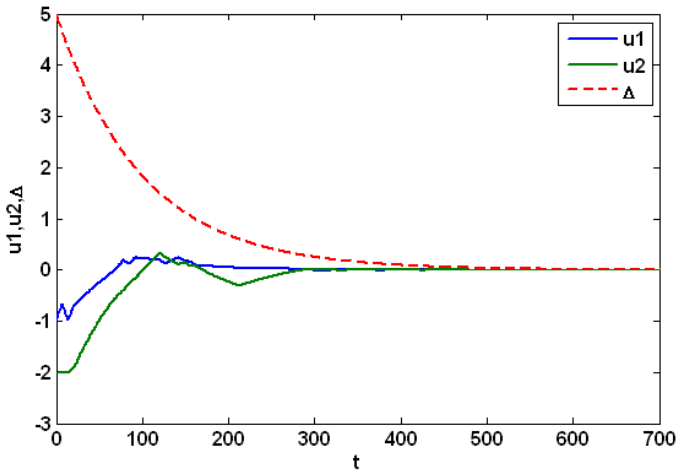

Figure 3.7 Evolution of the inputs and Δ for example AC

As it can be seen from this figure the centre of the circle (u_1,u_2) converges to the global optimum at (0,0), while the radius of the circle (Δ) is asymptotically reduced to ε. The contracted circles are depicted by dash-lines in figure 3.8. As is clear from this figure, the orientation of the shrinking depends on where the global optimum on the circumference is found.

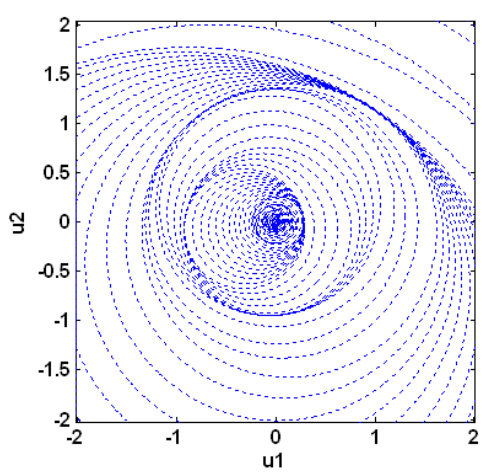

Figure 3.8 Evolution of the circles for example AC

In example 3.3.2 (DS1), the initial conditions are chosen to be u_{10} = 0.5, u_{20} = 0.5 and Δ_0 = 0.7. Other tuning parameters remain the same. As can be seen in figure 3.9, applying the global optimization algorithm using multiple units makes the system input converge to the global minimum at (u_1^{**},u_2^{**})= (0.563,0.25) (red line).

It is important to note that if the condition $T << T_{tot}$ is not satisfied i.e., if the fraction k_θ /k is not large enough, the shrinking of the circle becomes too fast and the global optimum could be missed. In this example, the period of each iteration on the circumference of the circle and the total time of integration were T=6.443sec and T_{tot}=657.88sec respectively and so the number of iterations was T_{tot}/T=102. However, for instance if this ratio is reduced to 10, the multi-unit optimization would converge to the local optimum at (0.67, 0.29) rather than the global one. This is depicted in figure 3.9 using the blue line.

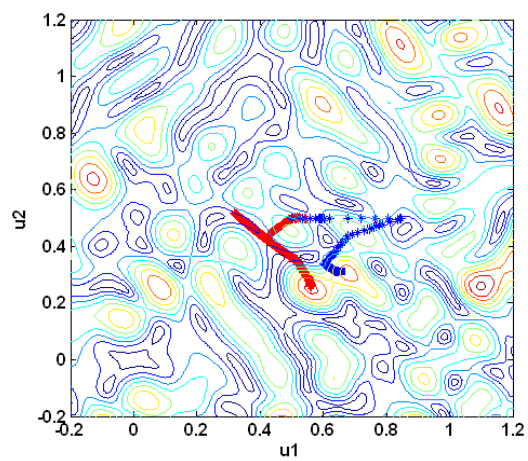

Figure 3.9 Evolution of the centre with $T_{tot}/T=102$ (red line) and with $T_{tot}/T=10$ (blue line) for example DS1

Figure 3.10 shows the time evolution of the angle θ_{mi} (in rad.) for example DS1. The value of this angle evolves depending on the polar position of the global optimum found on the circumference of the shrinking circle. However, its value remains constant within the time interval of each iteration "i". At t=420 sec., the algorithm hits upon the global optimum and θ_{mi} stays fixed at its value of 3.07 rad. Then, the radius shrinks to ε.

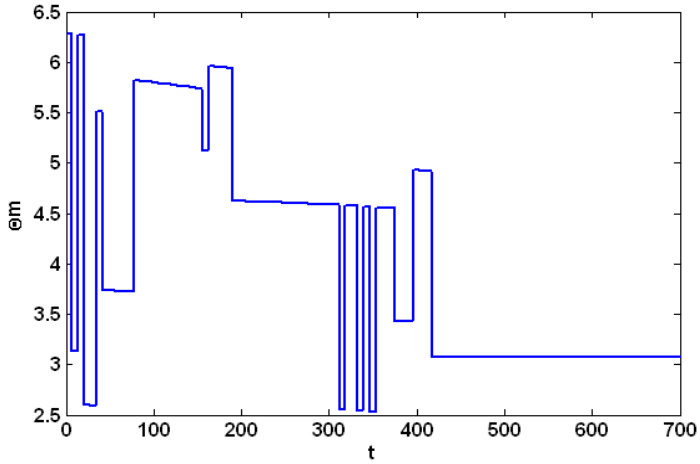

Figure 3.10 Evolution of θ_m for example DS1

For example 3.3.3 (DS2), the movement of the centre of the shrinking circle with exponential and linear adaptation of the radius Δ is presented in figure 3.11. The initial conditions and other tuning parameters remain the same as example DS1. The exponential and linear adaptation of Δ only affects the number of function evaluations needed for convergence, which is 1412 in the exponential case and 1449 in linear case.

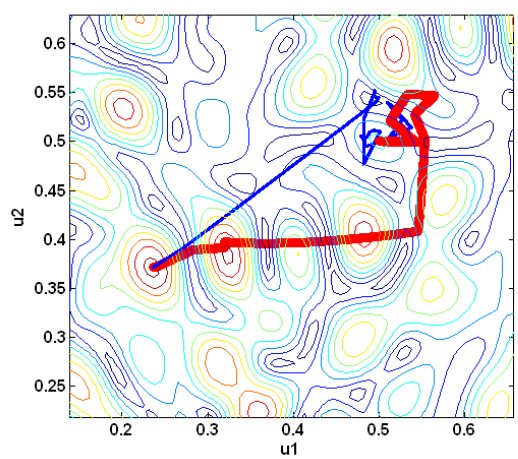

Figure 3.11 Evolution of the centre for exponential (red line) and linear (blue line) decreasing Δ for example DS2

This function has high dense local optima which effectively increases its complexity. In this problem though the area of global optimum is found, the real global minimum is difficult to achieve. In the close vicinity of this global optimum there are many local optima with very small

difference in their objective values. This feature would become important when comparing with other standard global optimization algorithms.

3.4 Comparison with Other Global Optimization Methods

In order to compare the performance of the global optimization by Multi-Units (MU), three other methods are considered. (*i*) Stochastic method - Genetic Algorithm (GA), (*ii*) Stochastic method - Simulated Annealing algorithm (SA) and (*iii*) Deterministic method - DIRECT algorithm (DIviding RECTangles). All the benchmarks presented in Section 4 were worked out.

The genetic algorithm generates a population of points at each iteration and selects the next population by mutation and cross-over (both random operations). The best point in the population approaches an optimal solution. In the simulated annealing algorithm, the optimal solution is gradually approached by systematically decreasing a control parameter (temperature) which determines the probability of accepting a worse solution at any step (Schneider & Kirkpatrick, 2006). In DIRECT (Jones et al., 1993) optimal hyper-rectangles are selected for further partitioning. Although the searching area for GA, SA and DIRECT algorithms is a square, the searching area for MU is a circle. The radius of the initial circle (Δ_0) was chosen to include the square.

The computational results of the (GA) and (SA) algorithms were obtained by using the optimization toolbox 4.0 of *MATLAB* version 7.8.0.347(R2009a). The *MATLAB* version of the DIRECT algorithm developed by Björkman and Hölmstrom (1999) has been used in this study. The multi-unit optimization is simulated by *SIMULINK* version 7.3(R2009a). For stochastic methods, a set of 100 individual implementations of the algorithm was considered.

The number of function evaluations of multi-unit optimization depends on the integrator type (*ode15s, ode23, ode45, ...-ode45* was applied in this thesis). It also depends on the error tolerances used in the integration. The error tolerances of *ode* solvers can be set by the command

odeset in MATLAB. Herein, *RelTol*, is the error tolerance relative to the size of each solution component. It controls the number of correct digits in all solution components, except those smaller than thresholds *AbsTol*. The absolute error tolerance *AbsTol* determines the accuracy when the solution approaches zero. This is a threshold below which the value of the solution component is unimportant. The default values of *RelTol* and *AbsTol* are 1e-3 and 1e-6 respectively. Another important property of the ode solvers which can affect the accuracy of the solution is step-size property. *MaxStep* sets an upper bound on solver step size. The default value of *MaxStep* is $0.1\times\left|t_0 - t_f\right|$ where t_0 and t_f are the starting and final points of integration time respectively. In short the accuracy of the method depends very much on the accuracy of the integrator. The precision of the converged global optimum by multiple units controlled by the integration parameters needs more investigation.

The tuning factors for Genetic algorithm such as (fitness scaling function, crossover function, mutation function, stopping criteria, …) and for simulated annealing algorithm such as (annealing parameters, temperature update function, stopping criteria, …) and for DIRECT algorithm (weight parameter) were set as their default values (Appendix IV). The parameters used in multi unit optimization for all the benchmarks were $k = 0.01$, $k_\theta = 1$, $\varepsilon = 0.007$, $\varepsilon_\theta = 0.01$. For each one of the algorithms, three sets of tests with different tuning parameters were performed.

The following variables during each set of these tests were changed: (*i*) population size (N_{pop}) for genetic algorithm, (*ii*) initial temperature ($T_{initial}$) for simulated annealing (*iii*) number of iterations ($N_{iteration}$) for DIRECT and (*iv*) maximum of variable step-size ($M_{step\text{-}size}$) for the proposed algorithm.

The initial starting point has been chosen at the centre of the square/circular bounds of the inputs for each benchmark. The initial starting point in each problem has been considered the same for all of the algorithms. Convergence is quantified in terms of percent error from the global optimum as follows (Björkman and Hölmstrom ,1999):

$$E = 100\frac{f_{\min} - f_{global}}{\left|f_{global}\right|} \tag{3.16}$$

where f_{min} is the best function value at some point in the search and f_{global} represents the known global optimum of the function. The results of deterministic algorithms were compared in terms

of the percentage of error from the global optimum (Table 3.1). On the other hand for probabilistic algorithms, the results were presented in terms of percentage of successful convergence to the global optimum (with $E\%$ <0.01) in table 3.2. The corresponding average number of function evaluations needed for global convergence is presented in table 3.3 as a measurement of the efficiency of these algorithms.

Table 3.1 Comparison between MU and DIRECT in terms of percent error from the global optimum

Global optimization method		E%		
		AC	*DS1*	*DS2*
Deterministic	**MU1** $M_{step\ size}=0.5$	0.012	0.002	0.05
	MU1 $M_{step\ size}=1$	0.012	0.002	0.32
	MU3 $M_{step\ size}=10$	0.013	0.002	4.628
	DIR1 $N_{iteration}=200$	0	0	0
	DIR2 $N_{iteration}=50$	0	0	7.2462
	DIR3 $N_{iteration}=20$	0	0.048	8.3473

Table 3.2 Comparison of GA and SA in terms of successful global convergence ($E\% < 0.01$)

Global optimization method		Successful global convergence %		
		AC	*DS1*	*DS2*
Probabilistic	**GA1** $N_{pop}=200$	100	100	58
	GA2 $N_{pop}=50$	100	57	25
	GA3 $N_{pop}=20$	100	35	17
	SA1 $T_{initial}=200$	100	88	69
	SA2 $T_{initial}=20$	100	80	60
	SA3 $T_{initial}=2$	48	59	44

Table 3.3 Comparison between global MU, DIRECT, GA and SA in terms of average number of function evaluations

	Global optimization method	Average number of function evaluations		
		AC	*DS1*	*DS2*
Deterministic	**MU1** $M_{step\ size}=0.5$	1409	1467	1412
	MU2 $M_{step\ size}=1$	794	810	835
	MU3 $M_{step\ size}=10$	735	753	771
	DIR1 $N_{iteration}=200$	82005	4877	3941
	DIR2 $N_{iteration}=50$	15243	851	641
	DIR3 $N_{iteration}=20$	2651	153	167
Probabilistic	**GA1** $N_{pop}=200$	10400	10400	10400
	GA2 $N_{pop}=50$	2600	2600	2600
	GA3 $N_{pop}=20$	1040	1040	1040
	SA1 $T_{initial}=200$	1505	1960	1958
	SA2 $T_{initial}=20$	1290	1987	1878
	SA3 $T_{initial}=2$	2781	1763	1816

As it is shown in table 3.1, for the three test problems the global optimization method with multiple units converges to the global optimum with a good accuracy. The error percentage from the global optimum in DIRECT algorithm decreases with increasing number of iterations. The error being zero means that the optimum coincides with one of the grid points used.

While the optimization by deterministic algorithms MU and DIRECT converge to the same optimum during all tests of each set, the algorithms GA and SA show a probabilistic nature in their final convergence. This means that the rate of successful global convergence for MU and DIRECT with certain parameters is always 100%. However, table 3.2 shows a significant percentage of convergence to a false optimum when the population size of genetic algorithm and the value of initial temperature for simulated annealing algorithm are chosen very small.

Increasing the population size of GA and the value of initial temperature in SA, enhance the percentage of the global convergence

The number of function evaluations in genetic algorithm depends on the size of initial population. In genetic algorithm when the population size is made large, the number of function evaluation significantly increases. However, there is no correlation between the number of function evaluations and the value of initial temperature in simulated annealing algorithm.

As is clear from table 3.3 the average number of function evaluations for global optimization by multi-units and DIRECT algorithm was generally less than the one for genetic algorithm and simulated annealing in all of the tests. This can indeed be expected since the dimension is small.

From Table 3, it can be seen that the number of function evaluations in DIRECT algorithm increases with the number of iterations and increase the precision. Also, for multi-unit optimization, decreasing the maximum step-size increases the number of function evaluations and also improves the precision. Now, comparing the two methods for a given amount of precision, it can be seen that the proposed method is at least as good as, or better than, the DIRECT algorithm. Because of the grids used by the DIRECT algorithm, sometimes, the error goes to zero, but it still requires considerable amount of evaluations to ascertain global optimality.

Figure 3.12 illustrates the sampling points on nonlinear map (AC) calculated by DIRECT algorithm with $N = 50$ iterations (15243 function evaluations). Figure 3.13 shows the dispersion of sampling points from the same test problem with multi unit global optimization with $M_{step\text{-}size}=1$ (794 function evaluations). In this particular example the DIRECT algorithm coincides the global optimum quite fast with one of the grid points used. However, this technique requires more function evaluations compared to the multi unit optimization in order to cover the feasibility region to increase the reliability of the final result as the global optimum.

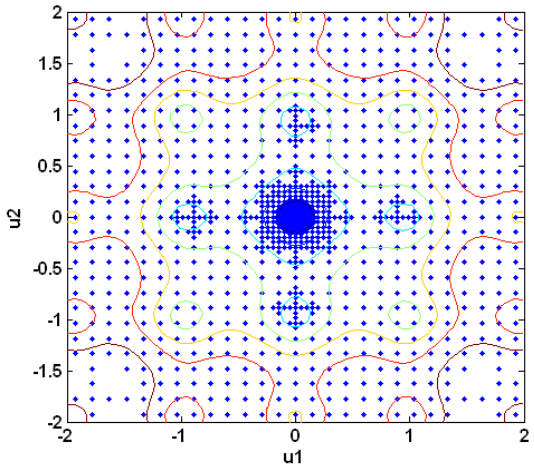

Figure 3.12 Sampling points by DIRECT algorithm for AC

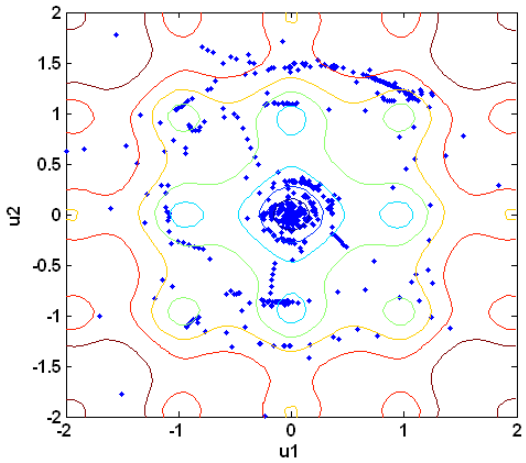

Figure 3.13 Sampling points by multi-unit optimization for AC

It can be seen from comparing figure 3.13 with figure 3.8 that in multi-unit optimization the sampling points are mostly concentrated on the areas where circles are jammed (local optima). However, all the local optima on the nonlinear map are not covered depending on the initial conditions of the optimization problem. This is because the adaptation laws of the multi-unit optimization reduce the variable step size taken by integration in these areas. The integration step-size is so adapted to avoid unnecessary function evaluations.

3.5 Conclusion

A model-free, unconstrained global optimization method using multi-units was proposed by controlling the centre of a shrinking circle on which the gradient is estimated using finite difference between two units operating with an offset. For two-input systems, the technique was performed on the circumference of a circle of reducing radius. The offset parameter between the inputs of the two units was monotonically and iteratively reduced to zero where the radius of the circle was monotonically shrinking in parallel. With this, it was shown that it is possible to converge to the global optimum of any two dimensional nonlinear static objective function, provided the global optimum is present in the initial circle composed by the centre of the initial inputs and the initial value of the radius.

Effectiveness of the proposed algorithm was shown using three benchmark two-input examples. Also, it was compared with other deterministic and stochastic algorithms and it was shown that the proposed algorithm is indeed efficient in terms of number of function evaluations. Development of the proposed algorithm to systems with more than two degrees of freedom is the next steps considered in this research framework.

CHAPTER 4 GLOBAL OPTIMIZATION OF THREE-INPUT SYSTEMS USING MULTI-UNIT ADAPTATION

In this chapter, the multi-unit global optimization is generalized to the three dimensional systems (systems with three inputs). The spirit of the previous algorithm in chapter 3 is kept in generalization of the method to three dimensions. Furthermore, in three-variable optimization the circular path -where the multi-unit optimization takes place- rotates systematically on a spherical space in order to cover the circumference of the sphere. On the other hand, the radius of the sphere is monotonically decreased in order to cover the feasible region. In other words, the entire area of the sphere is going to be swept by these rotations deterministically. To do this, a sufficient time scale separation between different dynamics of the algorithm is necessary. The three-dimensional rotation matrix (Rodrigues' rotation formula) is employed to handle such a rotation of the two units on a shrinking sphere. The contribution in this chapter is related to the adaptive laws between the dynamics of two units in three-variable space. The results are illustrated on common global optimization test-problems.

4.1 Construction of the algorithm

Consider the problem of minimizing, $y = f(u_1, u_2, u_3)$, where $f: R^3 \rightarrow R$, is a non-convex continuous and nonlinear function. The problem may have multiple local optima, $(u_{1k}^*, u_{2k}^*, u_{3k}^*)$ $k = 1, 2 \ldots m$, but a unique global minimum, $(u_1^{**}, u_2^{**}, u_3^{**})$. In the rest of the paper, it is assumed that the global minimum is unique.

The proposed algorithm uses the spirit of the global optimization by multi-units for two-input systems which has been presented in chapter 3. Similarly, we need two units referred to as "*a*" and "*b*". The idea for generalizing the algorithm to three dimensions is to perform the global optimization on the circumference of a circle of variant radius which systematically rotates on a shrinking spherical space. The main question that is addressed in this chapter is how the global

optimum of a three-dimensional nonlinear map can be found trough this adaptive rotation in the multi-unit optimization framework.

It is assumed that the feasible global optimum lies within the initial spherical space. The radius of the sphere is reduced to zero in a predefined fashion. If the centre of the sphere is so adapted as to keep the best optimum at the circumference of the rotating circle on the sphere, the algorithm converges to the global optimum of the nonlinear map when the radius of the sphere goes to zero. In order to mathematically formulate the above mentioned methodology, five iterative layers for the proposed optimization algorithm are considered.

4.1.1 Layer 1: Global optimization along the circumference of a rotating circle on a three-dimensional sphere

Consider a three-dimensional spherical space centered at the input values (u_1, u_2, u_3) and a radius of Δ. The multi-unit optimization along the circumference of the circle of variant radius placed on this sphere is repeated iteratively. Let (φ, θ_a) and (φ, θ_b) be the angles related to the position of two units in the spherical space. Then the input values of the two units are given by,

$$\mathbf{u}_a = \mathbf{u} + \mathbf{M}\mathbf{u}_{ra} \quad , \quad \mathbf{u}_b = \mathbf{u} + \mathbf{M}\mathbf{u}_{rb} \tag{4.1}$$

where,

$$\mathbf{u}_a = \begin{bmatrix} u_{1a} \\ u_{2a} \\ u_{3a} \end{bmatrix} \quad , \quad \mathbf{u}_b = \begin{bmatrix} u_{1b} \\ u_{2b} \\ u_{3b} \end{bmatrix} \quad , \quad \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{4.2}$$

$$\mathbf{u}_{ra} = \begin{bmatrix} \Delta\cos(\varphi) \\ \Delta\sin(\varphi)\cos(\theta_a) \\ \Delta\sin(\varphi)\sin(\theta_a) \end{bmatrix} \quad , \quad \mathbf{u}_{rb} = \begin{bmatrix} \Delta\cos(\varphi) \\ \Delta\sin(\varphi)\cos(\theta_b) \\ \Delta\sin(\varphi)\sin(\theta_b) \end{bmatrix} \tag{4.3}$$

Here, (φ,θ) represent the elevation and azimuth angles in the three-dimensional polar system. Azimuth is the angle between the perpendicular projection of a given vector and the reference vector on the reference plane (in the spherical coordination). The elevation angle φ represents the angle between the given vector and the reference vector which is perpendicular to the reference plane. The angle φ is common between the coordinates of vectors "*a*" and "*b*". Therefore, the end points of the vectors representing the two units move around the circumference of a circle as θ_a and θ_b vary. **M** is a rotation matrix which is changed iteratively. This matrix will be widely discussed in the next layers. If we consider $\mathbf{M}=\boldsymbol{I}_{3\times3}$ (where $\boldsymbol{I}_{3\times3}$ is the identical matrix), then equations (4.1), (4.2) and (4.3) represent a circular path on the sphere of radius Δ which is made by evolution of the input values "$\mathbf{u}_a$" and "$\mathbf{u}_b$". The adaptation laws (for minimization) along the circumference of the circle on the sphere stay the same as previous chapter,

$$\theta_a = \theta + \Delta_\theta \quad , \quad \theta_b = \theta - \Delta_\theta \tag{4.4}$$

$$\dot{\theta} = -k_\theta \Delta_\theta \; sign\,(f_a - f_b) \tag{4.5}$$

$$\dot{\Delta}_\theta = -k_\theta \Delta_\theta \tag{4.6}$$

Here, these equations correspond to the univariate global optimization along the circle of radius $\Delta\sin(\varphi)$ on a three-dimensional sphere.

4.1.2 Layer 2: Recursive global optimization along the circle

The initialization of the initial conditions for the adaptation laws (4.5) and (4.6), the period of each iteration on the circumference of the circle (T) and the number of iterations (i) stay the same as previous chapter,

$$\theta(iT_+) = \pi + \theta_{mi} \tag{4.7}$$

$$\Delta_\theta(iT_+) = \pi \tag{4.8}$$

In this case, θ_{mi} represents the Azimuth angle in three dimensional space which is associated to the best optimum in each iteration. At the beginning of first iteration, the initial value of θ_{m0} is arbitrarily set at zero. In the next iterations, $\theta_{mi,}$ is computed from the values of the angle θ corresponding to units "*a*" and "*b*" at the end of the previous iteration as follows,

$$\theta_{mi} = \begin{cases} \theta_a(iT_-) & if \quad f_a < f_b \\ \theta_b(iT_-) & if \quad f_a \geq f_b \end{cases} \tag{4.9}$$

Here f_a and f_b are objective function values provided by two units "*a*" and "*b*" which have three inputs. The period of each iteration on the circumference of the circle (T) is calculated by,

$$T = T_\theta = \int_{\varepsilon}^{2\pi} \frac{d\Delta_\theta}{k_\theta \Delta_\theta} = \frac{1}{k_\theta}(\ln \frac{2\pi}{\varepsilon}) \tag{4.10}$$

4.1.3 Layer 3: Expansion and contraction of the radius of the circle

The circle on the sphere which was introduced in layer 1 is the locus where multi-unit optimization takes place. In order to deterministically cover the whole feasible three dimensional region of optimization it is proposed that the mentioned circle is rotated on the sphere systematically. The sphere is contracted towards global optimum in parallel with a slower rate as it will be seen in layer 5.

In three dimensional space, any rotation can be represented by a unit vector as the axis of rotation and an angle of revolution about that vector. Thus, given a point with orientation $\mathbf{p}_1$ and another point with a different orientation $\mathbf{p}_2$ in three dimensional space, we can always find an axis and angle which will rotate from orientation $\mathbf{p}_1$ to orientation $\mathbf{p}_2$. The concept of the (angle-axis) rotation in three dimensional space can be expressed by the "Rodrigues" rotation matrix formula (Rodrigues, 1816). **M** is the Rodrigues rotation matrix that operates on any point on the sphere by rotating it about the unit vector $\mathbf{\Gamma} = (\Gamma_1,\Gamma_2,\Gamma_3)$ by the angle γ. This matrix is given by the representation (Appendix I),

$$\mathbf{M} = \begin{bmatrix} \cos\gamma + \Gamma_1^2(1-\cos\gamma) & \Gamma_1\Gamma_2(1-\cos\gamma) - \Gamma_3\sin\gamma & \Gamma_2\sin\gamma + \Gamma_1\Gamma_3(1-\cos\gamma) \\ \Gamma_3\sin\gamma + \Gamma_1\Gamma_2(1-\cos\gamma) & \cos\gamma + \Gamma_2^2(1-\cos\gamma) & -\Gamma_1\sin\gamma + \Gamma_2\Gamma_3(1-\cos\gamma) \\ -\Gamma_2\sin\gamma + \Gamma_1\Gamma_3(1-\cos\gamma) & \Gamma_1\sin\gamma + \Gamma_2\Gamma_3(1-\cos\gamma) & \cos\gamma + \Gamma_3^2(1-\cos\gamma) \end{bmatrix} \tag{4.11}$$

As will be seen in the rest of the algorithm, the angle and axis of rotation (γ, $\mathbf{v}$) are systematically changed from each iteration to another. This way, the whole surface of the spherical space is adaptively covered by the circle of layer 1. The initial axis of rotation for making the circular path of the multi-unit optimization is considered on $\mathbf{u}_1$-axis. Hence, the rotation about $\mathbf{u}_1$-axis is inherently a part of the basic algorithm in order to make a circular path and there is no need for compensation of the rotation about this dimension. However, the dynamics of the angle φ is changed linearly or exponentially. This angle repeatedly covers a complete range from 2π to ε and causes that the radius of the circle on the sphere varies between $\Delta sin(\varepsilon)$ and Δ. The exponentially decreasing dynamics of the angle φ is considered to be,

$$\dot{\varphi} = -k_\varphi \varphi \tag{4.12}$$

$$k_\varphi \ll k_\theta \tag{4.13}$$

Inequality (4.13) indicates an adequate time scale separation between the dynamics of the angles θ and φ. Here, $k_\varphi > 0$ is a parameter that determines the rate at which φ is reduced to ε. The extra rotations of the circular path of the units on the sphere which is caused by angle φ in three-input systems affect the dynamics of the algorithm which were established for two-input-systems in previous section. Therefore, the compensation of the extra rotations of the circle on the sphere is necessary to latch on the global optimum found on the circular path. This compensation is done by changing the axis and angle of rotation at each iteration.

Each iteration of the algorithm indicates a complete circular rotation of the units on the sphere. The converging point on the sphere where the last rotating circle ends, corresponds to particular latitude and longitude which is identified as the global optimum found on the surface of the

sphere. The coordinates of the point which correspond to the global optimum of each iteration are presented as,

$$\mathbf{u}_{mi} = \mathbf{u} + \mathbf{M}_i \mathbf{u}_{rmi} \tag{4.14}$$

$$\mathbf{u}_m = \begin{bmatrix} u_{1m} \\ u_{2m} \\ u_{3m} \end{bmatrix} \quad , \quad \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \quad , \quad \mathbf{u}_{rmi} = \begin{bmatrix} \Delta\cos(\varphi_{mi}) \\ \Delta\sin(\varphi_{mi})\cos(\theta_{mi}) \\ \Delta\sin(\varphi_{mi})\sin(\theta_{mi}) \end{bmatrix} \tag{4.15}$$

where φ_{mi} represents the second polar angle which is associated to the global optimum at each iteration. At the beginning of first iteration, the initial value of φ_{m0} is arbitrarily set at zero. In the next iterations, φ_{mi} is fixed equal to the value of the angle φ at the end of the previous iteration and remain fixed until the end of that iteration, i.e.,

$$\varphi_{mi} = \varphi(iT_{_}) \tag{4.16}$$

M*i* is the rotation matrix which is updated at each iteration and (φ_{mi}, θ_{mi}) corresponds to the global optimum of the previous iteration. Figure 4.1 depicts one iteration of the multi-unit global optimization by units "*a*" and "*b*" on the rotating circular path on the sphere.

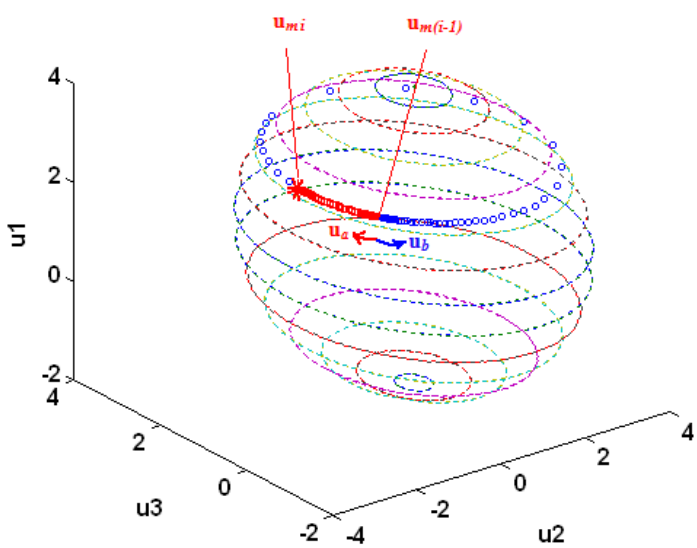

Figure 4.1 Global optimization along the circumference of a rotating circle on the sphere

In order to realize the above mentioned method, the angle and axis of such a dynamic continuous rotation must be so adapted that (φ_{mi}, θ_{mi}) of each iteration corresponds to the global optimum found at the previous iteration ($\varphi_{m(i-1)}$, $\theta_{m(i-1)}$). This is necessary in order to latch the rotating circles on the global optimum found at each iteration. Therefore, the starting point of the multi-unit optimization on the circle of iteration "*i*" must be equal to the global optimum found at the previous iteration "*i-1*", i.e., $\mathbf{u}_{m(i-1)}$. This can be shown as follows,

$$\mathbf{u}_{starta(i)} = \mathbf{u}_{startb(i)} = \mathbf{u}_{m(i-1)} \quad , \quad \forall i \geq 1 \tag{4.17}$$

$$\mathbf{u}_a(iT_+) = \mathbf{u}_b(iT_+) = \mathbf{u}_{m(i-1)} \tag{4.18}$$

$$\cancel{\mathbf{u}} + \mathbf{M}_i\mathbf{u}_{ra}(iT_+) = \cancel{\mathbf{u}} + \mathbf{M}_i\mathbf{u}_{rb}(iT_+) = \cancel{\mathbf{u}} + \mathbf{M}_{i-1}\mathbf{u}_{rm(i-1)} \tag{4.19}$$

Since the center of the sphere is not affected by the rotation matrix $\mathbf{M}_i$, the above equation becomes,

$$\mathbf{M}_i\mathbf{u}_{ra}(iT_+) = \mathbf{M}_i\mathbf{u}_{rb}(iT_+) = \mathbf{M}_{i-1}\mathbf{u}_{rm(i-1)} \tag{4.20}$$

If we consider the following substitutions in this equation,

$$\mathbf{v}_{si} = \mathbf{u}_{ra}(iT_+) = \mathbf{u}_{rb}(iT_+) \tag{4.21}$$

$$\mathbf{v}_{m(i-1)} = \mathbf{M}_{i-1}\mathbf{u}_{rm(i-1)} \tag{4.22}$$

it can be recast as,

$$\mathbf{M}_i\mathbf{v}_{si} = \mathbf{v}_{m(i-1)} \tag{4.23}$$

At the beginning of first iteration, the initial rotation matrix $\mathbf{M}_0$ is arbitrarily set at identical matrix $\mathbf{I}_{3\times3}$. In the next iterations, $\mathbf{M}i$ is iteratively computed as follows in the rest of this layer. Equations (4.9) and (4.16) show that (φ_{mi}, θ_{mi}) indicate the converged value by multi-units and would represent the global optimum along the circumference of the circle of iteration "*i-1*" if ε = 0. Therefore, according to (4.3), (4.15), equations (4.21) and (4.22) become,

$$\mathbf{v}_{si} = \begin{bmatrix} \Delta\cos(\varphi_{mi}) \\ \Delta\sin(\varphi_{mi})\cos(\theta_{mi}) \\ \Delta\sin(\varphi_{mi})\sin(\theta_{mi}) \end{bmatrix} \tag{4.24}$$

$$\mathbf{v}_{m(i-1)} = \mathbf{M}_{i-1} \begin{bmatrix} \Delta\cos(\varphi_{m(i-1)}) \\ \Delta\sin(\varphi_{m(i-1)})\cos(\theta_{m(i-1)}) \\ \Delta\sin(\varphi_{m(i-1)})\sin(\theta_{m(i-1)}) \end{bmatrix} \tag{4.25}$$

The main question that comes to mind from interpretation of equation (4.23) would be: what is the rotation matrix $\mathbf{M}i$ that when multiplies by the starting point ($\mathbf{v}_s$) on the rotating circle of the next iteration (i), it becomes equal to the global optimum ($\mathbf{v}_m$) found on the circle of previous iteration (i-1). On the other hand, Δ, $\varphi_{m(i-1)}$, φ_{mi}, $\theta_{m(i-1)}$, θ_{mi} and the rotation matrix of previous iteration $\mathbf{M}_{i-1}$ are known, therefore $\mathbf{v}_s$ and $\mathbf{v}_{m(i-1)}$ are known vectors.

As was mentioned before, any rotation in three dimensions is described by a rotational angle about an axis. Given the vectors $\mathbf{v}_s$ and $\mathbf{v}_m$, we need to find an axis and angle which will rotate $\mathbf{v}_s$ to $\mathbf{v}_m$. It is clear that the angle of rotation can be calculated by "dot" product of these vectors. The axis of rotation would be the line which is perpendicular to both vectors (cross product of these vectors). The angle and the axis of rotation at each iteration (γ_{mi}, $\mathbf{\Gamma}_{mi}$) are defined as,

$$\gamma_{mi} = \arccos\left(\frac{\mathbf{v}_{si}^T \mathbf{v}_{m(i-1)}}{\left|\mathbf{v}_{si}\right|\left|\mathbf{v}_{m(i-1)}\right|} \right) \tag{4.26}$$

$$\mathbf{\Gamma}_{mi} = \mathbf{v}_{si} \times \mathbf{v}_{m(i-1)} \tag{4.27}$$

The angle and axis of rotation (γ_{mi}, $\mathbf{\Gamma}_{mi}$) is adequate information needed for constructing the elements of rotation matrix $\mathbf{M}_i$ at each iteration. This way the starting point of the next iteration begins from the global optimum found during previous iteration and the effect of decreasing angle φ on the position of the circle on the sphere is cancelled out at each iteration. As a result, the algorithm rotates the whole circle by the angle of γ_{mi} about the axis $\mathbf{\Gamma}_{mi}$. In layer 5 we will see

that how the circle where the multi-unit optimization takes place always latches on the global optimum found in the previous iteration.

4.1.4 Layer 4: Recursive global optimization along the sphere

The initialization of the initial condition for the adaptation law (4.12) is set such that the dynamics of φ is repeatedly decreased from $2\pi + \varphi_{mi}$ to ε, i.e.,

$$\varphi(t_\varphi) = 2\pi + \varphi_{mi} \qquad , t_\varphi = t\Big|_{(\varphi \le \varepsilon)} \tag{4.28}$$

The period of a complete extraction-contraction of the circle on the sphere (T_φ) can be calculated by,

$$T_\varphi = \int_{\varepsilon}^{2\pi} \frac{d\varphi}{k_\varphi \varphi} = \frac{1}{k_\varphi}(\ln \frac{2\pi}{\varepsilon}) \tag{4.29}$$

Furthermore, since $k_\varphi \gg k_\theta$, it can be concluded that $T_\varphi \gg T_\theta$. If this condition is satisfied, then after $j = T_\varphi / T_\theta$ iterations the rotating "expanding-contracting" circles complete a whole coverage of the surface of the sphere from zero to 2π.

4.1.5 Layer 5: Reducing the radius of the sphere

It is assumed that the feasible global minimum lies within the initial sphere (centered at the initial inputs $(u_1(0), u_2(0), u_3(0))$ with the radius of $\Delta(0)$). This radius is monotonically reduced to zero i.e.,

$$\dot{\Delta} = -k_s \Delta \qquad \Delta(0) = \Delta_0 > 0 \tag{4.30}$$

$$k_s \gg k_\varphi \gg k_\theta \tag{4.31}$$

$k_s > 0$ is a parameter that determines the rate at which Δ is reduced to ε. On the other hand, the adaptation laws of the centre of the sphere are so chosen to keep $\mathbf{u}_{mi}$ at the same point. To do this, the derivative of the coordinates of the global optimum found at each iteration (equation 4.14) must be set at zero i.e.,

$$\dot{\mathbf{u}}_{mi} = \dot{\mathbf{u}} + \overline{\dot{\mathbf{M}}_i \mathbf{u}_{rmi}} = 0 \tag{4.32}$$

$$\dot{\mathbf{u}} = -\overline{\dot{\mathbf{M}}_i \mathbf{u}_{rmi}} = -(\dot{\mathbf{M}}_i \, \mathbf{u}_{rmi} + \mathbf{M}_i \, \dot{\mathbf{u}}_{rmi}) \tag{4.33}$$

In order to being latch on global optimum, $\dot{\theta}_{mi}$ and $\dot{\varphi}_{mi}$ intuitively must be zero within each iteration. That is why these angles are kept constant within each iteration equal to the angles of the global optimum found from previous iteration (equations 4.9 and 4.16). The angle and axis of rotation (γ_{mi}, $\mathbf{\Gamma}_{mi}$) are also constant within each iteration, thereby $\dot{\mathbf{M}}_{mi} = 0$. The only variable in equation (4.33) which is dynamically changed within each iteration is Δ. So, the rate of change in the coordinates of the center of the sphere is continuously calculated as follows,

$$\dot{\mathbf{u}} = -\overline{\dot{\mathbf{M}}_i \mathbf{u}_{rmi}} = -\dot{\Delta}\, \mathbf{M}_i \begin{bmatrix} \cos(\varphi_{mi}) \\ \sin(\varphi_{mi})\cos(\theta_{mi}) \\ \sin(\varphi_{mi})\sin(\theta_{mi}) \end{bmatrix} \tag{4.34}$$

This means that the sphere with the radius Δ and centre (u_1, u_2, u_3) is contracted towards the global optimum of the nonlinear function in such a manner as to keep the best optimum found within each iteration. The algorithm stops when Δ is reduced to a prefixed value ε. The rate of change of Δ must be slower than the dynamics of the angles θ and φ. Note that the slowest time scale that exists in this algorithm corresponds to dynamics of Δ. The total time of integration (T_{tot}) is calculated by the following formula,

$$T_{tot} = \int_{\varepsilon}^{\Delta_0} \frac{d\Delta}{k_s \Delta} = \frac{1}{k_s}(\ln \frac{\Delta_0}{\varepsilon}) \tag{4.35}$$

The number of total iterations can be calculated by $N = T_{tot} / T_\theta$, where T_θ is the period of each circular rotation of the units on the sphere.

4.1.6 Illustration and explanation of the layers of optimization

The contraction of the sphere is depicted in figures 4.2 and 4.3. Since the space of rotations is continuous, the centre of the sphere is expected to converge to the global optimum of the non-linear map when Δ reaches zero. As it is clear from these figures, the continuously shrinking spheres never come out of the prior one.

At t=0 consider the initial rotation in which the axis of rotation lies on the u_1-axis. Beginning with the initial conditions set at ((φ_{m0}, θ_{m0}) = (0, 0)), the point on the north pole of the sphere centered at (u_1, u_2, u_3) would be the starting point of multi-unit optimization. Any specific rotation realized by the rotation matrix **M** can be specified by a circular slice on the sphere. The multi-unit optimization along the circumference of the rotating circle is repeated every T time units. The position angle θ is periodically adapted based on the multi-unit adaptation laws along the circumference of the circle by two units (covering the horizontal rotation about the u_1-axis from the range of ε to 2π Radians). Meanwhile the rotating circle is systematically moved from the north towards the south pole by decreasing the perpendicular position angle φ from 2π to ε Radians which is the complete perpendicular rotation. The initial rotating circle made by multiplication of the rotational matrix **M** on this point would be very small near the north pole. Accordingly, at t=0 the three-input systems "a" and "b" start to rotate about the $\mathbf{u}_1$-axis moving in a circular path with a very small radius.

As the position angle φ changes, the circle moves southward and the radius of the rotating circle becomes larger. The dynamics of the radius of the rotating circle on the sphere depends on dynamics of Δ and φ. If the centre of the sphere is so adapted as to keep the best optimum at the circumference of the rotating circle, the starting point of each rotation on the sphere is specified by the global optimum found on the circle of previous rotation. Having φ decreased at the same time with a slower speed, the rotating circles continue to enlarge until the circle whose circumference is equal to the equator of the sphere is reached. From now on, the radii of the rotating circles become smaller. If an enough time scale separation between the dynamics of the angles θ and φ is considered, once this radius of the circle shrinks to zero, the entire surface of the

sphere is covered by these rotating circles. Thus, the set of expanding and contracting rotating circles would cover the entire surface of the sphere. Figure 4.4 depicts a part of repetitive expansion and contraction of the rotating circles latched on the global optimum found on their circumference iteratively ($\mathbf{u}_{mi}$).

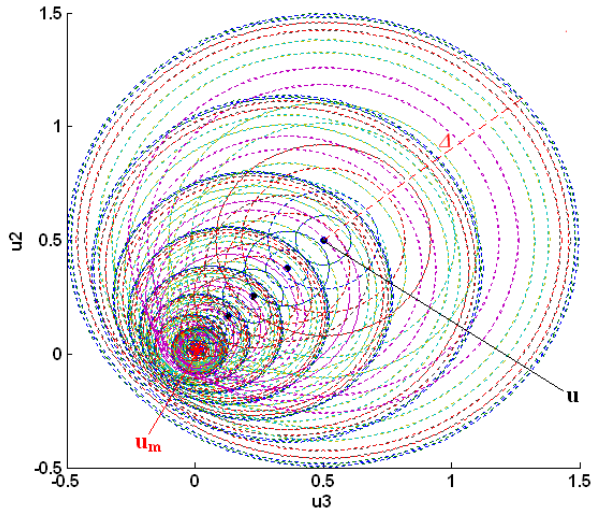

Figure 4.2 Contraction of the sphere towards the global optimum (top view)

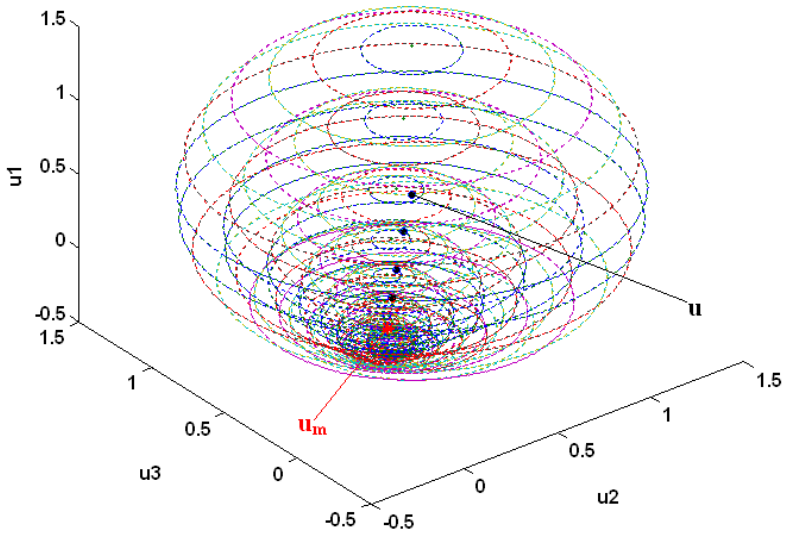

Figure 4.3 Contraction of the sphere towards the global optimum (side view)

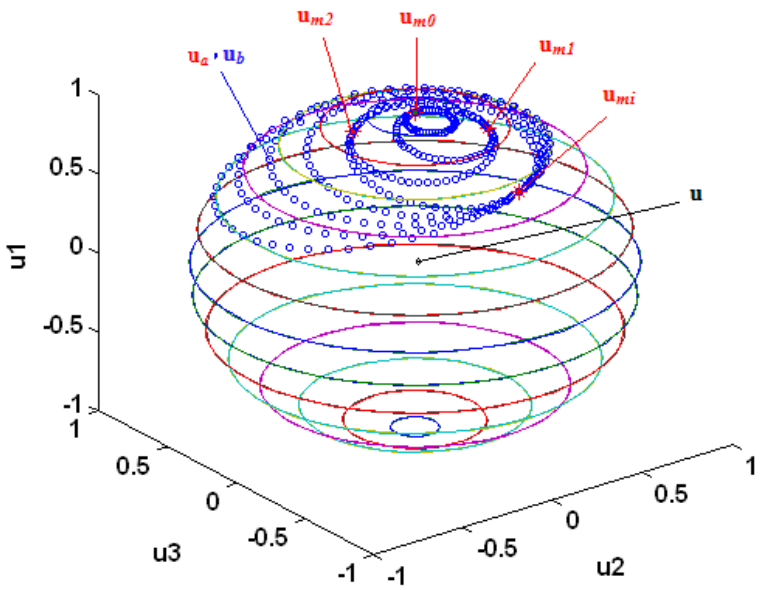

Figure 4.4 Illustration of repetitive expansion and contraction of the rotating circles on the sphere toward the global optimum ($\mathbf{u}_{mi}$)

The structure of the above mentioned algorithm is presented in flowchart at figure 4.5.

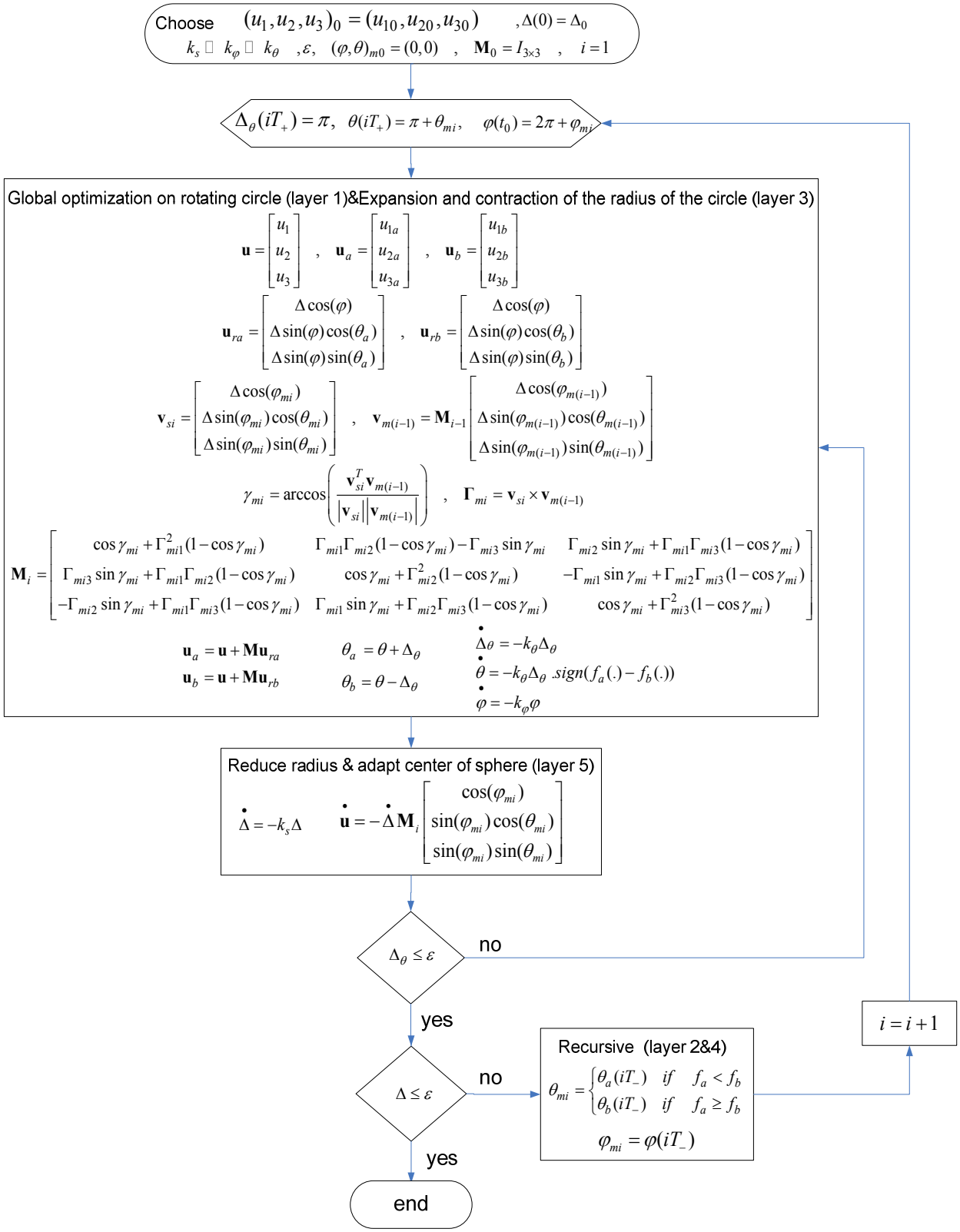

Figure 4.5 Flow chart of the global optimization of three-input systems using multi-units

4.2 Convergence

Consider the multi-unit optimization scheme with the adaptation laws (4.5),(4.6),(4.12),(4.30) and (4.33) with ε=0. If (a) $f(.)$ has a unique global minimum, (b) $k_s \ll k_\varphi \ll k_\theta$ and (c) $(u_1^{**} - u_{10})^2 + (u_2^{**} - u_{20})^2 + (u_3^{**} - u_{30})^2 \leq \Delta_0^2$, then $\lim_{t\to\infty} u_1(t)=u_1^{**}$, $\lim_{t\to\infty} u_2(t)=u_2^{**}$ *and* $\lim_{t\to\infty} u_3(t)=u_3^{**}$. Similarly, the basic condition for this algorithm to converge to the global optimum is $(u_1^{**} - u_{10})^2 + (u_2^{**} - u_{20})^2 + (u_3^{**} - u_{30})^2 \leq \Delta_0^2$. Since the location of $\mathbf{u}^{**}$ is not known a priori, the above condition will be satisfied by choosing a large enough the initial value for Δ_0. The downside of such a choice is that the algorithm requires more time to get to the optimum.

4.3 Illustrative examples

4.3.1 Test problems

Ten test problems were selected for evaluation of the developed algorithm in three variables case. These test problems have been widely used in global optimization literature. The analytical results for four of these problems are presented in details. The results of multi-unit global optimization are then compared with other competitive methods in this class. For the other six test problems only the numerical results are given for briefness. These global optimization benchmarks have been presented in appendix II.

Example 4.3.1 Levy function (Levy and Montalvo, 1985)

$$
\begin{aligned}
Min \quad & f(u) = \sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 (1 + 10\sin^2(\pi y_i + 1)) \\
& \qquad + (y_n - 1)^2 (1 + \sin^2(2\pi u_n)) \\
s.t. \quad & -10 \le u_i \le 10 \\
& y_i = 1 + \frac{u_i - 1}{4}, \\
& u \in \mathbb{R}^n \\
& f_{global} = f(1,1,...,1) = 0
\end{aligned}
\tag{4.36}
$$

This function has several local minima.

Example 4.3.2 Hartman function H3 (Dixon and Szegö, 1978)

$$
\begin{aligned}
Min \quad & f(u) = -\sum_{k=1}^{4} C_k \exp\left(-\sum_{i=1}^{3} A_{ki} (u_i - P_{ki})^2 \right) \\
s.t. \quad & 0 \le u_i \le 1 \\
& C_k > 0, \quad u, A_{ki}, P_{ki} \in \mathbb{R}^n \\
& f_{global} = f(0.114614,\ 0.555649,\ 0.852547) = -3.86278
\end{aligned}
\tag{4.37}
$$

The coefficients c_k, a_{ki}, p_{ki} have been given in appendix C. This function has four local minima.

Example 4.3.3 Perm function (Deng and Ferris, 2007)

$$
\begin{aligned}
Min \quad & f(u) = \sum_{k=1}^{n} \left[\sum_{i=1}^{n} (i + \beta) \left\{ (u_i)^k - (i)^{-k} \right\} \right]^2 \\
s.t. \quad & -1 \le u_i \le 1 \\
& u \in \mathbb{R}^n \\
& f_{global} = f(1, (2)^{-1}, (3)^{-1}, ..., (i)^{-1}) = 0
\end{aligned}
\tag{4.38}
$$

For a given $\beta=10$ and $n=3$, this function has the minimum objective value $f(\mathbf{u}^*) = 0$ at $u_i^* = (i)^{-1}$. The domain of definition is $x \in [-1,1]^3$. The smaller the β, the number of optima increases, thereby increasing the complexity of the test function.

Example 4.3.4 (Rosenbrock, 1960)

$$
\begin{aligned}
&Min \quad f(u) = \sum_{i=1}^{n-1}\left[100\left(u_i^2 - u_{i+1}\right)^2 + \left(u_i - 1\right)^2\right] \\
&s.t. \quad -5 \le u_i \le 10 \\
&\qquad\qquad u \in \mathbb{R}^n \\
&\qquad\qquad f_{global} = f(1,1,\ldots,1) = 0
\end{aligned}
\tag{4.39}
$$

This function has several local minima.

4.3.2 Application of global multi-unit optimization method

For example 4.3.1 (Levy function), the initial settings (u_{10}, u_{20}, u_{30}) = (3, 4.7, -5), and Δ_0 = 10 were considered such that the global minimum among the several other local ones lies in the sphere composed by the centre of (u_{10}, u_{20}, u_{30}) and the radius of Δ_0. The key condition to satisfy is the inequality $(u_1^{**} - u_{10})^2 + (u_2^{**} - u_{20})^2 + (u_3^{**} - u_{30})^2 \le \Delta_0^2$ which is in fact verified by choosing Δ_0 big enough. The other parameters used for all examples were k_s = 0.001, k_φ = 0.01, k_θ = 0.1, ε = 0.01.

Applying the global optimization algorithm using multiple units makes the system inputs to converge to the global minimum. The time evolution of the inputs and Δ are shown in figure 4.6.

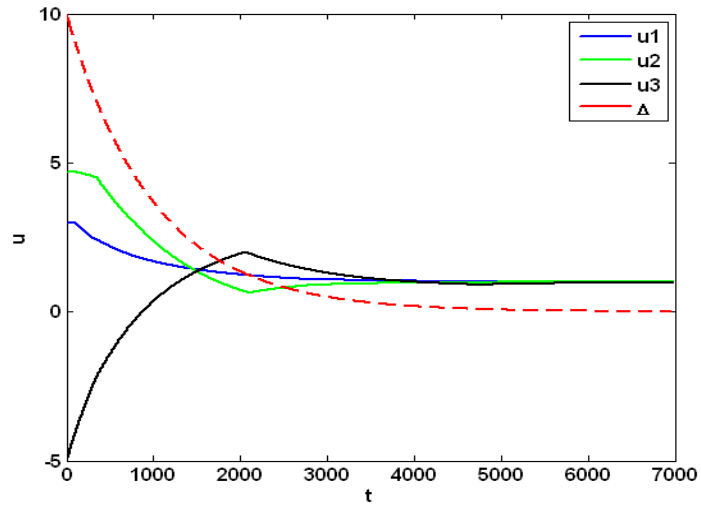

Figure 4.6 Evolution of the inputs and Δ for example Levy function

As it can be seen from this figure the centre of the circle (u_1,u_2,u_3) converges to the global optimum at (1,1,1), while the radius of the circle (Δ) is asymptotically reduced to ε. The shrinking spheres are depicted in figure 4.7. The orientation of the shrinking depends on where the global optimum on the circumference of the rotating circle on sphere is found.

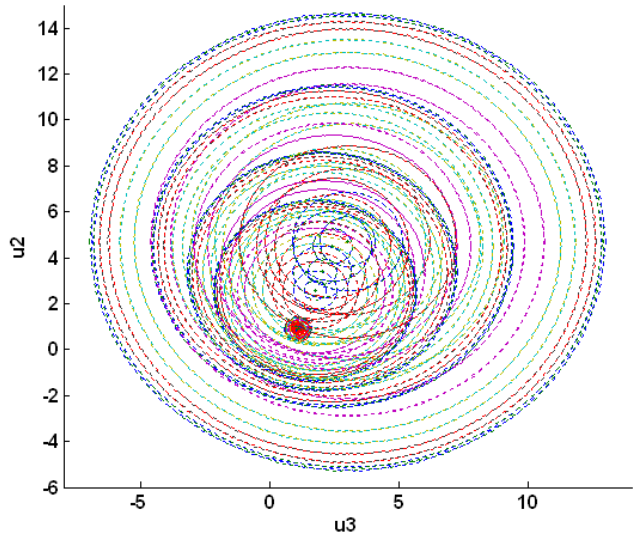

Figure 4.7 Evolution of the spheres for example Levy function

In example 4.3.2 (Hartman function), the initial conditions are chosen to be $(u_{10}, u_{20}, u_{30}) = (0.9, 0.1, 0.3)$ and $\Delta_0 = 1$. Other tuning parameters remain the same. As can be seen in figure 4.8, applying the global optimization algorithm using multiple units makes the system input to converge to the global minimum at $(u_1^{**}, u_2^{**}, u_3^{**}) = (0.2143, 0.5533, 0.8519)$ (red line).

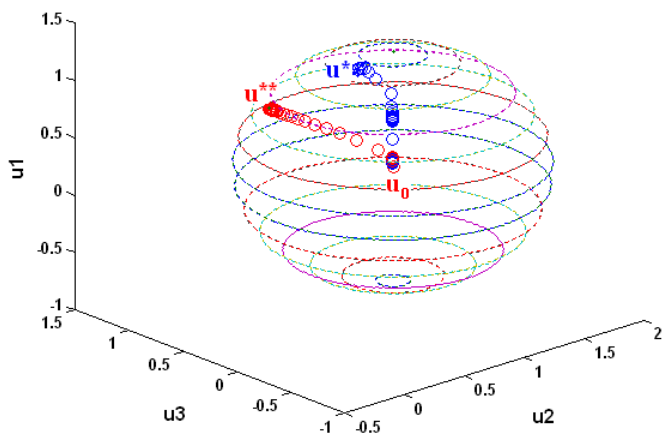

Figure4.8 Evolution of the centre of sphere with $T_{tot}/T=66$ (red line) and with $T_{tot}/T=10$ (blue line) for example Hartman function

Similar to the two-variable optimization, if the condition $T << T_{tot}$ is not satisfied i.e., if the fraction k_θ / k_φ or k_φ / k_s are not large enough, the shrinking of the circle (or sphere) becomes too fast and the global optimum could be missed. In this example, the period of each iteration on the circumference of the rotating circle, the period of a complete extraction-contraction of this circle on the sphere and finally the total time of integration were T_θ=60.443sec, T_φ=600.443sec and T_{tot}=4000.6052sec respectively.

It is clear that after $T_\varphi / T_\theta = 10$ iterations the rotating "expanding-contracting" circles complete a whole coverage of the surface of the shrinking sphere from zero to 2π. The number of total iterations (rotations on the sphere) was $T_{tot}/ T_\theta = 66$. However, for instance if this ratio is reduced to 10 (with $k_s = 0.01$, $k_\varphi = 0.05$, $k_\theta = 0.1$), the multi-unit optimization would converge to the local optimum at (0.8775, 0.4061, 1.0285) rather than the global one. This is depicted in figure 4.8 using the blue line.

Figure 4.9 shows the time evolution of the angles θ_{mi} , φ_{mi} for example 4.3.3 (Perm function) for the first forty iterations. The initial inputs were (u_{10}, u_{20}, u_{30}) = (0.2,-0.3, 0) and Δ_0 = 2. The values of these angles evolve depending on the polar position of the global optimum found on the circumference of the rotating circle on the shrinking sphere. However, their value remain constant within the time interval of each iteration "*i*".

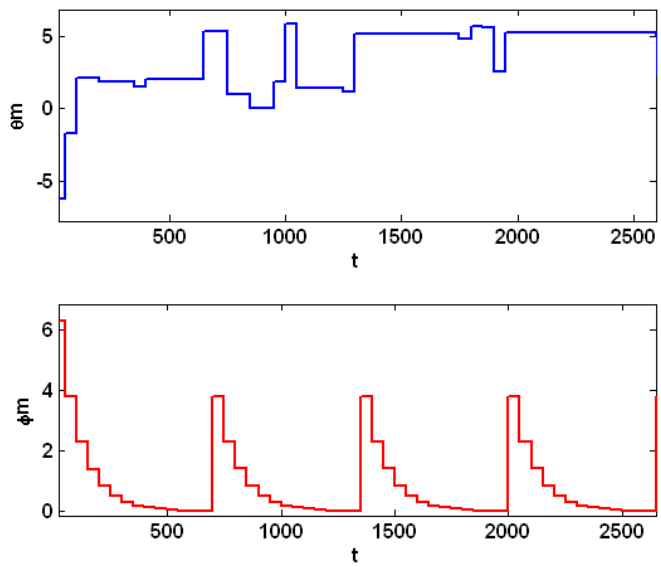

Figure 4.9 Evolution of θ_m and φ_m for example Perm function

For example 4.3.4 (Rosenbrock function), the movement of the centre of the shrinking circle with two different maximum step size ($M_{step\text{-}size}$) of the integrator in the adaptation laws is presented in figure 4.10 *ODE23* has been chosen for integration. The initial conditions and other tuning parameters were (u_{10}, u_{20}, u_{30}) = (3,-0.3,-3) and Δ_0 = 5. The maximum step size of the integration affects the precision and the number of function evaluations needed for convergence. If $M_{step\text{-}size}$ = 1, the number of function evaluations is 2218 and the algorithm converges to (0.9924, 0.9848, 0.9697) with 2e-4 difference from the real global optimum (red line). If $M_{step\text{-}size}$ = 5, the number of function evaluations is 2208. However, the algorithm converges to (0.9708, 09423, 0.8875) which is less precise with 0.0042 error (blue line).

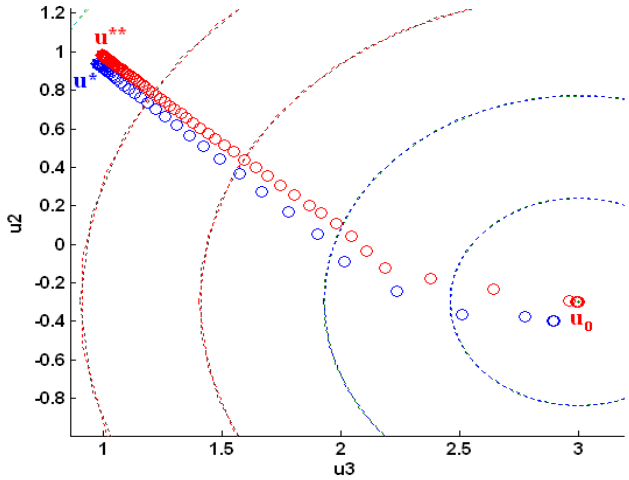

Figure 4.10 Evolution of the centre of sphere for $\boldsymbol{M}_{step\ size} = 1$ (red line) and $\boldsymbol{M}_{step\ size} = 5$ (blue line) for example Rosenbrock function

4.4 Comparison with other global optimization methods

In this comparison, two benchmarking criteria are taken into account according to Baritompa and Hendrix (2005):

1) "Effectiveness" which indicates whether the real global optimum has been found with a given approximation and reflects the accuracy of the solution. This benchmarking criterion can be evaluated by the number of successful convergences to the global optimum by a certain algorithm. To assess the effectiveness of an optimization method, usually the algorithm application is repeated several times.
2) "Efficiency" which reflects the computational cost required to global convergence and is measured by the number of function evaluations used by the algorithm.

In order to demonstrate the performance of the proposed method (similar to previous chapters), the results of multi-unit global optimization are compared with DIRECT algorithm, genetic algorithm and simulated annealing for all considered well-known test problems presented in section 4.3 and appendix II. All the algorithms have been implemented using the Optimization

Toolbox 4.0 of *MATLAB* version 7.8.0.347(R2009a) which is a registered trademark of the MathWorks. The *MATLAB* version of the DIRECT algorithm developed by Björkman and Hölmstrom (1999) has been also used in this study. For stochastic methods, a set of 100 individual implementations of the algorithm was considered.

In three variables optimization, although the searching area for GA, SA and DIRECT algorithms is a cube, the searching area for MU is a sphere. The radius of the initial sphere (Δ_0) was so chosen to include the cube. The initial input values and the other setting parameter values of the algorithm for different tests remain the same as previous section. The initial input values for all benchmarks are given in appendix II. The initial starting point of optimization has been chosen at the centre of the cube/sphere bounds of the inputs for each benchmark. The initial starting point in each problem has been considered the same for all of the algorithms.

The tuning factors for genetic algorithm and simulated annealing were set as their default values. Similarly, for each one of the algorithms, three sets of tests with different tuning parameters were performed. The following variables during each set of these tests were changed: (*i*) population size (N_{pop}) for genetic algorithm, (*ii*) initial temperature ($T_{initial}$) for simulated annealing (*iii*) number of iterations ($N_{iteration}$) for DIRECT and (*iv*) type of the integrator (*ODE*) for the proposed algorithm. In this study, the maximum variable step size of the integration routine for adaptation laws in multi-unit optimization is set fix (in contraction with chapter 3). The variable step size is adapted automatically during the integration routine by MATLAB. The integration step-size is so adapted to avoid unnecessary function evaluations. However, in this study the algorithm is analyzed in terms of number of function evaluations obtained by deferent integrator types (*ode45, ode113, ode23*). More information about these ordinary differential equation solvers can be found in MATLAB website. Convergence is quantified in terms of percent error from the global optimum as (3.16). However, for cases that the real global optimum value was zero, the obtained objective function value is recorded (i.e., the difference between the function and its accurate global value). Similar to Chapter 3, the effectiveness of deterministic algorithms were compared in terms of the percentage of error from the global optimum (Table 4.1). The effectiveness of probabilistic algorithms were evaluated in terms of percentage of successful convergence to the

global optimum (with $E\% < 0.01$) in table 4.2. The efficiency of all the algorithms is compared by evaluating the corresponding average number of function evaluations needed for global convergence (Table 4.3).

Table 4.1 and Table 4.3 show that the accuracy of multi-unit optimization and the number of function evaluations in this algorithm depend on the integrator type (*ode45, ode113, ode23*). The step size and precision of the integrator in these solvers varies differently. These integrator routines are designed for solving different nonlinear differential equations with different characters in MATLAB. On the other hand, depending on the position of the center of initial cube which is split by DIRECT algorithm, the number of function evaluations needed for global convergence varies. As it was shown in chapter 3, this technique requires more function evaluations compared to the multi unit optimization in order to cover the feasibility region to increase the reliability of the final result as the global optimum.

Table 4.3 presents the number of function evaluations needed for these convergences. As is evident from the results presented in this table, the DIRECT algorithm converges very fast and obtains accurate results. The convergence of the DIRECT algorithm to the global optimum is guaranteed in the limit when the number of iterations goes to infinity ($N_{iteration} \rightarrow \infty$). However, to achieve results in finite time the number of iterations must be finite. In the DIRECT algorithm, it is obvious that the greater the number of iterations, the more accurate the results will be. However, a greater number of iterations increases the number of function evaluations. Nevertheless, still with less computational cost the results are quite accurate. On the other hand, numerical comparisons of multi-unit optimization with GA and SA clearly indicate the advantage of MU in terms of accuracy of solutions. Herein, MU usually takes a smaller number of function evaluations comparing these two methods. In terms of successful global convergence, SA showed a better performance in comparison with GA (Table 4.2). The reason for this in general is that the genetic algorithm needs a greater number of function evaluations to improve the performance.

Table 4.1 Comparison between MU and DIRECT in terms of percent error from the global optimum

Global optimization method		E%			
		Levy	***Hartman***	***Perm***	***Rosenbrock***
Deterministic	**MU1** *ODE 45*	1e-4	2e-1	6e-5	3e-4
	MU2 *ODE 113*	3e-4	2e-1	5e-5	2e-4
	MU3 *ODE 23*	4e-6	2e-1	2e-4	2e-4
	DIR1 $N_{iteration}=200$	1e-9	1e-3	3e-7	3e-6
	DIR2 $N_{iteration}=50$	1e-8	2e-3	2e-3	2e-2
	DIR3 $N_{iteration}=20$	1e-5	4e-3	2e-1	4e-2

Table 4.2 Comparison between GA and SA in terms of successful global conv. ($E\% < 0.01$)

Global optimization method		Successful global convergence %			
		Levy	***Hartman***	***Perm***	***Rosenbrock***
Probabilistic	**GA1** $N_{pop}=200$	100	100	100	88
	GA2 $N_{pop}=50$	100	100	84	60
	GA3 $N_{pop}=20$	100	95	60	50
	SA1 $T_{initial}=200$	100	100	100	83
	SA2 $T_{initial}=20$	100	100	90	75
	SA3 $T_{initial}=2$	100	100	90	66

Table 4.3 Comparison between global MU, DIRECT, GA and SA in terms of average number of function evaluations

	Global optimization method	Average number of function evaluations			
		Levy	*Hartman*	*Perm*	*Rosenbrock*
Deterministic	**MU1** *ODE 45*	6595	5009	5042	5745
	MU2 *ODE 113*	3813	2961	2828	2569
	MU3 *ODE 23*	3366	2169	2011	2218
	DIR1 $N_{iteration}=200$	18499	6165	10293	11641
	DIR2 $N_{iteration}=50$	2425	951	1549	1917
	DIR3 $N_{iteration}=20$	381	303	417	541
Probabilistic	**GA1** $N_{pop}=200$	10400	10400	12500	13580
	GA2 $N_{pop}=50$	2600	2600	3450	4090
	GA3 $N_{pop}=20$	1040	1040	1500	1622
	SA1 $T_{initial}=200$	2433	2292	3053	3432
	SA2 $T_{initial}=20$	2417	2210	2552	3551
	SA3 $T_{initial}=2$	3245	3145	3195	5083

The comparison results are mostly similar to the results which were obtained in optimization of two-input systems. As shown in table 4.1, for all test problems the global optimization method with multi-units converges to the global optimum with a good accuracy. The error percentage from the global optimum in this algorithm depends on the accuracy of the integrator type and the error tolerances used in the integration. The error percentage from the global optimum in DIRECT algorithm decreases with increasing the number of iterations. Again, comparing the two methods for a given amount of precision, it can be seen that the proposed method is at least as good as, or better than, the DIRECT algorithm. The optimization by deterministic algorithms MU and DIRECT converge to the same optimum during all tests of each set. However, table 4.2 shows that the algorithms GA and SA show a probabilistic nature in their final convergence and converge to a false optimum when the population size of the genetic algorithm and the value of initial temperature for simulated annealing algorithm are chosen to be very small. Again,

increasing the population size of GA and the value of initial temperature in SA, enhances the percentage of the global convergence. It is clear from table 4.3 that the number of function evaluations for global optimization by multi-units has been increased compared to optimization of the two-dimensional functions in chapter 3. This can be expected since the dimension is increased by one. However, the number of function evaluations in this algorithm is still competitive with the other algorithms depending on the solver type chosen for integration.

According to table 4.1 and 4.4 it is difficult to find a correlation between the integrator type and precision in this algorithm. In general, it cannot be concluded which solver gives more precise solutions for all functions. This is because of the fact that *ODE*s in *MATLAB* solve the differential equations with different routines and their precision depends on different factors including stiffness of the function. A prior knowledge about some characteristics of the objective function would be very helpful to choose the right integrator in different optimization problems.

ODE45 and *ODE23* are one-step solvers meaning that in computation current solution they need only the solution at the previous time point. *ODE113* is a multi-step solver meaning that it needs the solutions at several preceding time points to compute the current solution (MATLAB version 7.6.0.324(R2008a) as a trademark of the Mathworks). In general, *ODE45* is the best solver to apply as a first try for most problems. *ODE23* may show more efficiency than *ODE45* in the presence of moderate stiffness. *ODE113* may be more efficient than *ODE45* for solving computationally intensive problems. The order of accuracy for *ODE45, ODE23* and *ODE113* are medium, low and low to high respectively. The average number of function evaluations for global optimization by DIRECT algorithm was generally less than the other methods when $N_{iteration}$ is chosen small. However, very small numbers of iterations in this algorithm decreases the accuracy (e.g., Perm function in table 4.1). In general, because of the grids used by the DIRECT algorithm it is capable to converge fast with a good accuracy. However, it still requires more function evaluations to ascertain global optimality. As was mentioned in chapter 3, the number of function evaluations in genetic algorithm depends on the size of initial population. In genetic algorithms, the larger the population size, the more is the number of function evaluation. Similar to optimization of two-input systems, there is no correlation between the number of function evaluations and the value of initial temperature in simulated annealing algorithm. The comparison results between multi-unit and DIRECT algorithms for more test problems listed in

appendix II are presented in tables 4.4 and 4.5. The set of 100 individual implementations of the stochastic methods for these problems was omitted for the sake of briefness.

Table 4.4 Comparison between MU and DIRECT in terms of percent error from the global optimum for the problems listed in appendix II

	Global optimization method	E%					
		Sum-Square	***Zakharov***	***Griewank***	***Ackley***	***Sphere***	***Dixon-Price***
Deterministic	**MU1** *ODE 45*	2e-5	1e-4	4e-2	2e-1	3e-5	6e-2
	MU2 *ODE 113*	3e-6	5e-5	1e-1	1e-2	2e-5	4e-5
	MU3 *ODE 23*	1e-4	1e-4	6e-2	1e-2	5e-5	4e-4
	DIR1 $N_{iteration}$=200	1e-10	1e-10	2e-1	3e-9	0	1e-10
	DIR2 $N_{iteration}$ =50	2e-10	4e-10	2e-1	2e-8	1e-10	1e-8
	DIR3 $N_{iteration}$ =20	1e-4	9e-2	2e-1	6e-3	3e-6	2e-2

Table 4.5 Comparison between global MU and DIRECT in terms of average number of function evaluations for the problems listed in appendix II

	Global optimization method	Average number of function evaluations					
		Sum-Square	***Zakharov***	***Griewank***	***Ackley***	***Sphere***	***Dixon-Price***
Deterministic	**MU1** *ODE 45*	6991	6763	9492	6747	6273	6563
	MU2 *ODE 113*	5223	5839	4283	3252	3287	2546
	MU3 *ODE 23*	2823	3381	3677	2632	2331	2546
	DIR1 $N_{iteration}$=200	22055	20235	10405	6967	20611	17275
	DIR2 $N_{iteration}$ =50	2947	2135	1689	943	2861	1983
	DIR3 $N_{iteration}$ =20	467	409	393	251	427	389

4.5 Discussion on extension of the algorithm to higher dimensions

Consider an n-dimensional spherical space centered at the input values $(u_1, u_2 \ldots u_n)$ and a radius of Δ. Let $(\Phi_{1a}, \Phi_{2a} \ldots \Phi_{(n-1)a})$ and $(\Phi_{1b}, \Phi_{2b} \ldots \Phi_{(n-1)b})$ be the angles of the two units. Then the input values of the two units are given by,

$$\mathbf{u}_a = \mathbf{u} + \mathbf{M}\mathbf{u}_{ra} \quad , \quad \mathbf{u}_b = \mathbf{u} + \mathbf{M}\mathbf{u}_{rb} \tag{4.40}$$

where,

$$\mathbf{u}_a = \begin{bmatrix} u_{1a} \\ u_{2a} \\ u_{3a} \\ \vdots \\ u_{(n-1)a} \\ u_{na} \end{bmatrix} \qquad \mathbf{u}_b = \begin{bmatrix} u_{1b} \\ u_{2b} \\ u_{3b} \\ \vdots \\ u_{(n-1)b} \\ u_{nb} \end{bmatrix} \qquad \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{(n-1)} \\ u_n \end{bmatrix} \tag{4.41}$$

$$\mathbf{u}_{ra} = \begin{bmatrix} \Delta\cos(\phi_{1a}) \\ \Delta\sin(\phi_{1a})\cos(\phi_{2a}) \\ \Delta\sin(\phi_{1a})\sin(\phi_{2a})\cos(\phi_{3a}) \\ \vdots \\ \Delta\sin(\phi_{1a})\cdots\sin(\phi_{(n-2)a})\cos(\phi_{(n-1)a}) \\ \Delta\sin(\phi_{1a})\cdots\sin(\phi_{(n-2)a})\sin(\phi_{(n-1)a}) \end{bmatrix} \tag{4.42}$$

$$\mathbf{u}_{rb} = \begin{bmatrix} \Delta\cos(\phi_{1b}) \\ \Delta\sin(\phi_{1b})\cos(\phi_{2b}) \\ \Delta\sin(\phi_{1b})\sin(\phi_{2b})\cos(\phi_{3b}) \\ \vdots \\ \Delta\sin(\phi_{1b})\cdots\sin(\phi_{(n-2)b})\cos(\phi_{(n-1)b}) \\ \Delta\sin(\phi_{1b})\cdots\sin(\phi_{(n-2)b})\sin(\phi_{(n-1)b}) \end{bmatrix} \tag{4.43}$$

M would be the rotation matrix for *n*- input systems that affects the points on the hyper-sphere. This matrix would be the extension of "Rodrigues" rotation matrix to higher dimensions. The idea for generalization of the algorithm to higher dimensions is that the multi-unit global optimization takes place repeatedly along the circumference of a rotating circle on an *n*-dimensional hyper-sphere. In optimization of a system with *n*-variables the circle of optimization rotates systematically on the *n*-dimensional hyper-sphere in order to cover the entire feasible hyper-spherical surface. In other words, the entire area of the hyper-sphere is going to be swept by these rotations. On the other hand, the angle and axis of such a dynamic continuous rotation must be so adapted that $(\Phi_1, \Phi_2, \ldots, \Phi_{(n-1)})_{mi}$ in each iteration (rotation) correspond to the global optimum of the previous iteration.

Any rotation in *n*-dimensions is described by a rotational angle about an axis. The concept of rotation explained in three-dimensional systems can be analogously extended to a general rotation in *n*-dimensional hyper-spherical space. Generally the motion of an *n*-dimensional point in a circular path is called rotation in an *n*-dimensional hyper-spherical space. When a point rotates about an axis, it traces a circular path as it moves through space. The axis of rotation about which the rotation of point takes place can be a point, line, plane, volume,…or an *n*-dimensional space depending on the dimension of the system of rotation (2,3,4,…, *n* respectively). A good visualization of rotation matrices entries for spaces up to dimension sixteen can be found in http://demonstrations.wolfram.com/RotationMatrixEntries/.

The initial rotating circle made by multi-unit optimization between two units starts from a point which is predefined by the initial position angles $(\Phi_1, \Phi_2, \ldots, \Phi_{(n-1)})_0$ in the polar system. The position angle $\Phi_{(n-1)}$ is periodically adapted based on the multi-unit optimization along the circumference of the circle by adaptation laws between two units (covering the rotation range of ε to 2π Radians). This way a complete rotation about the u_1 axis would be completed. The global optimum found on the circumference of the rotating circle is retrieved at each iteration. Meanwhile the position angle $\Phi_{(n-2)}$ increases monotonically (with a slower dynamic compared to dynamics of $\Phi_{(n-1)}$) and the circle grows until this angle reaches π Radians. At this point the rotating circle begins to shrink once again as the angle $\Phi_{(n-2)}$ approaches 2π Radians. This way a

complete rotation about the u_2 axis would be completed. Having complete rotation about both of the axes u_1 and u_2, a complete rotation about the plane of u_1u_2 would be completed in the range of ε to 2π. In a similar manner the position angles $\Phi_{(n-3)}$, $\Phi_{(n-4)}$,…, Φ_1 monotonically increase (from ε to 2π Radians) with an enough time scale separation between their dynamics thereby completing an entire rotation about their relevant rotation axes u_3, u_4, …, u_1 respectively.

In 3, 4, 5,… , n dimensional space the basic axes of rotation about which the rotation of a point takes place would be the lines of u_1, u_2, u_3 - the planes of u_1u_2, u_1u_3, u_1u_4, u_2u_3, u_2u_4, u_3u_4 - the volumes of $u_1u_2u_3$, $u_1u_2u_4$, $u_1u_2u_5$, $u_1u_3u_4$, $u_1u_3u_5$, $u_1u_4u_5$, $u_2u_3u_4$, $u_2u_3u_5$, $u_2u_4u_5$, $u_3u_4u_5$ and eventually the n-2 dimensional spaces respectively. For n dimensional space the total number of basic rotations about the basic axes is the number of n-2 combinations from n, i.e., C^{n}_{n-2}.

In the n-dimensional space, let U represent the set of all possible rotation axes made by u_1, $u_2 \ldots u_n$. For instance, in the four-dimensional space (n=4) the mentioned set of axes has six members and it would be the set of following planes $U = \{u_1u_2, u_1u_3, u_1u_4, u_2u_3, u_2u_4, u_3u_4\}$. Let U_N represent each member of this set where N is the number of members, e.g., in the four-dimensional space (n=4), N=6 and $U_1 = u_1u_2$, $U_2 = u_1u_3 \ldots$ $U_6 = u_3u_4$. In general case the first member is considered as $U_1 = u_1u_2 \ldots u_{n-2}$.

Having the structure of the rotation matrix in n-dimensional space, the axis and angle of rotation of any point corresponding to units "a" and "b" can be obtained in a similar manner for three dimensions. This way an entire rotation about all axes of rotation made by all variables u_1, $u_2 \ldots u_n$ must be established. These rotations would be about the set of axes of lines, planes, volumes and etc. made by variables u_1, $u_2 \ldots u_n$ depending on the dimension of the hyper-spherical space (n). Therefore, each of these rotations takes place about the axes of rotations in U which was defined earlier. The rotations are performed in order to cover the entire n-dimensional hyper-sphere. However, the extra rotations about different axes in higher dimensions affect the dynamics of the two units on the circle of the basic algorithm (in two dimensions) which was established in chapter 3. Therefore, the compensation of the extra rotations about the axes in higher dimensions is necessary to latch on the global optimum found on the circular path.

At t=0 consider the initial rotation in which the axis of rotation lies on U_1. Beginning the multi-unit optimization at the north pole of the hyper-sphere centered at ($u_1, u_2, \ldots, u_n$), a point very close to the north pole would be the starting point of optimization $(\Phi_1, \Phi_2, \ldots, \Phi_{(n-1)})_{m0} = (\varepsilon, \varepsilon \ldots, \varepsilon)$. Any specific rotation realized by the rotation matrix **M** would be on a circular slice on the n dimensional sphere. The multi-unit optimization along the circumference of this rotating circle inside the hyper-sphere is repeated every T time units. The position angle Φ_{n-1} is periodically adapted based on the multi-unit adaptation laws along the circumference of the circle by two units (covering the rotation about the axis U_1 in the range of ε to 2π Radians). In other words, Φ_{n-1} plays the role of angle θ in chapter 4. Meanwhile the rotating circle is moved from the north to the south pole by simultaneous increasing the other position angles $\Phi_1, \Phi_2, \ldots, \Phi_{n-2}$ from ε to 2π Radians (with an enough time scale separation between their dynamics).

The initial rotating circle on the hyper-sphere made by multiplication of the combinatorial rotational matrix M on this n-dimensional point would be very small near the north pole if the value of the initial rotation angles $(\Phi_1, \Phi_2, \ldots, \Phi_{(n-1)})_0$ are chosen small $(\varepsilon, \varepsilon, \ldots, \varepsilon)$. In other words, at t=0 the n-input systems "a" and "b" start to rotate about U_1-axis moving in a circular path on a hyper-sphere with a very small radius. As the position angles $\Phi_1, \Phi_2, \ldots, \Phi_{n-2}$ accordingly become larger, the circle moves southward and the radius of the rotating circle becomes larger. The dynamics of the radius of the rotating circle on the hyper-sphere depends on the dynamics of Δ, the dynamics of position angles $\Phi_1, \Phi_2, \ldots, \Phi_{n-1}$. The rotation angle γ compensate the effect of the dynamics of position angles $\Phi_1, \Phi_2, \ldots, \Phi_{n-1}$ in order to latch the circumference of the rotating circle on the global optimum found in each iteration. This is realized through rotation matrix **M**.

If the centre of the hyper-sphere is so adapted as to keep the best optimum at the circumference of the rotating circle, the starting point of each rotation on the hyper-sphere is specified by the global optimum found on the circle of previous rotation. Having $\Phi_1, \Phi_2, \ldots, \Phi_{n-2}$ increased at the same time with dynamics of Φ_{n-1} but with a slower hierarchical speed, the rotating circles continue to enlarge until the circle whose circumference is equal to the equator of the hyper-sphere is reached. From now on, the radii of the rotating circles become smaller. Once the radius

of the circle completely shrinks to zero, the entire surface of the sphere is covered by these rotating circles. Briefly, if these rotations are repeated with enough time scale separation between the dynamics of the angles $\Phi_{(n-1)}$, $\Phi_{(n-2)}$,…, Φ_1, the set of expanding and contracting rotating circles would cover the entire surface of the n-dimensional hyper-sphere. These angles must increase with different predefined speeds at the same time. This is consistent since an n-dimensional axis of rotation offers enough space (degrees of freedom) for rotation with different speeds in different dimensions. $(\Phi_1, \Phi_2, \ldots, \Phi_{(n-1)})_{mi}$ would represent the global optimum along the circumference of the circle in iteration "i-1" if $\varepsilon = 0$. Each iteration of the algorithm is a complete circular rotation on the hyper-sphere. The converging point where the last rotating circle on the hyper-sphere ends up corresponding to a particular latitude and longitude which is identified as the global optimum found on the surface of the hyper-sphere.

The time scale separation between the dynamics of the position angles Φ_1, Φ_2, …, Φ_{n-1} has the main role in convergence of the algorithm to the n-dimensional global optimum. The difference between the values of the adaptation gains for these angles must be chosen large enough as follows,

$$k_{\phi 1} \gg k_{\phi 2} \gg \ldots \gg k_{\phi(n-1)}$$

The time scale separation between the different layers of this method makes the convergence of the optimization algorithm slow when the number of variables increases.

4.6 Conclusion

A new method for solving unconstrained global optimization problems with three variables is proposed in this chapter. This technique was the extension of the global optimization of two-input systems with multi-units to three dimensions. The proposed multi-unit method was found to be effective in terms of accurate convergence to the global optimum and computationally efficient. The computational comparisons with three global optimization techniques in this class (DIRECT, genetic algorithm and simulated annealing) illustrate the competitive performance of the proposed approach.

A discussion on an extension of the proposed technique to a higher number of variables was presented. The other objective of such an extension would be to develop this technique to constrained optimization problems. Comparison results for multi-unit optimization in two- and three-input systems show that when the number of variables is small, then the multi-unit global optimization method is acceptable, sometimes superior to the other competitive methods. However, as the number of variables increase, this method may rapidly become inefficient. This situation arises from the fact that many of optimization iterations are made repeatedly and systematically on the limited subspaces of the slowly shrinking circles. This might prevent the algorithm to coincide with the global optimum at earlier iterations depending on the initial input values. Thereby, multi-unit algorithm cannot jump to the global optimum (stochastically or deterministically) as the other competitive methods. This causes only restricted systematic progress to the global minimum. Although this can be considered as a drawback of this method, however the convergence to global optimum in this technique is guaranteed. One way to improve this drawback would be usage several units (extra floating units) in multi-unit optimization framework instead of just to two of them.

CONCLUSIONS AND RECOMENDATIONS

Conclusions

The first contribution of this thesis is the extension of the local-based extremum seeking control using multi-units to global optimization of the static nonlinear and continuous scalar systems. Thus, in Chapter 2, we first studied the core idea of such an extension and the proof of convergence to the global optimum of the static curves having two identical units. According to the hypothesis when the non-convex curvatures of the static units are the same, it has been demonstrated that the convergence of the algorithm can be ensured by a choice of varying disturbance introduced between the operating points of each unit (Δ). The major breakthrough came when it was realized that the global optimum can be reached arbitrarily close by relatively low computational cost by monotonically decreasing the offset to zero. The algorithm deterministically converges to the small vicinity of the global optimum by the proper choice of this offset. Choosing an initial offset (Δ_0) sufficiently small, minimizes the number of function evaluations and ends up with a faster convergence. On the other hand, if the value of the initial disturbance is chosen too small, the system converges to an equilibrium having a distance with the real global optimum. However, it is guaranteed that the obtained equilibrium is always the global optimum in the operating interval which is imposed by Δ_0 to the multi-units. It has been shown that depending on the "Lipschitzian" characteristic of the static maps, the adaptation laws may introduce stiffness in the integration process. Modification to the original design of the algorithm has been introduced to relax this effect. An alternative solution to overcome this stiffness was to replace the "Sign" function by "Tangent Hyperbolic" in the main adaptation law. It has been demonstrated that the number of function evaluations and the convergence speed of the system depend on the relaxation of the adaptation gain for the integration of operating points. Moreover, the simultaneous online adaptation of multi-units towards the global optimum requires no interruption.

Another advantage of this algorithm is that it does not require the assumption of differentiability throughout this work. Example 2.3.4 presented in Chapter 2 introduces a case of a non-differentiable static characteristic at the global optimum.

The proof of convergence for this algorithm has been provided using mathematical contradiction formalism. The simulation results confirmed the theoretical developments on several test benchmarks for global optimization. The efficiency of the method and its required number of function evaluations were compared to the Genetic algorithm and Simulated annealing methods. Moreover, a switching control law has been incorporated to handle the constrained optimization problems. The proof of convergence of this method using this logic has also been established.

Although this method is not a real time extremum seeker in terms of following the variable optimum, it is a black-box global optimization strategy since it uses an extremum-seeking method as a tool. The algorithm uses a recursive procedure as a real-time optimizer in order to converge to the optimum. However, this recursion stops after the convergence to the global optimum. This is because the offset (Δ) between the multi-unit inputs converge to zero at the end of the optimization. In order to keep the process on this point or follow the variable global optimum, some other control strategies must be deliberated.

The development of the global optimization algorithm along the circumference of a shrinking circle in Chapter 3 is another contribution of this dissertation. The required time separation between the dynamics of the iterative adaptation towards the global optimum on the shrinking circle in one hand and the dynamics of the contraction of the same circle in another hand was a necessity in this method. This fact was demonstrated trough a typical proof by contradiction in the limit case.

In Chapter 4, the global optimization algorithm using multi-units has been extended to the three variable systems. Generalization of the algorithm to higher dimension has been discussed. The results showed that the scalability of the method is the main challenge in multivariable case. The

impact of this drawback becomes more outstanding by increasing the size of the problem in terms of degrees of freedom. This could reduce the effectiveness of the algorithm. Some modifications have been introduced to make the algorithm more scalable. Considering an additional floating unit in the center of the shrinking circle can significantly improve the scalability of the algorithm. The adaptive rotating control of the circle with varying radius on a shrinking spherical space was the main contribution of Chapter 4. The rotating variant circle covers the feasible region where the multi-unit optimization takes place. This achievement has been developed based on the concepts of rotating circle on a plane with reducing radius in Chapter 3.

Recommendations

As a result of fact, this dissertation opens a new area of research and there are many interesting things to investigate concerning global optimization by multiple units. Based on the primary results obtained in this dissertation, some principal ideas on the preliminary structure of global optimization using multi-units have been established. With the work done on the global optimization technique by multi-units so far, the future research can now focus on the other progresses that can be discovered with this method. The following suggested areas for investigation are related to the results of the chapters presented in this thesis including global optimization of constrained and unconstrained scalar and multivariable static systems using multi-unit adaptation. The future potential experimental application of this method to optimize some engineering and industrial problems is also discussed. The main aspects that could help to complete this work are as follows,

1- The computational load of the proposed algorithm in this dissertation increases significantly when the dimension of the optimization problem is increased. Intuitively, it seems to be logical to use more units to solve the optimization problem, but the challenging question is how many units would be more efficient and how they would be controlled by the offset between their inputs.

One way to modify the growing number of function evaluations would be increasing the number of multi-units according to the dimension of problem. This way the efficiency of the global optimization algorithm will be significantly improved. For instance an extra unit could be placed on the center of the contracted circle. If so, the third unit is going to be systematically searching for the global optimum inside the area of the circle and be adapted with the other units on the circumference. The algorithm coincides the global optimum comparatively fast thereby relaxing the impact of number of function evaluations. However, regarding the simplicity of the algorithm the number of units is kept two in this thesis.

As a result, scalability and extension of the proposed algorithm to higher dimensions (more than three variables) is the first question come in mind. Only two units have been considered in the structure of the algorithm so far. Henceforth, having increased one variable in the multivariable system, an extra unit can be added to the nth center of the n-dimensional hyper-spherical space established by multiple units. This simple modification allows the algorithm to estimate the gradient more efficiently and made it scalable. It can increase the speed of convergence and reduce the number of function evaluations significantly. It also gives a simple rule of thumb in order to calculate the number of required units in multivariable optimization using multiple units.

2- The extension of global optimization algorithm in this thesis is limited to static systems. It would be interesting to consider the more general case, where the system is dynamic. In this case, since the units are identical, the estimation of the gradient by finite differences eliminates the effect of dynamics on this gradient (Woodward, 2009). In multivariable systems, the complexity of analysis is increased with increasing the number of manipulated variables.

3- When there is only one exclusive unit and an appropriate available model, it would be interesting to analyze the efficiency of the global multi-unit optimization when the other units are replaced by adapted virtual models. In this framework, it is proposed that a system identification algorithm firstly estimate the different parameters of the model. Once the optimization algorithm is engaged, the excitation signal of the identification method will be cut to half. The performance of the method when it is coupled with the parameter estimation and system identification

algorithms should be compared. The temporal excitation signal used in system identification process may affect the convergence velocity of the algorithm.

4- The analysis presented takes no account of measurement noise. All of the algorithms in this dissertation have been simulated in the ideal environment without any noise or disturbances. In the ideal conditions, each measurement represents the exact value of the process. However, any kind of real process is affected by high or low-frequency drifts and measurement noises. The impact of measurement noise on the performance of the method deserves attention. Presence of noise in system can make a real challenge in the gradient estimation. Preliminary studies in local multi-unit extremum seekers have shown that the measurement noise causes an error estimation of the gradient. This error can be eliminated by choosing value of Δ greater than the standard deviation of the noise (Woodward, 2009). However, too high a value of this parameter in the local extremum seeking by multi-units means a loss of performance. This solution would not be practical in global optimization using multi-units since the value of offset (Δ) asymptotically decreases to zero as the optimization process proceeds. As a result, it would be essential to investigate the effect of noise on the presented optimization algorithms through stochastic methods. This study would characterize the error of the gradient estimation by this algorithm in presence of noise and would propose some solutions to improve the performance.

The impact of noise on the performance of the algorithm would be worse in the case of optimization with multiple virtual models (which is proposed in 3), because the identification of the parameters in noisy conditions takes more time. The measurement noise slows down the convergence to the optimality.

5- The main assumption in the presented algorithm was having the identical systems. For example, the process may contain several identical micro-reactors or several photovoltaic cells. This means that it was assumed that there is no difference between the operating curves of their static maps. This is a strong assumption. It means that, in practice, the assumption that the units are identical process is rarely encountered which greatly limits its application. However, in case that the curves of static maps are not identical, some correctors would be needed to compensate for these differences. These correctors have been developed for local optimization using non-

identical multi-units by Woodward et al. (2009). An appropriate study in order to extend the similar correctors for global optimization of scalar systems using non-identical units would be another possible extension of this work. The analysis can be repeated for multivariable systems but the complexity of analysis increases with number of manipulated variables.

6- It has been shown that in local optimization of dynamic systems using multi-units, the differences between the dynamics of units affects the stability of the algorithm (Reney, 2008). The stability of global optimization algorithms for dynamic systems with multi-units that represent differences in their dynamics is a subject that can also be followed in this framework. The next step would be the development of an appropriate correction strategy which ensures the stability of this method.

7- The algorithm was extended to the constraint optimization problem where a switching adaptation law was used to handle the constraints. However, it was seen that the proposed algorithm can chatter when the solution is on the boundary of the feasible region. This means that the gradient is not equal to zero at the optimum. The non-zero gradient then tries to seek an operating point with a higher cost by pushing the system into the infeasible region. However, the feasibility part of the algorithm pushes the system back into the feasible region. Thus, the solution chatters around the optimal point. The frequency of chattering depends on the final value of Δ which is determined by ε. Gradient projection methods (e.g., Woodward et al., 2007) can be used to alleviate this chattering.

8- Extension of the constrained global optimization method to the multivariable problems forms another future step of this research work. On the other hand, the constrained global optimization algorithm has been presented for identical scalar units. The application of the method for similar but non-identical units deserves to be studied.

9- In final summary, it appears that a set of experimental set up can be developed in practice based on this study in order to confirm the predicted results by the theory and simulations. The

optimization of producing electricity using “microbial fuel cells” (i.e., use of waste water as an energy source and bacteria as the catalyst) which has been experimentally verified by the local multi-unit optimization method (Woodward et al., 2009) seems to be a nice fit to be experienced by the global optimization method. Another alternative would be the set up which can implement other types of multiple parallel bio-reactors.

10- This research is an appropriate base for developing the global optimization tools by multiple units in some engineering or industrial applications. The presented global optimization theory has been developed so far only to continuous processes. Another long-term research program would be the extension of the technique to the domain of “batch processes” (Srinivasan). As a matter of fact, “batch processes” could be a proper candidate for the global optimization by multiple units. If the presented algorithm is well developed and implemented for dynamic systems, it can significantly reduce the operating costs of these processes. In this framework, the number of pertinent set-points, the corresponding degrees of freedom, the influent manipulated variables and all involved constraints in optimization by multiple batch units would be analyzed. Different trajectories such as convergence to the optimal operating points must be compared in order to find the best compromise between robustness and performance.

11- It is proposed to create an auxiliary “parallel-processing” algorithm in order to reduce the integration time and to decrease the computation impact. The established gradient estimation strategy using this algorithm is such that some simultaneous computational load must be executed by several identical and parallel units. It would be interesting to extend an algorithm such that the integration of each unit is able to share data with others. This can be achieved through an appropriate multi-processing algorithm on multiple processors. This algorithm can significantly reduce the number of function evaluations by eliminating the extra calculations.

BIBLIOGRAPHY

Ackley, D.H. (1987). An empirical study of bit vector function optimization. *Genetic Algorithms and Simulated Annealing*, 170–204.

Ariyur,K.B., & Kristic,M.(2003). *Real-time Optimization by Extremum-Seeking Control.* New York: John Wiley.

Archetti, F., Schoen, F. (1984). A Survey on the Global Optimization Problem: General Theory and Computational Approaches. *Annals of Operations Research*, **1**(1), 87-110.

Baker, M.J. (1998). URL;http://www.euclideanspace.com/maths//geometry/rotations/conversions (last access on September 2010)

Baritompa, W.P., Hendrix, E.M.T. (2005). On the Investigation of Stochastic Global Optimization Algorithms, *Journal of Global Optimization*, **31**, 567-578.

Bartholomew-Biggs, M.C., Parkhurst, S.C., Wilson, S.P. (2003). Global optimization approaches to an aircraft routing problem. *European Journal of Operational Research*, **146**(2), 417-431.

Betrò, B. (1983). A Bayesian Nonparametric Approach to Global Optimization. *Methods of Operation Research* **45**, 47-59.

Björkman, M., Holmström, K. (1999). Global Optimization Using the DIRECT Algorithm in MATLAB. *Advanced Modeling and Optimization*, **1**(2), 17-37

DeHaan, D., & Guay, M. (2005). Extremum Seeking Control of State-Constrained Nonlinear Systems, *Automatica,* **41**(9)**,** 1567-1574.

De Jong, K.A. (1975). An analysis of the behavior of a class of genetic adaptive systems. *Dissertation Abstracts International*, **36**(10), 5140B (University Michigan No. 76-9381).

Deng, G., Ferris, M.C. (2007). Extension of the DIRECT Optimization Algorithm for Noisy Functions. *Proceedings - Winter Simulation Conference*, 497-504.

Dixon, L.C.W., Szegö, G.P., (1978). The Global Optimization Problem: An Introduction. *Towards Global optimization II*, Amsterdam: North-Holland, 1-15.

Esmaeilzadeh Azar, F., Perrier, M., & Srinivasan, B. (2010). A Global Optimization Method Based on Multi-unit Extremum-seeking for Scalar Nonlinear systems. *Computers and Chemical Engineering*, Elsevier, doi:10.1016/j.compchemeng.2010.04.003.

Esmaeilzadeh Azar, F., Perrier, M., & Srinivasan, B. (2009). Real-Time Global Optimization using Multiple Units. In: *ICONS 2009, The 2nd IFAC International Conference on Intelligent Control Systems and Signal Processing.* Istanbul, Turkey.

Esmaeilzadeh Azar, F., Perrier, M., & Srinivasan, B. (2009). Real-Time Global optimization of Constrained Nonlinear Systems using Multiple Unit Optimization Scheme. In: *8th World Congress of Chemical Engineering.* Montreal, Canada.

Esmaeilzadeh Azar, F., Perrier, M., & Srinivasan, B. (2010). Global Optimization of Tow-input Systems using Multiple-Unit Adaptation. *DYCOPS 2010, The 9th International Symposium on Dynamics and Control of Process Systems,* Leuven, Belgium, 743-748.

Ferreira, A.G., Žerovnik, J (1993). Bounding the Probability of Success of Stochastic Methods for Global Optimization. *Computers & Mathematics with Applications* **25**, Issues 10-11, 1-18

Finkel, D.E. (2005). Global Optimization with the DIRECT Algorithm. *Ph.D. Thesis*, North Carolina State University, USA.

Floudas, C.A., & Gounaris, C.E. (2008) A Review of Recent Advances in Global Optimization. *Journal of Global Optimization.* doi: 10.1007/s10898-008-9332-8.

Fogel, L.J., Owens, A.J., Walsh, M.J. (1996). Artificial Intelligence trough Simulated Evolution. New-York: John Wiley.

Gablonsky, J.M. (2001). Modifications of the DIRECT Algorithm. *Ph.D. Thesis*, North Carolina State University, USA

Glover, F., (1989). Tabu Search. *ORSA Journal of Computing*, **1**(3), 190-206.

Goldberg, D.E. (1989). Genetic Algorithms in Search, Optimization, and machin Learning. New-York: Addison-Wesley.

Goldberg, D.E. (1990). A Note on Boltzmann Tournament Selection for Genetic Algorithms and Population-Oriented Simulated Annealing. *Complex Systems*, **4**(4), 445-460.

Guay, M., Dochain, D., & Perrier, M. (2004) Adaptive Extremum Seeking Control of Continuous Stirred Tank Bioreactors with Unknown Growth Kinetics. *Automatica,* **40**, 881-888.

Holland, J.H. (1973). Genetic algorithms and the optimial Allocation of Trials. *SIAM Journal on Computing*, **2**(2), 88-105.

Hart, W.E. (1994). Adaptive Global Optimization with Local Search. *Ph.D. Thesis*, Departement of Computer Science and Engineering, University of California, USA.

Jones, D.R., Perttunen, C.D., Stuckman, B.E. (1993). Lipschitzian Optimization Without the Lipschitz Constant, *Journal of Optimization Theory and Applications*,**79**(1), 157-181.

Jones, D.R., Perttunen, C.D., Stuckman, B.E. (1992). Global optimization: beyond the Lipschitzian model. *IEEE International Conference on Systems, Man and Cybernetics* (Cat. No.92CH3176-5), **1**, 566-70.

Khalil, H.K. (2002). *Nonlinear Systems*. (3rd edition). Upper Saddle River, NJ: Prentice Hall. ISBN 0-13-067389-7.

Kargupta, H., Goldberg, D.E. (1997). SEARCH, Blackbox optimization and Sample Complexity. *Foundation of genetic Algorithms 4 and LAUR-96-80.*

Kirpatrick, S., Gelatt, C.D., Vecchi, M.P. (1983). Optimization by Simulated Annealing. *Science*, **220**(4598), 671-680.

Krstic, M., Wang, H.H. (2000). Stability of Extremum Seeking Feedback for General Nonlinear Dynamic Systems, *Automatica*, **36**(4), 595-601.

Krstic, M. (2000). Performance Improvement and Limitations in Extremum Seeking Control, *System and Control Letters*, **39**, 313-326.

Kartik, Ariyur, K.B., & Krstic, M. (2002). Multivariable Extremum Seeking Feedback: Analysis and Design, *15th International Symposium on Mathematical Theory of Networks and Systems*, University of Notre Dame, South Bend, Indiana, USA.

Kohonen, J. (1999). A Brief Comparison of Simulated Annealing and Genetic Algorithm Approaches. *Term paper for the"Three Concepts:Utility" course*, Department of Computer Science, University of Helsinki URL; http://www.cs.helsinki.fi/u/kohonen/papers/gasa.html (last access on September 2010)

Laguna, M., Molina, J., Pérez, F., Caballero, R., Hernández-Díaz, A.G. (2010). The challenge of optimizing black boxes: A scatter search/rough ser theory approach. *Journal of the Operational Research Society*, **61**(1), 53-67.

Lacks, D.J. (2003) Real-Time Optimization in Nonlinear Chemical Processes: Need for Global Optimization. *AICHE Journal,* **49**(11).

Leblanc, M. (1922) Sur l'électrification des Chemins de Fer au Moyen de Courants Alternatifs de Fréquence élevée. *Revue Générale de l'électricité.* **12**(8), 275-277.

Levy, A., Montalvo, A. (1985). The Tunneling Algorithm for the Global Minimization of Functions, *SIAM Journal of Scientific and Statistical Computing* **6**, 15–29.

Mann, J.W., Smith G.D. (1996). A Comparison of Heuristics for Telecommunications Traffic Routing. In *Modern Heuristic Search Methods*, John Wiley & Sons Ltd. 235-254.

Mahfoud, S.W., Goldberg, D.E. (1992). A Genetic Algorithm for Paraller Simulated Annealing. In manner, R., Manderick, B. (Eds.), *Parallel Problem Solving from Nature* (301-310). Amsterdam: North-Holland.

Manikas, T.W., Cain, J.T. (1996). Genetic Algorithms vs Simulated Annealing:A Comparison of Approaches for Solving the Circuit Partitioning Problem. *Technical Report* 96-101 ,The University of Pittsburgh.

Marlin, T.E., Hrymak, A.N. (1997). Real-Time Optimization of Continuous Processes. volume 316 of *AIChE Symposium Series*, page 156.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E. (1953) Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, **21**, 1087-1092.

Marlin, T.E., Hrymak, A.N. (1997). Real-Time Operations Optimization of Continuous Processes, In: *Fifth international conference on chemical process control (CPC-5, Lake Tahoe, Jan. 1996)*, volume 93 of AIChE Symposium Series, Lake Tahoe, Nevada, 156–164.

Popović, D. (2004). Topics in Extremum Seeking , *Ph.D. Thesis*, University of California, USA.

Propović, D., Jancović, M., Manger, S. & Teel, A.R. (2003). Extremum Seeking Methods for Optimization of Variable Cam Timing Engine Operation. In: *American Control Conference,* Denver, Colorado, 3136-3141.

Ram, D.J., Sreenivas, T.H., Subramaniam, K.G. (1996). Parallel simulated annealing algorithms. *Journal of Parallel and Distributed Computing,* **37**(2), 207-12, 15

Rotea, M.A. (2000). Analysis of Multivariable Extremum Seeking Algorithms, Proceedings of *the American Control Conference*, **1**, 433-437.

Rechenberg, I. (1973). Bionik, evolution und optimierung. Naturwissenschaftliche Rundschau, **26**, 465-472. See: http://www.bionik.tu-berlin.de/institut/xs2evost.html (last access on September 2010)

Reney, F. (2008). Etude de La Technique d'Optimisation Multi-Unités avec Dynamiques Différentes. *Master's Thesis*, Ecole Polytechnique de Montreal, Montreal, Canada.

Rinnooy kan, A.H.G., Timmer, G.T. (1987) Stochastic Global Optimization Methods. Part I: Clustering Methods. Mathematical Programming, **39**(1), 27-56.

Rodrigues, O. (1816). Mémoire sur l'attraction des Spheroides*, Correspondence sur l'École Polytechnique, **3**, 361–385.*

Rosenbrock, H.H. (1960). An Automatic Method for Finding the Greatest or Least Value of a Function, *The Computer Journal* **3**, 175–184, doi:10.1093/comjnl/3.3.175.

Saber, N., Shaw, J.M. (2008). Rapid and Robust Phase Behavior Stability Analysis Using Global Optimization . *Fluid Phase Equilibria*, **264**(1-2), 137-146.

Srinivasan, B., Bonvin, D., Visser, E., & Palanki, S. (2003). Dynamic Optimization of Batch Processes: II Role of Measurements in Handling Uncertainty. *Computer and Chemical Engineering*, **44**, 27-44.

Srinivasan, B. (2007). Real-time Optimization of Dynamic Systems using Multiple Units. *International Journal of Robust and Nonlinear Control.* **17**, 1183–1193.

Strongin, R.G., Sergeyev, Y.D. (1992). Global multidimensional optimization on parallel computer. *Parallel Computing*, **18**(11), 1259-1273.

Schneider, J.J., & Kirkpatrick, S. (2006) *Stochastic Optimization.* Springer: Germany.

Schoen, F. (1991). Stochastic Techniques for Global Optimization: A Survey of Recent Advances. *Journal of Global Optimization*, **1**, 207-228.

Shubert, B.O. (1972). A Sequentional Method Seeking The Global maximu of a Function. SIAM *Journal on Numerical Analysis*, **9**(3), 379-388,

Tan, Y., Nešić, D., & Mareels, I.M.Y. (2005). On Non-Local Stability Properties of Extremum Seeking Control. In: *16th IFAC World Congress,* Prague, Czech Republic, 2483-2489.

Tan, Y., Nešić, D., & Mareels, I.M.Y., & Astolfi, A. (2006 a). On Global Extremum Seeking in the Presence of Local Extrema. In: *45th IEEE Conference on Decision and Control,* San Diego, CA, USA.

Tan, Y., Nešić, D., & Mareels, I.M.Y. (2006 b). On Non-Local Stability Properties of Extremum Seeking Control. *Automatica,* **42**(6), 889-903.

Tan, Y., Nešić, D., & Mareels, I.M.Y. (2006 c). On the Choice of Dither in Extremum Seeking Systems. In: *Proceedings of IEEE Conference on Decision and Control*, 2789-2794.

Teel, A.R., Popović, D.I. (2001). Solving Smooth & Non-Smooth Extremum Seeking Problems by Methods of Nonlinear Programming. In : *Proceedings of the American Control Conference.*

Törn, A., Žilinskas, A. (1989). Global optimization. Berlin: Springer-Verlag.

The Mathworks, *Optimization Toolbox 4.0.*, Matlab version 7.6.0.347(R2009a). The URL address; http://www.mathworks.com/

Vavasis, S.A., (1991). Nonlinear Optimization: Complexity Issues. New York: Oxford university Press.

Vanderbit, D., Louie, S.G. (1984). A Monte Carlo Simulated Annealing Approach to Optimization over Continuous Variables. *Journal of Computational Physics* **56**(2), 259-271.

Walsh, G.C. (2000). On the Application of Multi-Parameter Extremum Seeking Control, *Proc. of the American Control Conference,* Chicago, IL, USA, 411-415.

Woodward, L. (2009). Adaptation de la Méthode Multi-Unités à l'Optimisation sous Contraintes en Présence d'Unités Non Identiques, *Ph.D. Thesis* , Ecole Polytechnique de Montreal, Montreal, Canada.

Woodward, L., Perrier, M., & Srinivasan, B. (2007). Multi-Unit Optimization with Gradient Projection on Active Constraints. In: *8th International Symposium on Dynamics and Control of Process Systems, Preprints DYCOPS 2007,* **1**, 129-134.

Woodward, L., Perrier, M., & Srinivasan, B. (2009). Improved Performance in the Multi-Unit Optimization Method with Non-Identical Units. *Journal of Process Control.* **19**(2), 205-215.

Zhu, H., Bogy, D.B. (2002). Direct algorithm and its application to slider air bearing surface optimization. *Intermag Europe 2002 Digest of Technical Papers. 2002 IEEE International Magnetics Conference (Cat.No.02CH37323)*, pp DP9.

Zhang, C. (2006). Numerical Optimization and Perturbation Based Extremum Seeking Control and Applications, *Ph.D. Thesis*, University of Dayton, USA.

Zitzler, E., (2003). Bio-Inspired Optimization and Design. *Computer Engineering and Networks Laboratory (Lecture)*. Swiss Federal Institute of Technology, Zurich.

APPENDIX I

RODRIGUES' ROTATION MATRIX

The rotation matrix in three dimensional space is called "Rodrigues" rotation formula (Rodrigues, 1816) and there are different ways to obtain it. In the following section, the proof from (Baker, 1998) has been used as reference. However, figures and notations have been modified according to formulas in chapter 4. The rotation of the point $\mathbf{V}_1$ about the axis $\mathbf{\Gamma}$ is desired. These vectors are depicted by the blue and red lines in figure I.1. The track of the rotating point forms a circular path in a given plane (shown in green). In the diagram below, this plane has been moved down so that it passes through the origin. This plane is defined by the rotation axis (the plane is perpendicular to the axis).

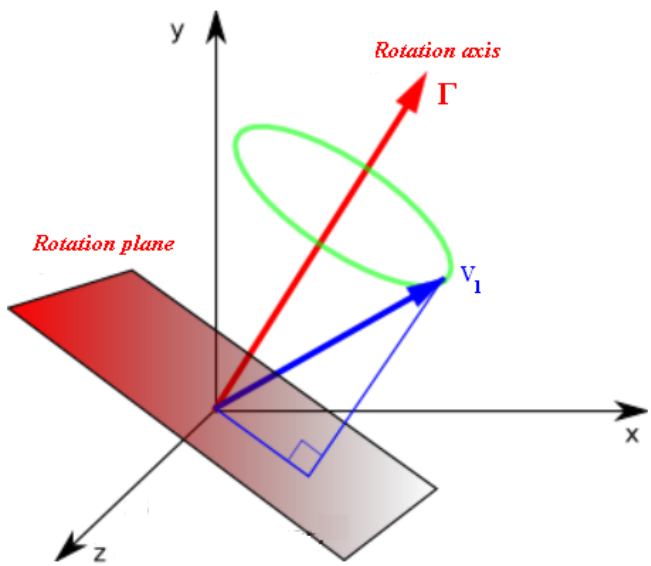

Figure XI Rotation of vector $\mathbf{v}_1$ about axis $\mathbf{\Gamma}$

Now, we generate two basis vectors in the mentioned plane which can be used for defining the circle in the coordinate system. In order to get one of these basis vectors, we cross multiply the axis with $\mathbf{V}_1$. This gives a vector on the plane ($\mathbf{b}_2$) which is perpendicular to the projection of $\mathbf{V}_1$

onto the plane ($\mathbf{b}_1$). If (×) denotes the vector cross product, the following equation holds (figure I.2),

$$\mathbf{b}_2 = \mathbf{\Gamma} \times \mathbf{v}_1 \tag{AI.1}$$

where,

$\mathbf{V}_1$ = point being rotated

$\mathbf{\Gamma}$ = *axis* = direction of rotation as normalized (unit length) vector

$\mathbf{b}_1$ = $basis_1$ = first basis vector

$\mathbf{b}_2$ = $basis_2$ = second basis vector

The first basis vector $\mathbf{b}_1$ is just the projection of the vector $\mathbf{V}_1$ onto the plane. This is a vector perpendicular to axis and $\mathbf{b}_2$, namely,

$$\mathbf{b}_1 = \mathbf{b}_2 \times \mathbf{\Gamma} \tag{AI.2}$$

$$\mathbf{b}_1 = (\mathbf{\Gamma} \times \mathbf{v}_1) \times \mathbf{\Gamma} \tag{AI.3}$$

These basis vectors are shown in the following figure,

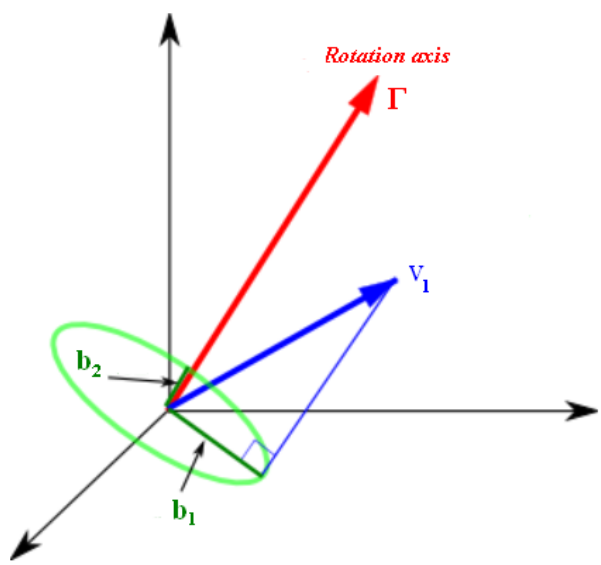

Figure XII Illustration of the basis vectors $\mathbf{b}_1$ and $\mathbf{b}_2$

Therefore, the circle is given by a linear product of the basis vectors as follows (figure I.3),

$$\cos(\gamma)b_1 + \sin(\gamma)b_2 \tag{AI.4}$$

where,

γ = *angle* = angle of rotation

Since the circle has been offset to the origin, this is not quite the circle required. Therefore, we need to move it back to its original place as follows,

$$\mathbf{v}_2 = offset + \cos(\gamma)b_1 + \sin(\gamma)b_2 \tag{AI.5}$$

where,

$\mathbf{V}_2$ = transform of point $\mathbf{V}_1$

offset = distance of centre of circle from origin

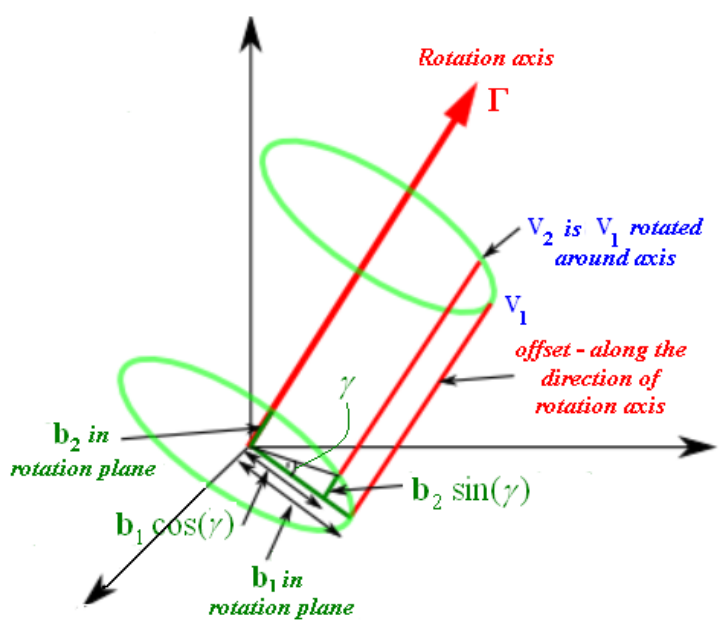

Figure XIII Illustration of the angle of rotation γ about the axis of rotation $\mathbf{\Gamma}$

From the diagram we can see that:

$$offset = \mathbf{v}_1 - b_1 \tag{AI.6}$$

Therefore substituting this value gives:

$$\mathbf{v}_2 = \mathbf{v}_1 - b_1 + \cos(\gamma)b_1 + \sin(\gamma)b_2 \tag{AI.7}$$

$$\mathbf{v}_2 = \mathbf{v}_1 + (\cos(\gamma) - 1)b_1 + \sin(\gamma)b_2 \tag{AI.8}$$

Substituting the basis values above gives:

$$\mathbf{v}_2 = \mathbf{v}_1 + (\cos(\gamma) - 1)((\mathbf{\Gamma} \times \mathbf{v}_1) \times \mathbf{\Gamma}) + \sin(\gamma)(\mathbf{\Gamma} \times \mathbf{v}_1) \tag{AI.9}$$

This equation includes vector algebra. However, it is desired to convert this formula into matrix algebra. This can be done by using the skew symmetric matrix; Cross production is only applied to three dimensional vectors and it represents a vector which is perpendicular to both of the vectors being multiplied. We want to find a 3×3 matrix which is equivalent to vector cross production. We know that,

$$if \quad \mathbf{V} = \mathbf{A} \times \mathbf{B} \quad then \quad \mathbf{V} = \tilde{\mathbf{A}}\mathbf{B} \tag{AI.10}$$

where $\tilde{\mathbf{A}}$ is a square skew-symmetric (anti-symmetric) matrix as follows,

$$\tilde{\mathbf{A}} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{AI.11}$$

Therefore, converting $\mathbf{V}_2$ to the matrix form gives:

$$\mathbf{v}_2 = \mathbf{v}_1 + (\cos(\gamma) - 1)(\tilde{\mathbf{\Gamma}}\mathbf{v}_1 \times \mathbf{\Gamma}) + \sin(\gamma)\tilde{\mathbf{\Gamma}}\mathbf{v}_1 \tag{AI.12}$$

There is still one vector cross product in this formula. In order to remove it we will first change the order by using the anti-commute law:

$$\mathbf{v}_2 = \mathbf{v}_1 + (1 - \cos(\gamma))(\mathbf{\Gamma} \times \tilde{\mathbf{\Gamma}}\mathbf{v}_1) + \sin(\gamma)\tilde{\mathbf{\Gamma}}\mathbf{v}_1 \tag{AI.13}$$

Now, we can substitute the skew symmetric $\tilde{\mathbf{\Gamma}}$ as before:

$$\mathbf{v}_2 = \mathbf{v}_1 + (1-\cos(\gamma))(\tilde{\mathbf{\Gamma}}^2\mathbf{v}_1) + \sin(\gamma)\tilde{\mathbf{\Gamma}}\mathbf{v}_1 \tag{AI.14}$$

Gathering the $\mathbf{V}_1$ terms together gives:

$$\mathbf{v}_2 = [\mathbf{I} + (1-\cos(\gamma))\tilde{\mathbf{\Gamma}}^2 + \sin(\gamma)\tilde{\mathbf{\Gamma}}]\mathbf{v}_1 \tag{AI.15}$$

Let the rotation matrix be **M** where:

$$\mathbf{v}_2 = \mathbf{M}\mathbf{v}_1 \tag{AI.16}$$

Therefore, the rotation matrix is:

$$\mathbf{M} = \mathbf{I} + (1-\cos(\gamma))\tilde{\mathbf{\Gamma}}^2 + \sin(\gamma)\tilde{\mathbf{\Gamma}} \tag{AI.17}$$

where,

M = rotation matrix

I = identity matrix

$\mathbf{\Gamma}$ = axis vector ($\Gamma_1, \Gamma_2, \Gamma_3$) normalised to unit length

$\tilde{\mathbf{\Gamma}}$ = skew-symmetric matrix

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad , \quad \tilde{\mathbf{\Gamma}} = \begin{bmatrix} 0 & -\Gamma_3 & \Gamma_2 \\ \Gamma_3 & 0 & -\Gamma_1 \\ -\Gamma_2 & \Gamma_1 & 0 \end{bmatrix} \tag{AI.18}$$

$$\tilde{\mathbf{\Gamma}}^2 = \begin{bmatrix} -\Gamma_3^2 - \Gamma_2^2 & \Gamma_1\Gamma_2 & \Gamma_1\Gamma_3 \\ \Gamma_1\Gamma_2 & -\Gamma_3^2 - \Gamma_1^2 & \Gamma_2\Gamma_3 \\ \Gamma_1\Gamma_3 & \Gamma_2\Gamma_3 & -\Gamma_2^2 - \Gamma_1^2 \end{bmatrix} \tag{AI.19}$$

since $\Gamma_1^2 + \Gamma_2^2 + \Gamma_3^2 = 1$,

$$\tilde{\mathbf{\Gamma}}^2 = \begin{bmatrix} \Gamma_1^2 - 1 & \Gamma_1\Gamma_2 & \Gamma_1\Gamma_3 \\ \Gamma_1\Gamma_2 & \Gamma_2^2 - 1 & \Gamma_2\Gamma_3 \\ \Gamma_1\Gamma_3 & \Gamma_2\Gamma_3 & \Gamma_3^2 - 1 \end{bmatrix} \tag{AI.20}$$

therefore,

$$\mathbf{M} = \begin{bmatrix} 1 + (1-\cos(\gamma))(\Gamma_1^2 - 1) & \Gamma_1\Gamma_2(1-\cos\gamma) - \Gamma_3\sin\gamma & \Gamma_2\sin\gamma + \Gamma_1\Gamma_3(1-\cos\gamma) \\ \Gamma_3\sin\gamma + \Gamma_1\Gamma_2(1-\cos\gamma) & 1 + \Gamma_2^2(1-\cos\gamma) & -\Gamma_1\sin\gamma + \Gamma_2\Gamma_3(1-\cos\gamma) \\ -\Gamma_2\sin\gamma + \Gamma_1\Gamma_3(1-\cos\gamma) & \Gamma_1\sin\gamma + \Gamma_2\Gamma_3(1-\cos\gamma) & 1 + \Gamma_3^2(1-\cos\gamma) \end{bmatrix} \tag{AI.21}$$

The Rodrigues' rotation matrix can be written as,

$$\mathbf{M} = \begin{bmatrix} \cos\gamma + \Gamma_1^2(1-\cos\gamma) & \Gamma_1\Gamma_2(1-\cos\gamma) - \Gamma_3\sin\gamma & \Gamma_2\sin\gamma + \Gamma_1\Gamma_3(1-\cos\gamma) \\ \Gamma_3\sin\gamma + \Gamma_1\Gamma_2(1-\cos\gamma) & \cos\gamma + \Gamma_2^2(1-\cos\gamma) & -\Gamma_1\sin\gamma + \Gamma_2\Gamma_3(1-\cos\gamma) \\ -\Gamma_2\sin\gamma + \Gamma_1\Gamma_3(1-\cos\gamma) & \Gamma_1\sin\gamma + \Gamma_2\Gamma_3(1-\cos\gamma) & \cos\gamma + \Gamma_3^2(1-\cos\gamma) \end{bmatrix} \tag{AI.22}$$

APPENDIX II

GLOBAL OPTIMIZATION TEST PROBLEMS

A) Sum Squares function

$$
\begin{aligned}
&Min \quad f(u) = \sum_{i=1}^{n} i u_i^2 \\
&s.t. \quad -10 \le u_i \le 10 \\
&\qquad\qquad u \in \mathbb{R}^n \\
&\qquad\qquad f_{global} = f(0,0,...,0) = 0
\end{aligned}
\tag{AII.1}
$$

B) Zakharov function

$$
\begin{aligned}
&Min \quad f(u) = \sum_{i=1}^{n} u_i^2 + \left(\sum_{i=1}^{n} 0.5 i u_i \right)^2 + \left(\sum_{i=1}^{n} 0.5 i u_i \right)^4 \\
&s.t. \quad -5 \le u_i \le 10 \\
&\qquad\qquad u \in \mathbb{R}^n \\
&\qquad\qquad f_{global} = f(0,0,...,0) = 0
\end{aligned}
\tag{AII.2}
$$

C) Griewank function

$$
\begin{aligned}
&Min \quad f(u) = \sum_{i=1}^{n} \frac{u_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{u_i}{\sqrt{i}} \right) + 1 \\
&s.t. \quad -600 \le u_i \le 600 \\
&\qquad\qquad u \in \mathbb{R}^n \\
&\qquad\qquad f_{global} = f(0,0,...,0) = 0
\end{aligned}
\tag{AII.3}
$$

D) Ackley function

$$
\begin{aligned}
&Min \quad f(u) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} u_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi u_i)\right) + 20 + \exp(1) \\
&s.t. \quad -5.12 \le u_i \le 5.12 \\
&\qquad u \in \mathbb{R}^n \\
&\qquad f_{global} = f(0,0,\ldots,0) = 0
\end{aligned}
\tag{AII.4}
$$

E) Sphere function

$$
\begin{aligned}
&Min \quad f(u) = \sum_{i=1}^{n} u_i^2 \\
&s.t. \quad -10 \le u_i \le 10 \\
&\qquad u \in \mathbb{R}^n \\
&\qquad f_{global} = f(0,0,\ldots,0) = 0
\end{aligned}
\tag{AII.5}
$$

F) Dixon & Price function

$$
\begin{aligned}
&Min \quad f(u) = \sum_{i=1}^{n} i\left(2u_i^2 - u_{i-1}\right)^2 + \left(u_1 - 1\right)^2 \\
&s.t. \quad -10 \le u_i \le 10 \\
&\qquad u \in \mathbb{R}^n \\
&\qquad f_{global} = f(0,0,\ldots,0) = 0
\end{aligned}
\tag{AII.6}
$$

G) Initial values and parameters chosen for tests problems in section 4.4

Table VI.1 Initial values and parameters chosen for tests problems in section 4.4

Global optimization method		Parameters $k_s, k_\varphi, k_\vartheta = 1^{-3}, 1^{-2}, 1^{-1}$ $\varepsilon_\vartheta, \varepsilon = 1^{-2}, 1^{-2}$									
		Levy	***Hartman***	***Perm***	***Rosenbrock***	***Sum-Square***	***Zakharov***	***Griewank***	***Ackley***	***Sphere***	***Dixon-Price***
Deterministic	**MU u_0**	[3,4.7,-5]	[0.9,0.1,0.3]	[0.2,-0.3,0]	[3,-0.3,-3]	[-3,5 ,4]	[-2,5 ,3]	[-100,-200,100]	[2, 4,3]	[3,1 ,2.5]	[3,-4 ,2]
	MU Δ_0	10	1	2	5	10	10	600	10	5	10
	DIRECT $_{UL}$	[-7,-5.3,15]	[-0.1,-0.9,-0.7]	[-2,-2.3,-2]	[-2,-5.3,-8]	[-13, -5, -6]	[-12,-5,-7]	[-700,-800,-500]	[-8,-6,-7]	[-2,-4, -2.5]	[-7,-14,-8]
	DIRECT $_{UU}$	[13,14.7, 5]	[1.9, 1.1, 1.3]	[2,1.7,2]	[8, 4.7, 2]	[7, 15, 14]	[8,15,13]	[500,400,700]	[12,14,13]	[8,6,7.5]	[13,6,12]

APPENDIX III

COEFFICIENTS OF TEST PROBLEMS

A) Coefficients of test problem (DS1)

$$A=\begin{bmatrix} 0.2714 & -0.9599 & -0.2352 & 0.1934 & 0.8117 & -0.1594 & -0.3950 \\ 0.6780 & 0.8478 & -0.6509 & 0.2117 & -0.5066 & -0.3630 & 0.3367 \\ -0.6045 & -0.7784 & -0.1385 & -0.7399 & 0.0548 & 0.5176 & 0.5422 \\ 0.6150 & -0.7725 & -0.0826 & -0.1830 & -0.2971 & 0.2992 & 0.3729 \\ 0.8460 & -0.9961 & -0.3039 & -0.9387 & -0.0711 & -0.4147 & 0.4793 \\ 0.4871 & -0.9662 & 0.4973 & 0.4568 & -0.4924 & -0.1889 & -0.6153 \\ -0.2686 & -0.4950 & -0.7329 & -0.0492 & 0.9723 & 0.7968 & 0.3779 \end{bmatrix} \quad B=\begin{bmatrix} -0.7478 & 0.7777 & 0.5781 & 0.6247 & -0.0197 & -0.6259 & -0.5367 \\ 0.9192 & -0.5652 & -0.2816 & 0.2286 & 0.2714 & -0.6753 & 0.4826 \\ -0.7971 & 0.3184 & -0.5575 & -0.1719 & 0.6967 & -0.7428 & -0.8809 \\ -0.0947 & 0.0412 & 0.6120 & 0.5545 & 0.8851 & 0.2781 & -0.0653 \\ 0.9056 & -0.0303 & 0.8543 & -0.8339 & 0.5849 & 0.8431 & -0.7274 \\ -0.9190 & 0.5441 & 0.5619 & -0.5208 & -0.3413 & -0.6352 & 0.8680 \\ -0.0583 & -0.0347 & 0.2549 & -0.7239 & -0.5960 & 0.0400 & 0.2571 \end{bmatrix}$$

$$C=\begin{bmatrix} -0.0033 & 0.5523 & 0.5428 & 0.2097 & 0.2095 & -0.3882 & -0.4306 \\ 0.1521 & 0.4337 & 0.7967 & 0.0160 & 0.2824 & 0.4217 & 0.3263 \\ 0.9865 & -0.3172 & 0.1594 & -0.8101 & -0.6548 & 0.2544 & 0.5004 \\ -0.8917 & 0.6944 & -0.4253 & 0.1426 & -0.2606 & 0.3791 & -0.7831 \\ -0.0126 & -0.3281 & -0.5339 & 0.2327 & -0.2886 & -0.6984 & -0.3127 \\ -0.8019 & -0.5602 & 0.6121 & 0.4142 & -0.2340 & 0.8427 & 0.7567 \\ -0.3766 & -0.4785 & -0.1592 & -0.3374 & -0.0859 & 0.2698 & -0.8729 \end{bmatrix} \quad D=\begin{bmatrix} 0.4717 & -0.3836 & 0.4009 & -0.6915 & -0.9248 & 0.4964 & -0.4788 \\ -0.4928 & 0.4223 & 0.4874 & 0.8992 & -0.3956 & -0.8207 & 0.1599 \\ -0.0355 & -0.8624 & -0.3941 & 0.6236 & -0.1875 & -0.0179 & -0.9762 \\ 0.4306 & 0.7387 & 0.0297 & -0.6392 & -0.1405 & 0.3847 & 0.3676 \\ 0.2101 & 0.4296 & -0.0873 & 0.9564 & 0.8684 & -0.1059 & -0.7559 \\ 0.3953 & 0.7658 & -0.3411 & 0.8870 & -0.2851 & -0.9324 & -0.7488 \\ 0.8353 & 0.0183 & 0.0457 & -0.9942 & -0.2124 & 0.4405 & 0.0908 \end{bmatrix}$$

B) Coefficients of test problem (DS2)

$$A=\begin{bmatrix} 0.9484 & 0.3877 & -0.4345 & 0.8094 & -0.1901 & -0.0202 & -0.8334 \\ -0.6054 & -0.8163 & -0.6896 & -0.7381 & -0.6529 & 0.3897 & 0.0222 \\ -0.7776 & -0.1958 & -0.9987 & 0.6675 & 0.1504 & -0.1772 & -0.2663 \\ -0.4053 & -0.4096 & -0.4328 & 0.6009 & 0.2124 & -0.9304 & 0.4790 \\ -0.2072 & -0.3870 & 0.1016 & 0.8358 & -0.5711 & -0.4143 & 0.0495 \\ -0.1585 & -0.7889 & 0.7418 & -0.7254 & 0.0399 & 0.6029 & 0.6090 \\ -0.3770 & 0.1877 & -0.9155 & 0.0095 & 0.9784 & -0.3070 & 0.6338 \end{bmatrix} \quad B=\begin{bmatrix} -0.6211 & 0.0890 & -0.3360 & -0.9118 & -0.1196 & -0.1823 & 0.5064 \\ -0.7526 & 0.2129 & 0.6795 & 0.3734 & -0.6864 & 0.4160 & 0.4001 \\ 0.6420 & 0.5209 & -0.2566 & 0.4675 & -0.3479 & -0.7127 & -0.5710 \\ 0.2758 & 0.7107 & 0.6564 & -0.1257 & -0.3719 & 0.7426 & 0.3598 \\ -0.9678 & -0.2343 & -0.6470 & -0.2403 & 0.7890 & -0.8337 & 0.1146 \\ 0.7919 & -0.8307 & -0.7410 & 0.9593 & -0.5060 & -0.0765 & 0.7014 \\ 0.0308 & 0.4677 & 0.7598 & -0.2020 & -0.3786 & -0.9392 & 0.1171 \end{bmatrix}$$

$$C=\begin{bmatrix} 0.8035 & -0.4920 & -0.2528 & 0.6947 & -0.6754 & -0.6517 & -0.8624 \\ -0.1610 & -0.1376 & -0.5566 & 0.3598 & -0.9887 & 0.4572 & -0.6329 \\ -0.2837 & 0.4051 & -0.5620 & -0.7267 & 0.5430 & 0.0686 & 0.4741 \\ -0.0220 & -0.1953 & 0.0445 & 0.7168 & 0.5296 & -0.4939 & 0.3934 \\ -0.4881 & -0.6363 & -0.1332 & -0.6003 & -0.1579 & 0.8341 & 0.5540 \\ 0.8583 & 0.7125 & 0.4826 & 0.2147 & -0.8864 & 0.5164 & 0.0038 \\ -0.0665 & 0.1684 & -0.8591 & 0.0861 & 0.1715 & 0.7741 & -0.1490 \end{bmatrix} \quad D=\begin{bmatrix} 0.2225 & 0.1354 & -0.9938 & 0.8523 & 0.7133 & -0.8330 & 0.4924 \\ 0.7115 & 0.7757 & -0.8251 & -0.4030 & -0.2314 & -0.4403 & -0.0642 \\ 0.3416 & 0.6859 & -0.4785 & -0.3238 & 0.3914 & -0.1060 & 0.7217 \\ 0.0472 & 0.7976 & -0.9544 & 0.7190 & 0.2558 & 0.1751 & -0.0670 \\ -0.4024 & 0.8780 & -0.1518 & -0.3190 & -0.0992 & 0.7553 & -0.0038 \\ 0.4079 & 0.6309 & -0.3179 & -0.7238 & -0.0528 & -0.0618 & -0.0251 \\ -0.2368 & -0.9973 & 0.0827 & 0.0156 & 0.8994 & -0.1252 & -0.5411 \end{bmatrix}$$

C) Coefficients of test problem (H3)

$$C_k = \begin{bmatrix} 1 & 1.2 & 1 & 3.2 \end{bmatrix}^T$$

$$A_{ki} = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}$$

$$P_{ki} = \begin{bmatrix} 0.3689 & 0.1170 & 0.2673 \\ 0.4699 & 0.4387 & 0.7470 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}$$

APPENDIX IV

DEFAULT VALUES OF THE TUNING PARAMETERS FOR STOCHASTIC ALGORITHMS

Table VII Default values of the tuning parameters of Simulated Annealing in the optimization Toolbox 4.0 of MATLAB version 7.6.0.347 (R2009a)

	Simulated Annealing	Default values
Tuning factors	*Annealing function*	Fast annealing
	Reannealing interval	100
	Temperature update function	Exponential temperature update
	Acceptance probability function	Simulated annealing acceptance
	Problem type	Double
	Hybrid function	None
	Hybrid function call interval	End
Stopping criteria	*Maximum iterations*	Inf
	Maximum function evaluation	3000×number of variables
	Time limit	Inf
	Function tolerance	1e-6
	Objective limit	-Inf
	Stall iteration	500×number of variables

Table VIII Default values of the tuning parameters of Genetic Algorithm in the optimization Toolbox 4.0 of MATLAB version 7.6.0.347 (R2009a)

Genetic Algorithm		Default values
Tuning factors	*Population type*	Double vector
	Creation function	Constraint dependent
	initial population	[]
	Initial scores	[]
	Initial range	[0;1]
	Fitness scaling	Rank
	Selection function	Stochastic uniform
	Elite count	2
	Crossover fraction	0.8
	Mutation function	Constraint dependent
	Crossover function	Scattered
	Migration direction	Forward
	Migration fraction	0.2
	Migration interval	20
	Initial penalty	10
	Penalty factor	100
	Hybrid function	None
	Fitness and constraint function evaluation	In Serial
Stopping criteria	*Generations*	100
	Time limit	Inf
	Fitness limit	-Inf
	Stall generations	50
	Stall time limit	Inf
	Function tolerance	1e-6
	Nonlinear constraint tolerance	1e-6