UNIVERSITE DE BLIDA 1

Faculté de Technologie

Département d'Electronique

THESE DE DOCTORAT

en Génie Electrique

FUZZY PREDICTIVE CONTROL USING META-HEURISTIC ALGORITHMS

Par

Oussama AIT SAHED

Devant le jury composé de :

A. GUESSOUM	Professeur, U. de Blida 1	Président
F. BOUDJEMA	Professeur, ENP, Alger	Examinateur
K. BENMANSOUR	Professeur, ESDAT, Alger	Examinateur
M. CHETTOUH	MCA, U. de Blida 1	Examinateur
K. KARA	Professeur, U. de Blida 1	Rapporteur

Blida, 17 Décembre 2015

REMERCIEMENTS

Je remercie Dieu le compassion et le miséricordieux de m'avoir aidé et donné patience et courage pour accomplir ce modeste travail.

Je tiens à exprimer ma reconnaissance à mon directeur de thèse, le professeur Kamel KARA, qui m'a donné courage et motivation à aller jusqu'au bout du chemin et je suis reconnaissant pour sa présence à mes côtés aux moments de doutes, pour ses précieux conseils, aide et soutient.

Je tiens à remercier le professeur Abderrezak GUESSOUM pour l'honneur qu'il m'a fait en acceptant la présidence de mon jury de thèse.

Je remercie vivement le professeur Farès BOUDJEMA d'avoir accepté d'examiner mon travail et de faire partie du jury.

J'exprime toute ma gratitude au professeur Khelifa BENMANSOUR d'avoir accepté d'examiner ce travail et de faire partie du jury.

J'exprime mes vifs remerciements à monsieur Mourad CHETTOUH, qui a accepté de faire partie des membres du jury, et d'examiner cette thèse.

Je remercie également les professeurs M'hamed BOUNEKHLA, Noureddine GOLEA et Ahmed BENALLAL pour leurs conseils et recommandations.

Je tiens aussi a remercié mon collègue le doctorant Abousoufyane Benyoucef pour son aide et pour l'atmosphère de fraternité et le confort que j'ai senti durant mes travaux dans le laboratoire d'automatique à ses côtés.

Je remercie aussi le docteur Mohamed Laid HADJILI, le professeur Vincent WERTZ, le staff du laboratoire LABSET et du département d'électronique de l'université de Blida1 ainsi que mes collègues et mes amis, pour leur aide et soutient tout au long de ce travail.

A la fin, je remercie mes parents, ainsi que toute ma famille, pour leur amour, soutient et encouragement. Ils m'ont toujours donné confiance en soi et m'ont motivé pour toujours marcher en avant.

ABSTRACT

The aim of this work is to develop simple and efficient nonlinear predictive control algorithms. The idea is to use the meta-heuristic approach to find, in a reasonable computational time, an accurate suboptimal solution to the nonlinear optimization problem. Mainly, particle swarm optimization and artificial bee colony approaches are envisaged. Without loss of generalization to other models, Takagi-Sugeno fuzzy models are used to construct prediction models.

The first proposed control algorithm is based on a modified version of the PSO algorithm. The key idea of this algorithm is to use a reduced population size with a small number of iterations in order to reduce the required computational time. This can be achieved by distributing the initial particles positions, according to the normal distribution law within the area around the current best position. The radius limiting this area is adapted according to the tracking error value.

Artificial bee colony algorithm, a recently introduced meta-heuristic optimization algorithm, has several characteristics that make it more attractive than other meta-heuristic methods to design efficient nonlinear predictive control algorithms. Two improved versions of this algorithm that allow overcoming some of its deficiencies are proposed and used to develop two nonlinear predictive control algorithms.

The performance and computational efficiency of the different proposed algorithms are evaluated by considering a series of numerical benchmark functions, the control of two benchmark systems and a DSP based experimental setup. The obtained results are commented and compared with those of several other methods.

Keywords: nonlinear predictive control; meta-heuristic algorithms; DSP

RESUME

Le but de ce travail est de développer des algorithmes simples et efficaces pour la commande prédictive non linéaires. L'idée est d'utiliser l'approche d'optimisation méta-heuristique pour trouver, dans un temps de calcul raisonnable, une solution sous-optimale au problème d'optimisation non linéaire. Principalement, les approches optimisation par essaim de particules et colonies d'abeilles artificielles sont envisagées. Sans perte de généralisation à d'autres types de modèles, les modèles flous de Takagi-Sugeno sont utilisés dans la mise en œuvre des modèles de prédiction.

Le premier algorithme de commande proposé est basé sur une version modifiée de l'algorithme PSO. L'idée de base de cet algorithme est d'utiliser un nombre réduit de particules et d'itérations pour réduire le temps de calcul nécessaire. Ceci peut être obtenu en distribuant les positions initiales des particules, en utilisant la loi de distribution normale, à l'intérieur de la zone autour de la meilleure position actuelle. Le rayon délimitant cette zone est adapté en fonction de la valeur de l'écart entre la sortie du système et la trajectoire de référence.

L'algorithme des colonies d'abeilles artificielles, un algorithme d'optimisation méta-heuristique récemment introduit, a plusieurs caractéristiques plus attractives que les autres méthodes d'optimisation méta-heuristiques. Deux versions améliorées de cet algorithme permettant de surmonter un certain nombre de ses inconvénients, sont proposées et utilisées pour développer deux algorithmes de commande prédictive non linéaire.

Les performances et l'efficacité de calcul des différents algorithmes proposés sont évaluées en considérant une série de fonctions benchmark, la commande de deux systèmes de complexité différente et un dispositif expérimental basé sur une implémentation sur DSP. Les résultats obtenus sont commentés et comparés à ceux de plusieurs autres méthodes.

Mots-clés: Commande prédictive non-linéaire; algorithmes méta-heuristiques; DSP

ملخص

ان الهدف من وراء هذا العمل هو اشتقاق خوارزميات تحكم بسيطة وفعالة من أجل حل مشكلة التحكم التنبؤي غير الخطي. ان الفكرة تتلخص في استعمال خوارزميات البحث المتقدم من أجل الحصول على حلول دقيقة لمشكلة ايجاد الحل الأمثل باستعمال ابسط امكانيات حسابية ممكنة. لقد قمنا في هذا العمل باستعمال خوارزميات البحث المتقدم المسماة PSO و ABC، بالاضافة الى تبني النماذج التنبؤية غير الخطية من نوع Takagi-Sugeno، مع العلم أن النتائج المحصّلة تبقى صالحة في حال تم تبني نوع آخر من النماذج.

لقد قمنا باقتراح خوارزمية تحكم استنادا على PSO. ان الفكرة العامة وراء هذا الاقتراح تتلخص في استعمال مجموعة سكانية صغيرة مع عدد أجيال محدود لتقليص الامكانيات الحسابية المطلوبة عن طريق التوزيع الذكي لعناصر المجموعة السكانية الأولية حول الحل الأمثل الحالي باستعمال قانون التوزيع الغاوسي. هذا التوزيع يتم في دائرة نصف قطرها يتم تحديده دوريا باستعمال خطأ التتبع بين مخارج النظام و مسارات المرجعية المنشودة.

مستعمرة النحل الاصطناعية (ABC)، خوارزمية بحث متقدم حديثة الانشاء، تتميز بالعديد من الخصائص التي تجعلها أكثر جاذبية من الخوارزميات الأخرى عند تصميم خوارزميات التحكم التنبؤي غير الخطي البسيطة و الفعالة. تم اقتراح و استعمال نسختين محسنتين من هذه الخوارزمية بهدف التغلب على بعض النواقص المسجلة و من أجل الحصول على حلول أكثر دقة، خاصة عند حل مشكلة ايجاد الحل الأمثل الخاصة بنظام التحكم التنبؤي غير الخطي.

آداء و كفاءة الخوارزميات الحسابية تم تقييمها من خلال استعمال مجموعة من الدّوال الرقمية، مشكلتي تحكم قياسيتين بالاضافة الى تجارب تطبيقية باستعمال DSP. لقد قمنا بعد ذلك بتجميع النتائج و التعليق عليها، بالاضافة الى مقارنتها مع نتائج العديد من الخوارزميات الأخرى.

كلمات البحث: تقنية التحكم التنبؤي غير الخطية ; خوارزميات البحث المتقدم ; DSP

CONTENT

PREFACE 1

ABSTACT 2

CONTENT 5

LIST OF ILLUSTRATIONS, GRAPHICS AND TABLES 9

INTRODUCTION 12

1. MODEL PREDICTIVE CONTROL

1. Introduction 17

2. Predictive control principle 17

2.1. Cost function 19

2.2. Prediction model 21

2.3. Constraints 22

3. Linear model predictive control 24

3.1. Prediction 25

3.2. Control law 27

3.3. The constrained problem 28

3.3.1. Formulation as a quadratic programming (QP) problem 28

3.3.2. Solving convex QP problems 30

4. Nonlinear model predictive control 39

4.1. Nonlinear model predictive control with successive linearization (NMPC-SL) 40

4.2. Nonlinear model predictive control with nonlinear predictions and linearization (NMPC-NPL) 40

4.3. Nonlinear model predictive control with nonlinear optimization (NMPC-NO) 41

4.3.1. Deterministic numerical approaches 42

4.3.2. Stochastic numerical approaches 49

5. Efficient ways to reduce the NMPC on-line computing requirement 50

6. Feasibility issues 52

7. Stability 54

7.1. State terminal equality constraint 55
7.2. Dual mode (Terminal constraint set) 56
7.3. Contractive NMPC 57
7.4. Quasi-infinite horizon NMPC 57
8. Conclusion 58

2. META-HEURISTIC ALGORITHMS
1. Introduction 59
2. Meta-heuristic algorithms: Basics 59
2.1. Classification 60
2.2. Populations based meta-heuristics 61
2.2.1. Initial population 62
2.2.2. Population size 62
2.2.3. Exploitation versus exploration 63
2.2.4. Stopping criteria 64
3. Solving optimization problems 65
3.1. General approach 65
3.2. Constraints handling 66
3.2.1. Reject strategies 66
3.2.2. Penalizing strategies 67
3.2.3. Repairing strategies 67
3.2.4. Preserving strategies 67
4. Variants of meta-heuristic algorithm 67
4.1. Genetic algorithm 68
4.1.1. Natural selection 69
4.1.2. Pairing 69
4.1.3. Mating 70
4.1.4. Mutation 71
4.2. Particle swarm optimization 71
4.3. Artificial bee colony 74
4.3.1. Basic algorithm 74
5. Proposed variants of the ABC algorithm 77
5.1. ABC Enhanced Version (ABCEV) 78
5.1.1. Initialization phase 78

5.1.2. Update equation 78
5.1.3. Scout bee 79
5.2. Equal Exploitation ABC (EEABC) 79
5.2.1. New probability equation 79
5.2.2. Adaptive exploration rate 81
5.2.3. Adaptive update mechanism 82
5.3. Experimental study on numerical benchmark functions 83
5.3.1. Benchmark functions 84
5.3.2. Experimental setup 86
5.3.3. Comparative results 86
6. Conclusion 94

3. FUZZY MODEL PREDICTIVE CONTROL BASED ON META-HEURISTIC ALGORITHMS

1. Introduction 95
2. Fuzzy based model predictive control 95
2.1. Takagi-Sugeno dynamic fuzzy modelling 95
2.2. Notation 98
2.3. Solving the fuzzy NMPC optimization problem 99
2.3.1. Constraints handling 99
2.3.2. Why using the artificial bee colony? 102
3. Proposed control algorithms 103
3.1. Efficient PSO based controller 103
3.2. The ABCEV based controller 107
3.3. The EEABC based controller 110
4. Applications 113
4.1. CSTR 113
4.1.1. Process description 113
4.1.2. Fuzzy identification 114
4.1.3. Controllers implementation 115
4.2. Industrial Boiler 122
4.2.1. Process description 122
4.2.2. Fuzzy identification 125
4.2.3. Controller implementation 126

5. DSP-based implementation 133
5.1. Description of the eZDSP2812 test bench 133
5.2. Fuzzy identification and controllers implementation 134
5.3. Comparative study 135
6. Conclusion 139
CONCLUSION 140
APPENDIX
A. LIST OF ABBREVIATIONS 145
B. LIST OF SYMBOLS 147
C. CEC 2015 TECHNICAL REPORT 150
D. LIST OF PUBLICATIONS 163
REFERENCES 164

LIST OF ILLUSTRATIONS, GRAPHICS AND TABLES

Figure 1.1	Principle of Model Predictive Control Strategy	18
Figure 2.1	Flowchart of the standard GA	69
Figure 2.2	Flowchart of the ABC algorithm	77
Figure 2.3	Convergence speed for function f1 with D=60	92
Figure 2.4	Convergence speed for function f6 with D=30	92
Figure 2.5	Convergence speed for function f8 with D=100	93
Figure 2.6	Convergence speed for function f10 with D=30	93
Figure 3.1	Bloc diagram of the proposed NMPC algorithm	99
Figure 3.2	Weight function $\Gamma_y(y)$	100
Figure 3.3	Continuous stirred tank reactor (CSTR)	113
Figure 3.4	Model and process validation for the CSTR	114
Figure 3.5	Reference trajectory for the CSTR	115
Figure 3.6	Control performances of the proposed control algorithms for the CSTR	118
Figure 3.7	Control signals of the proposed control algorithms for the CSTR	118
Figure 3.8	Control performances of the different control algorithms for the CSTR	119
Figure 3.9	Control performances of the different control algorithms between 40 and 50 min for the CSTR	120
Figure 3.10	MCV average values for the CSTR	121
Figure 3.11	The considered industrial boiler	122
Figure 3.12	Stabilization scheme for the industrial boiler	124
Figure 3.13	TS fuzzy model validation for the industrial boiler (Blue solid line: Process output and red solid line: Process model output)	125
Figure 3.14	Stabilization scheme for the boiler without direct feedthrough	128
Figure 3.15	Response of the boiler using the proposed control algorithms and the linear MPC controller (Blue dashed line:	129

	ABCEV based controller; the red dashed dotted is the EEABC based controller; brown dotted line is the efficient PSO based controller and the green dotted is the linear MPC based controller)	
Figure 3.16	Response of the boiler for the different proposed control algorithms and the linear MPC controller between the 200th and 400th samples for the output y_4 (Blue dashed line: ABCEV based controller; the red dashed dotted is the EEABC based controller; brown dotted line is the efficient PSO based controller and the green dotted is the linear MPC based controller)	130
Figure 3.17	Control signals for the considered controllers and the linear MPC controller (Blue dashed line: ABCEV based controller; the red dashed dotted is the EEABC based controller; brown dotted line is the efficient PSO based controller and the green dotted is the linear MPC based controller)	131
Figure 3.18	MCV average values for the industrial boiler	132
Figure 3.19	DSP based test bench	134
Figure 3.20	Reference trajectory for the DSP based implementation	135
Figure 3.21	Speed of the free load motor for the implemented controllers	136
Figure 3.22	Speed of the motor for the implemented controllers	137
Figure 3.23	Control signals for the implemented controllers	138

Table 2.1	Characteristics of the different considered variants of the ABC algorithm	83
Table 2.2	Unimodal benchmark functions	85
Table 2.3	Multimodal benchmark functions	85
Table 2.4	Summary of the CEC2015 expensive optimization test problems	86
Table 2.5	Comparative results of the convergence quality for the	89

	standard benchmark functions	
Table 2.6	Comparative results of the convergence iteration	90
Table 2.7	Comparative results of the convergence quality for the CEC 2015 benchmark functions	91
Table 3.1	Execution time of the considered algorithms for the CSTR	121
Table 3.2	Design parameters	126
Table 3.3	Execution time of the considered control algorithms for the industrial boiler	133
Table 3.4	Execution time of the considered control algorithms (DSP)	139

INTRODUCTION

Model Predictive Control (MPC), also known as moving horizon control or receding horizon control, is an advanced control strategy that has been developed in late seventies. It is based on the use of an explicit model to online predict the process future behaviour over a given finite horizon, and then computes a control sequence that minimizes a given cost function. This function usually takes the form of a quadratic function of the errors between the predicted responses and the reference trajectory, and includes in most cases the control effort.

Since its creation, the predictive control has received a lot of attention from the research community. Richalet, Rault, Testud and Pagon were the first to introduce the predictive control philosophy with their Model Predictive Heuristic Control (MPHC) strategy [1, 2], later followed by the Dynamic Matrix Control (DMC) [3], and a number of other design techniques like: Internal Model Control (IMC) [4], Linear Dynamic Matrix Control (LDMC) [5], Quadratic Dynamic Matrix Control (QDMC) [6], and the famous Generalized Predictive Control (GPC) technique introduced by Clarke and Mohtadi [7, 8].

The MPC strategy is an advanced control technique that has been successfully applied in many fields. This success can be attributed to three important factors:

1) The use of an explicit process model which allows the controller to consider all of the process dynamics.

2) The MPC algorithm considers process behaviour over a future horizon in time. This means that the effects of feedforward and feedback disturbances can be anticipated and removed providing better performances.

3) The ability to handle constraints directly during the design of the controller [9].

In fact, this last property is the main reason behind its popular use in many practical applications such as chemicals, polymers, air and gas processing, refining, petrochemical, and food processing industries [10, 11]. Furthermore, MPC techniques are able to deal with complex control problems which involve

multivariable process interactions, non-minimum phase behaviour, and variable or unknown time delays [12, 13].

Although classical MPC algorithms, which use a linear prediction model, provide satisfactory performance in many applications, against highly nonlinear process, severe degradation in control performance can occur, unless the operating conditions are very close to the steady state around which the model is linearized [13]. Since the most practical processes are nonlinear by nature, new efficient techniques, based on nonlinear models, have to be derived to incorporate nonlinearities and ensure higher control performance. Indeed, a lot of attention was given to the Nonlinear MPC (NMPC) strategies and several nonlinear control algorithms have been proposed [14-16]. The use of nonlinear models allows enhancing the overall controller performance; however the corresponding optimization problem, which is nonlinear and non-convex, requires complex and time consuming procedures [17-19]. Moreover, driving appropriate nonlinear models is often a difficult task. It is clear that good and accurate system output predictions are a prerequisite in any efficient predictive control scheme. Therefore the developed model must have the ability to faithfully mimic the behaviour of the real process, especially the dependence between the outputs, the current measured variables and the current/future inputs. Nevertheless, obtaining an exact representation of the physical process is impossible, only an approximation can be constructed to predict the future process behaviour for a given control sequence.

A large class of various nonlinear models, which can be used within predictive control strategies, exists. Nonetheless, the simplest model that gives enough accurate predictions should be used [20]. To get reliable predictions, it is not always necessary to include all of the physical, chemical and internal behaviour of the process into the constructed model; but rather to only include the dynamics that affect the predictions.

Fuzzy inference systems (FIS) are particularly interesting approaches that were satisfactory used in nonlinear systems modelling [13, 17, 21]. Indeed, it has been shown that FIS are capable of approximating any continuous function with a certain level of accuracy [22]. Takagi-Sugeno (TS) models, a subdivision of fuzzy models, are widely used and particularly suitable for NMPC algorithms [17, 23].

These models are able to express the dynamic nature of systems with characteristics of randomness, large delay time and strong nonlinearity [24, 25].

Obtaining optimal solutions to the optimization problem, when a nonlinear model is used, is a difficult task. In fact, all the proposed NMPC techniques seek to find a suboptimal solution of adequate quality that can meet the control objectives. Depending on how it is proceeded to obtain the suboptimal solution, Tatjewski [26] has classified the existing methods into three main categories:

- Linearization of the nonlinear model in order to obtain an approximate linear and convex optimization problem, which has a straightforward solution.

- Using a prediction model that is given by the sum of two terms: a free response term, obtained from the nonlinear model of the system, and a forced response term, computed using a linear model. The obtained optimization problem is convex and can be resolved using quadratic programming algorithms.

- Nonetheless, with highly nonlinear processes that have quick reaction times, approximating the nonlinear model in order to obtain a convex optimization problem is insufficient. In fact, nothing but the complete use of the nonlinear model appears to be acceptable. This class of algorithms tackles the nonlinear and generally non-convex optimization problem directly using nonlinear optimization techniques.

Numerical approaches were extensively used to solve nonlinear optimization problem (NOP) [26-28]. We can distinguish two major families:

- Deterministic numerical approaches, which include the famous sequential quadratic programming (SQP) and the nonlinear interior point (NIP) methods. When using these methods, the nonlinear optimization problem will be handled by solving a series of linear sub-problems. A large number of NMPC algorithms have been proposed using this approach [29-32]. The major drawbacks of these methods are the sensitivity to the initial conditions; as suitable initial guesses must be provided to start the iterative process and to ensure the convergence toward the global optimum [33-35]. For this reason, these approaches are sometimes called local methods [33]. Furthermore, these methods could not be used with some empirical models where the derivative details are either inaccessible or extremely hard to get [36]. An additional drawback is the fact that these

approaches could only be used with continuous variables; they cannot handle discrete problems [37].

- Stochastic numerical approaches which rely on stochastic meta-heuristic optimization algorithms to directly solve the NOP. They are based on fundamental elements and procedures that produce evolution and intelligent behaviours in natural systems [38] or on physical principles [39], and are capable of handling the majority of existing optimization problems [39, 40]. Stochastic meta-heuristic algorithms have known huge success, both in academia and the industry [39, 41]. More enhanced variants and completely new meta-heuristic algorithms are continuously developed and proposed by an increasingly active research community, and the industry is adopting these methods to solve the different existing engineering problems. This success could be attributed to the fact that the meta-heuristic algorithms are simple, flexible, derivative-free, and able to handle local optima while generating high quality solutions within a reasonable period of time [39, 42-44]. Due to their heuristic and random natures, these algorithms have a better searching capability than those of the classical approaches [45]. They are also unaffected by initial conditions and can handle both discrete and continuous problems.

Although different approaches to solve the NMPC problems using meta-heuristic algorithms have been proposed [34, 46-48], compared to the deterministic numerical methods, their use within the predictive control framework is relatively limited.

The ABC, a recently introduced optimization algorithm, has been a distinctive meta-heuristic algorithm. Several comparative studies [49, 50] between this algorithm and several other meta-heuristic algorithms such as GA (Genetic Algorithms), PSO (Particle Swarm Optimization) and DE (Differential Evolution), have shown that the ABC performances are better or at least similar to the performances of these algorithms. Furthermore, this algorithm was found to be simpler, more computing efficient and to generate more accurate solutions compared to other established algorithms. A survey [51] reviewing the advances related to the ABC algorithm and its applications has indicated that more than 330 research papers were published within the scope of merely seven years of its

creation. Seeing that more scholars adopt this algorithm, this number is expected to increase exponentially in the near future.

The efficiency and the performance of any NMPC based controller depend on two key aspects of its design. The modelling of the nonlinear physical process and the way the NOP is handled. In the present work, Takagi-Sugeno fuzzy approach is used to construct the different required nonlinear models. This choice is based on the previously mentioned advantages of fuzzy systems and Takagi-Sugeno models.

The second design aspect, which deals with the NOP, is in fact the focal point of our work. The main objective of this work is to propose efficient nonlinear model predictive controllers based on meta-heuristic algorithms, as viable and practical substitutes to the conventional NMPC controllers. Indeed, three control algorithms are proposed: The first one is based on an efficient PSO algorithm, and the other two are based on proposed ABC algorithms. In order to evaluate the performances of these algorithms, several comparative studies using several numerical benchmark functions and two control benchmark systems are carried out. To further asses the efficiency of the developed algorithm, their implementation using a DSP (digital signal processor) board is also considered.

The thesis layout is as follows: The first chapter deals with the main concepts and the implementation issues of model predictive control. Both linear and nonlinear control technique along with several control algorithms are described. The feasibility and the stability issues are also addressed. The second chapter is fully dedicated to meta-heuristic optimization algorithms. The chapter starts with a description of the different notions associated with these algorithms, followed by a detailed description of three meta-heuristic algorithms, namely the genetic algorithm (GA), the particle swarm optimization (PSO) and the artificial bee colony (ABC). The proposed ABC algorithms are presented and evaluated against several meta-heuristic optimization algorithms using a number of numerical benchmark functions. In the last chapter, three proposed NMPC based control algorithms are presented and their control performances are evaluated and compared with that of several conventional linear and nonlinear control techniques by considering the control of SISO continuous stirred tank reactor model and the MIMO industrial boiler model.

CHAPTER1:

MODEL PREDICTIVE CONTROL

1. Introduction

This chapter introduces the main concepts of the model predictive control and gives some techniques used to solve both linear and nonlinear Model Predictive Control (MPC) problems. The feasibility issue and some approaches used to guarantee the stability of the closed loop are given at the end of this chapter.

This chapter is built, if not otherwise indicated, based on the books given by the references [10, 12, 26].

2. Predictive control principle

The general principle of the model predictive control is based on using an explicit model to online predict the process future behaviour over a given finite prediction horizon, and on determining a control sequence that optimizes a given cost function. This strategy is illustrated, for a single-input single-output (SISO) process, in figure 1.1. Usually, the control input sequence $\hat{U}(t|t)=\{\hat{u}(t|t),\hat{u}(t+1|t),\ldots,\hat{u}(t+N_p-1|t)\}$ along the prediction horizon N_p is computed over a smaller horizon known as control horizon $N_u\ (N_u \le N_p)$, and the control inputs beyond the horizon N_u are kept constant (that is: $\hat{u}(t+j|t)=\hat{u}(t+N_u-1|t)$ for $j \ge N_u$). The notation $\hat{u}(t+j|t)$ for $0 \le j \le N_p-1$ is used to express the value of the control input u at the sampling time $t+j$ that is computed at the sampling time t. $y(t)$ is the measured process output, and $\hat{y}(t+j|t)$ and $w(t+j)$ $(1 \le j \le N_p)$ are the predicted values of the process output and the desired reference trajectory over the prediction horizon, respectively.

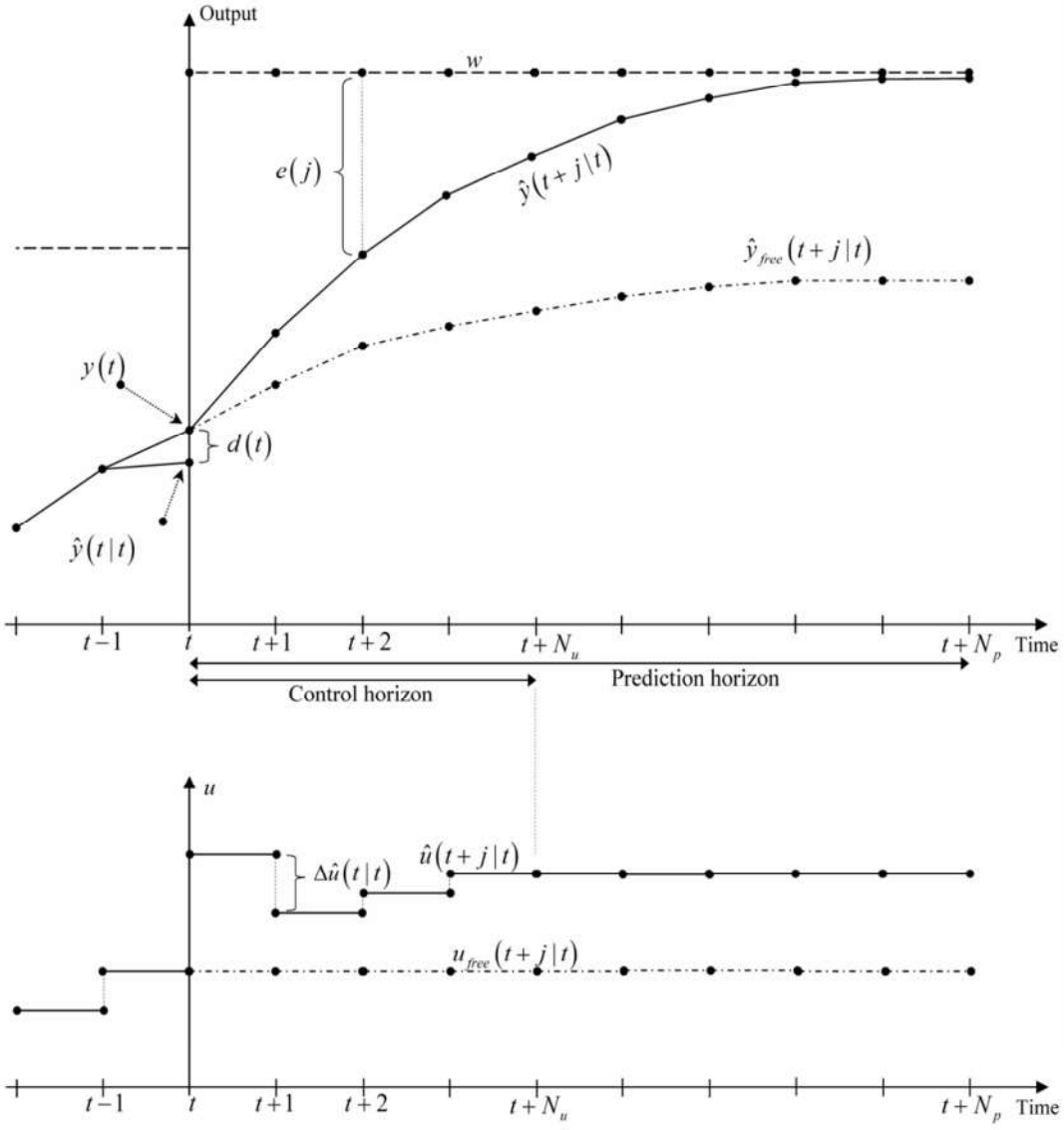

Figure 1.1 : Principle of model predictive control strategy.

All the MPC strategies can be described by the following steps:

- At each sampling time, the future values of the process outputs are computed over a prediction horizon using the process model.
- A reference trajectory is specified over at least the prediction horizon.
- A future control sequence that minimizes the cost function over a control horizon is computed. Only, the first element of this sequence is applied to the system.
- In the next sampling time, the preceding steps are repeated according to the receding horizon concept.

It is clear that good and accurate system output predictions are prerequisite in any efficient predictive control scheme. Nevertheless, obtaining an exact representation of the physical process is impossible. Only an approximation can be constructed to predict the future process behaviour for a given control sequence. Furthermore, the manipulated variables are often affected by perturbations and measurement noise. The difference between the outputs predictions and the actual measured outputs is denoted by $d(t) = y(t) - \hat{y}(t \mid t)$.

To simplify notation, the vertical bar '$\mid t$' used within the predicted variables is sometimes omitted.

2.1. Cost function

The predictive control is formulated as a minimization problem of a given objective function (cost function) that includes all the desired control objectives over the prediction horizon. Mainly, it contains a term representing the tracking error between the actual process output and the reference trajectory and another one representing the control signals energy. To make solving the predictive control problem simpler and quicker, the cost function is usually expressed as a quadratic function. The most commonly used one has the following form:

$$\min_{\Delta\hat{u}(t)} J\left(\Delta\hat{u}(t), \hat{y}(t), w(t)\right)$$

where $J\left(\Delta\hat{u}(t), \hat{y}(t), w(t)\right) = \sum_{j=N_1}^{N_2}\left[\left(\hat{y}(t+j \mid t) - w(t+j)\right)^T Q(j)\ \left(\hat{y}(t+j \mid t) - w(t+j)\right)\right]$

$$+ \sum_{j=1}^{N_u}\left[\Delta\hat{u}(t+j-1)^T\ R(j)\ \Delta\hat{u}(t+j-1)\right] \qquad (1.1)$$

$$= \sum_{j=N_1}^{N_2}\left\|\hat{y}(t+j \mid t) - w(t+j)\right\|^2_{Q(j)} + \sum_{j=1}^{N_u}\left\|\Delta\hat{u}(t+j-1)\right\|^2_{R(j)}$$

where J is the cost function, N_1 and N_2 are respectively the minimum and the maximum prediction horizons and Q and R are the weight matrices.

In the evaluation of the cost function of equation (1.1), the tracking errors starting from $t+N_1$ to the end of the prediction horizon $t+N_2$, are considered, where $1 \le N_1 \le N_2$. It is not necessary to start penalizing the tracking errors from the sampling period $t+1$ by taking $N_1 = 1$, because a delay could exist between

applying a given control action and seeing its effect on the system output. Therefore, it is unnecessary to include the tracking errors of samples that cannot be influenced by the current control sequence. Generally N_1 is chosen based on this delay, although sometimes in the literature, $N_1 = 1$ is assumed to simplify the notation [26]. This choice will not change the obtained results, but rather increases slightly the computational burden of the problem. For simplification, the above notation of $N_1 = 1$, unless indicated otherwise, will be adopted in the remaining of this dissertation. We note that it is possible to penalize the tracking errors of some chosen samples within the prediction horizon and ignore the others. The control horizon on the other hand must satisfy the following constraint $0 < N_u \leq N_2$. Nevertheless, to decrease the dimensionality of the optimization problem and obtain a more computing efficient formulation, it is usually preferred to select $N_u < N_2$.

The matrix $Q(j)$ is used to penalize the tracking error $e(j)$ between the predicted outputs and the reference trajectory along the prediction horizon. Most predictive controllers adopt this approach, although, in some situations, the tracking errors of the different sampling times are not evenly penalized. In the case where the scaling of the tracking error is not required, $Q(j)$ is replaced with a unity matrix I of the same dimensionality. The second matrix $R(j)$ has the same role as $Q(j)$ but with the control input increments $\left(\Delta\hat{u}(t+j-1) \text{ for } j = 1,\ldots,N_u\right)$ instead of the tracking errors $\left(e(t+j) \text{ for } j = 1,\ldots,N_2\right)$. This matrix is sometimes called the move suppression factor since any relative increase in its values relative to the weights of $Q(j)$ has the effect of reducing the control activity [12]. To ensure that $J \geq 0$, we must have $Q(j) > 0$ and $R(j) > 0$.

When choosing, in the formulation given by equations (1.1), $Q(j) = I$ and $R(j) = \lambda I$, the following typical cost function is obtained [20, 26]:

$$\min_{\Delta\hat{u}(t)} J\left(\Delta\hat{u}(t), \hat{y}(t), w(t)\right) = \min_{\Delta\hat{u}(t)} \sum_{j=1}^{N_p} \left\| \hat{y}(t+j \mid t) - w(t+j) \right\|^2 + \lambda \sum_{j=1}^{N_u} \left\| \Delta\hat{u}(t+j-1 \mid t) \right\|^2 \qquad (1.2)$$

where λ is a positive scalar that defines the weighting factor of the control inputs variations.

The second term in the right side of (1.2) could be completely taken out by setting $\lambda = 0$ to remove any restriction on the control actions changes. This will often lead to the presence of huge input changes and insufficient robustness against modelling errors [26]. In some situations, the values of the control inputs are also penalized by introducing the term $\sum \left\| \hat{u}(t+j \mid t) - u_0 \right\|_S^2$ to force the control inputs to follow some ideal resting value. This is done when there is more inputs than variables to be controlled [12].

The prediction horizons N_1 and N_2, the control horizon N_u, the weight matrices $Q(j)$ and $R(j)$ are the design parameters of the predictive controller. These parameters, in addition to the reference trajectory $w(t)$ affect the behaviour of the closed-loop combination of the process and the predictive controller. Choosing the appropriate values for these parameters is sometimes dictated by the economic objectives, but generally, they are tuned to meet the desired control performance.

2.2. Prediction model

The first step in designing model predictive controllers is obtaining the prediction model, which will be used to predict the future values of the process outputs over a given prediction horizon. This model must have the ability to faithfully mimic the behaviour of the real process, especially the dependence between the outputs, the current measured variables and the current/future inputs. The empirical (black box models), the fundamental (white box models or first principle models obtained from balance equations) or the grey box (developed from combining the two previous approaches) approach can be used to design such model. Although, a large class of linear and nonlinear models exist and can be used in MPC, the simplest model that gives enough accurate predictions should be used. Indeed, in order to get reliable predictions, it is not always necessary to include all of the physical, chemical and internal behaviour of the process into the constructed model; but rather to include only the dynamics that affect the predictions. Reduction techniques like singular perturbations could be

used to derive a more simpler model from a rigorous model that can retain the basic dynamic behaviour of the full-scale model [52]. This approach has been successfully applied in chemical engineering by Duchêne and Rouchon [53]-[54].

One of the strong characteristics of the model predictive control strategy is its indifference toward the adopted modelling approach used in the construction of the predictor; of course, this remains valid as long as the constructed model is precise. This fact has ensured that a wide variety of modelling techniques have been successfully implemented within the predictive control scheme. We could mention the Artificial neural networks approaches [18, 55, 56], fuzzy modelling techniques [25, 47, 57, 58], Hammerstein and Wiener models [59-62]. More linear and nonlinear modelling techniques are given in [63, 64].

If the constructed model used to generate the predictions has a linear form, the optimization problem (1.2) is a minimization of a convex quadratic function that has a unique and a global minimum, and for which a solution could easily be obtained analytically especially if no inequality constraints are imposed. When these constraints are present, the solution could be easily obtained using quadratic programming techniques. This is very convenient for the on-line applications. However, when the prediction model is nonlinear, the situation is more complex. Due to the nonlinear relation between the predicted process outputs and the control sequence, the optimization problem (1.2) becomes nonlinear and non-convex. The analytical approach to solve this problem is generally unfeasible even if no additional inequality constraints are present [26, 28], while the numerical optimization algorithms could easily be trapped in local minima [33]. Moreover, it is difficult to estimate the time needed to solve the optimization problem, and it is not guaranteed that the obtained solution is the global optimum. Of course, nonlinear optimization requires complex and time consuming procedures [17-19]. In order to simplify the optimization problem, the most practical MPC algorithms are still using linear models [20, 26].

2.3. Constraints

One of the major advantages of the MPC scheme is its direct and systematic ability to handle constraints. Indeed, the majority of practical processes have limitations imposed on their variables that must be included, as constraints,

in the optimization problem. Depending on the nature of these limitations, different types of constraints could be distinguished. Mainly, we could encounter:

- Constraints on the control inputs values:

$$u_{\min} \le \hat{u}(t+j \mid t) \le u_{\max} \quad \text{j=0,…,} N_u - 1 \tag{1.3}$$

- Constraints on the increments inputs:

$$\Delta u_{\min} \le \Delta\hat{u}(t+j \mid t) \le \Delta u_{\max} \quad \text{j=0,…,} N_u - 1 \tag{1.4}$$

- Constraints on the values of the system outputs:

$$y_{\min} \le \hat{y}(t+j \mid t) \le y_{\max} \quad \text{j=1,…,} N_p \tag{1.5}$$

Other types of constraints could also be considered such as the system outputs variation, imposing constraints for a sub window within the prediction horizon, or instead of using two-sided constraints (known also as a band constraints or range constraints), the special case of one sided constraints could be used:

$$\hat{y}(t+j \mid t) \le y_{\max} \quad \text{j=1,…,} N_p$$

These constraints originate mostly from physical limitations, safety reasons, economic and environmental objectives. Input constraints, usually considered as hard constraints that cannot be exceeded, are the physical limitations presented by actuators with a limited range of action and a limited slew rate. On the other hand, constraints on the output are of technological nature and can physically be exceeded [26], although doing so, could lead to incorrect product specifications or even cause damage to the equipment. Nevertheless, these constraints could be treated as ‘soft constraints’; this means that a temporary violation of the constraints is possible but under severe and special conditions.

Now, let us reformulate the optimization problem (1.2), to include all the elements found in a general MPC problem:

$$\min_{\Delta\hat{u}(t)} J\left(\Delta\hat{u}(t),\hat{y}(t),w(t)\right)$$

subject to : the constraints (1.3), (1.4) and (1.5) (1.6)

where $J\left(\Delta\hat{u}(t),\hat{y}(t),w(t)\right) = \sum_{j=1}^{N_p}\left\|\hat{y}(t+j\,|\,t)-w(t+j)\right\|^2 + \lambda\sum_{j=1}^{N_u}\left\|\Delta\hat{u}(t+j-1|\,t)\right\|^2$

The first step when solving the new optimization problem is to try to translate all the linear inequality constraints ((1.3), (1.4) and (1.5)) into inequalities concerning the decision variables $\Delta\hat{u}(t+j\,|\,t)$. If the prediction model is linear, the resulting inequalities will remain linear. This property is going to be crucial if a solution to the optimization problem were to be found quickly and efficiently. The admissible solutions (feasible solutions) set of the constrained optimization problem could become empty (unfeasible problem) if it is not possible to simultaneously satisfy all the imposed constraints. The practical implementation of MPC algorithms requires a special care to avoid getting in this undesirable situation.

Even if the prediction model is linear, the mere presence of the constraints will usually result in a nonlinear control law. The MPC controller behaves linearly as long as the system is operating far from the constraints and nonlinearly when the constraints are approached [12].

3. Linear Model Predictive Control

The first generations of model predictive algorithms and most commercial MPC programs use a linear prediction model. Depending on the type of the prediction model, several strategies have been proposed. The dynamic matrix control (DMC), introduced by Cutler and Ramaker [3], was one of the first successful implementation of the MPC control scheme in the industry. It uses a discrete step response as a prediction model and can handle the constraints only in an approximate way. This problem was solved by introducing the quadratic dynamic matrix control (QDMC) in 1986 by Garcia and Morshedi [6]. Model algorithmic control (MAC), also known as model predictive heuristic control (MPHC), is another approach that uses an impulse response model and is almost identical to the DMC strategy [10, 26]. The DMC and the MAC are both considered

to be the first generation of MPC algorithms [12], however, their use is limited to stable processes.

Richalet [65] has introduced the predictive functional control (PFC) for the case of fast processes. This approach uses a state space representation of the process and can handle nonlinear dynamics and unstable linear internal models. One of the most famous and successful MPC strategy, both in the industry and the academia, is known as the generalized predictive control (GPC). It was first proposed by Clarke et al. in 1987 [7, 8]. The GPC uses a process model, in the form of discrete transfer function (or difference equations), that allows the use of a wider class of disturbance models. In addition to these strategies, other linear approaches were proposed [10, 12, 26].

In the next sections, the formulation of the GPC algorithm, for the siso case, is given.

3.1. Prediction

The key idea is to formulate the prediction equation using the superposition principle into a separately free output component and a forced one that explicitly depends on future inputs.

The GPC formulation is based on the use of the CARIMA (Controlled Auto-Regressive Integrated Moving-Average) model, which is defined, for a single-input single-output (SISO) process, by:

$$A\left(z^{-1}\right)y(t) = B\left(z^{-1}\right)z^{-d}u(t-1) + C\left(z^{-1}\right)\frac{e_G(t)}{\Delta} \tag{1.7}$$

where $e_G(t)$ is a zero mean white noise, d is the dead time of the system, $\Delta = 1 - z^{-1}$ is the differentiating operator, while A, B and C are polynomials described in the backward shift operator z^{-1} by:

$$A\left(z^{-1}\right) = 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_{n_a} z^{-n_a}$$
$$B\left(z^{-1}\right) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_{n_b} z^{-n_b}$$
$$C\left(z^{-1}\right) = 1 + c_1 z^{-1} + c_2 z^{-2} + \ldots + c_{n_c} z^{-n_c}$$

To simplify the formulation, the C polynomial is chosen to be 1.

Now let us consider the following Diophantine equation:

$$1 = E_j\left(z^{-1}\right)\tilde{A}\left(z^{-1}\right) + z^{-j}F_j\left(z^{-1}\right) \quad \text{with} \quad \tilde{A}\left(z^{-1}\right) = \Delta A\left(z^{-1}\right) \tag{1.8}$$

The polynomials E_j and F_j are uniquely defined with degrees $j-1$ and n_a respectively. To get their expressions, it suffices to divide 1 by $\tilde{A}\left(z^{-1}\right)$ and factorize the remainder as $z^{-1}F_j\left(z^{-1}\right)$. The quotient of this division will be the polynomial $E_j\left(z^{-1}\right)$.

Multiplying equation (1.7) by $\Delta E_j\left(z^{-1}\right)z^j$ we get:

$$\tilde{A}\left(z^{-1}\right)E_j\left(z^{-1}\right)y\left(t+j\right) = E_j\left(z^{-1}\right)B\left(z^{-1}\right)\Delta u\left(t+j-d-1\right) + E_j\left(z^{-1}\right)e_G\left(t+j\right) \tag{1.9}$$

which, using equation (1.8), could be written as:

$$\left(1-z^{-j}F_j\left(z^{-1}\right)\right)y\left(t+j\right) = E_j\left(z^{-1}\right)B\left(z^{-1}\right)\Delta u\left(t+j-d-1\right) + E_j\left(z^{-1}\right)e_G\left(t+j\right) \tag{1.10}$$

or as:

$$y\left(t+j\right) = F_j\left(z^{-1}\right)y\left(t\right) + E_j\left(z^{-1}\right)B\left(z^{-1}\right)\Delta u\left(t+j-d-1\right) + E_j\left(z^{-1}\right)e_G\left(t+j\right) \tag{1.11}$$

The noise terms in equation (1.11) are all in the future, thus the best prediction of $y\left(t+j\right)$ is:

$$\hat{y}\left(t+j\,|\,t\right) = G_j\left(z^{-1}\right)\Delta u\left(t+j-d-1\right) + F_j\left(z^{-1}\right)y\left(t\right) \tag{1.12}$$

where $G_j\left(z^{-1}\right) = E_j\left(z^{-1}\right)B\left(z^{-1}\right)$.

The polynomials E_j and F_j could be obtained easily by the recursion of the Diophantine equation [7, 10].

Consequently, we could estimate the future process outputs along the prediction horizon, assuming $N_1 = d+1$, as follows:

$$\begin{aligned}
\hat{y}\left(t+N_1\,|\,t\right) &= G_{N_1}\Delta u\left(t\right) + F_{N_1}y\left(t\right)\\
\hat{y}\left(t+N_1+1\,|\,t\right) &= G_{N_1+1}\Delta u\left(t+1\right) + F_{N_1+1}y\left(t\right)\\
&\vdots\\
\hat{y}\left(t+N_2\,|\,t\right) &= G_{N_2}\Delta u\left(t+N_2-1\right) + F_{N_2}y\left(t\right)
\end{aligned}$$

or, using the matrix representation, as:

$$\hat{Y} = G\Delta\hat{U} + F\left(z^{-1}\right)y(t) + G'\left(z^{-1}\right)\Delta u(t-1) \tag{1.13}$$

where:

$$\hat{Y} = \begin{bmatrix} \hat{y}(t+N_1 \mid t) \\ \hat{y}(t+N_1+1 \mid t) \\ \vdots \\ \hat{y}(t+N_2 \mid t) \end{bmatrix} \quad \Delta\hat{U} = \begin{bmatrix} \Delta u(t) \\ \Delta u(t+1) \\ \vdots \\ \Delta u(t+N_2-N_1-1) \end{bmatrix} \quad G = \begin{bmatrix} g_0 & 0 & \cdots & 0 \\ g_1 & g_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_{N_2-N_1-1} & g_{N_2-N_1-2} & \cdots & g_0 \end{bmatrix}$$

$$G'\left(z^{-1}\right) = \begin{bmatrix} \left(G_{N_1}\left(z^{-1}\right) - g_0\right)z \\ \left(G_{N_1+1}\left(z^{-1}\right) - g_0 - g_1 z^{-1}\right)z^2 \\ \vdots \\ \left(G_{N_2}\left(z^{-1}\right) - g_0 - g_1 z^{-1} - \cdots - g_{N_2-N_1-1} z^{-(N_2-N_1-1)}\right)z^{N_2-N_1} \end{bmatrix}$$

$$F\left(z^{-1}\right) = \begin{bmatrix} F_{N_1}\left(z^{-1}\right) \\ F_{N_1+1}\left(z^{-1}\right) \\ \vdots \\ F_{N_2}\left(z^{-1}\right) \end{bmatrix}$$

The first column of matrix G, that is the coefficients $g_0,\ldots,g_{N_2-N_1-1}$, represents the step response of the plant when a unit step is applied.

The last two terms in the prediction equation (1.13) are the free response of the system; they depend solely on previous control increments and inputs. Thus, this equation could be rewritten as:

$$\hat{Y} = G\Delta\hat{U} + F_{ree} \tag{1.14}$$

3.2. Control law

The typical MPC cost function is usually formulated as:

$$J\left(\Delta\hat{u}(t), \hat{y}(t), w(t)\right) = \sum_{j=N_1}^{N_2}\left(\hat{y}(t+j \mid t) - w(t+j)\right)^2 + \lambda\sum_{j=1}^{N_u}\left(\Delta\hat{u}(t+j-1 \mid t)\right)^2 \tag{1.15}$$

Using the matrix representation and substituting the expression of (1.14) in equation (1.15) result in:

$$J = \left(G\Delta\hat{U} + F_{ree} - W\right)^T \left(G\Delta\hat{U} + F_{ree} - W\right) + \lambda\Delta\hat{U}^T\Delta\hat{U} \tag{1.16}$$

This equation could be expressed as:

$$J = \frac{1}{2}\Delta\hat{U}^T H\Delta\hat{U} + b^T\Delta\hat{U} + f_0 \tag{1.17}$$

where:

$$H = 2\left(G^T G + \lambda I\right)$$
$$b^T = 2\left(F_{ree} - W\right)^T G$$
$$f_0 = \left(F_{ree} - w\right)^T \left(F_{ree} - w\right)$$

If no constraints are imposed, the minimization of this type of cost function can be analytically obtained by taking the derivative of J equal to 0. The solutions of this equation give the optimal control sequence ΔU. They are given by:

$$\Delta U = -H^{-1}b = \left(G^T G + \lambda I\right)^{-1} G^T \left(W - F_{ree}\right) \tag{1.18}$$

Inversing the matrix $\left(G^T G + \lambda I\right)$ is a computing demanding process that is proportional to its dimensionality. To reduce the computing requirement of the algorithm, the control horizon concept, in which the control actions will be made constant after $N_u \le N_2$ samples, is introduced. Using this concept the algorithm will have to inverse a matrix of $N_u \times N_u$ elements instead of $\left(N_2 - N_1\right)\times\left(N_2 - N_1\right)$.

3.3. The constrained problem

3.3.1. Formulation as a quadratic programming (QP) problem

Most practical control systems include constraints, thus, making the manner in which they are handled by the control algorithm extremely important. In the linear case, this is done by translating both constraints on the inputs and on the outputs to be related directly to the control input increment through the dynamic matrix G, as follows:

- Constraints on the values of the system outputs: from equation (1.5), we could, using equation (1.14), write:

$$Y_{V\min} \le G\Delta\hat{U}\left(t \mid t\right) + F_{ree}\left(t\right) \le Y_{V\max} \tag{1.19}$$

where $Y_{V\min}$ and $Y_{V\max}$ are vectors of $N_2 - N_1 + 1$ elements, with:

$$Y_{V\min} = \begin{bmatrix} y_{\min} \\ \vdots \\ y_{\min} \end{bmatrix} \qquad Y_{V\max} = \begin{bmatrix} y_{\max} \\ \vdots \\ y_{\max} \end{bmatrix}$$

- Constraints on the increments inputs: from equation (1.4), we have:

$$\Delta U_{V\min} \le \Delta \hat{U}(t \mid t) \le \Delta U_{V\max} \tag{1.20}$$

where $\Delta U_{V\min}$ and $\Delta U_{V\max}$ are vectors of N_u elements:

$$\Delta U_{V\min} = \begin{bmatrix} \Delta u_{\min} \\ \vdots \\ \Delta u_{\min} \end{bmatrix} \qquad \Delta U_{V\max} = \begin{bmatrix} \Delta u_{\max} \\ \vdots \\ \Delta u_{\max} \end{bmatrix}$$

- Constraints on the control inputs values: equation (1.3) will be rewritten to have the following expression:

$$u_{min} \le \hat{u}(t+j \mid t) = \sum_{i=0}^{j} \Delta \hat{u}(t+i \mid t) + u(t-1) \le u_{max} \tag{1.21}$$

We could write:

$$U_{V\min} \le J_V \Delta \hat{U}(t \mid t) + U_V(t-1) \le U_{V\max} \tag{1.22}$$

where $U_{V\min}$, $U_{V\max}$ and U_V are vectors of dimensionality N_u that have the following values:

$$U_{V\min} = \begin{bmatrix} u_{\min} \\ \vdots \\ u_{\min} \end{bmatrix} \quad U_{V\max} = \begin{bmatrix} u_{\max} \\ \vdots \\ u_{\max} \end{bmatrix} \quad U_V = \begin{bmatrix} u(t-1) \\ \vdots \\ u(t-1) \end{bmatrix}$$

while J_V is a matrix of dimensionality $N_u \times N_u$:

$$J_V = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Inequalities (1.19), (1.20) and (1.22), could easily be reformulated to obtain the following form:

$$C\Delta\hat{U}(t\,|\,t) \le \overline{c} \tag{1.23}$$

where C and $\overline{c}$ are matrix and vector known at time t [12, 66].

Using equation (1.17), the optimization problem will be given by:

$$\begin{aligned} &\min_{\Delta\hat{U}} \ \frac{1}{2}\Delta\hat{U}^T H \Delta\hat{U} + b^T \Delta\hat{U} + f_0 \\ &\text{subject to} \\ &\qquad C\Delta\hat{U} \le \overline{c} \end{aligned} \tag{1.24}$$

Equation (1.24) is a standard optimization problem, known as (convex) quadratic programming (QP) problem, which can always be solved (or shown to be infeasible) within a limited number of iterations. Its computing requirement depends strongly on the characteristics of the objective function and the number of inequality constraints [67]. One of the approaches described in the next sub-section can be used to solve, at each sampling period, this optimization problem. This approach could be easily transformed to accommodate multi-input multi-output (MIMO) processes [26, 66].

Solving a quadratic programming (QP) problem, even in quadratic form, is not a trivial task. The analytical approach is no longer viable due to the presence of constraints. Instead, the use of numerical algorithms from the group of active set methods and interior point methods is considered to be the most effective way to solve these problems [26]. The convex QP problems could also be solved by other approaches like the augmented Lagrangian methods or by means of exact penalty method [67].

3.3.2. Solving convex QP problems

Let us consider the following general form of a convex QP problem:

$$\min_{x} \quad \mathrm{q}(x) = \frac{1}{2}x^T G x + x^T d \tag{1.25}$$

Subject to

$$a_i^T x = b_i, \ \text{for all } i \in \mathcal{E}, \tag{1.26}$$

$$a_i^T x \ge b_i, \ \text{for all } i \in \mathcal{I} \tag{1.27}$$

where G is a positive semi-definite hessian matrix of $n \times n$ elements, $\mathcal{E}$ and $\mathcal{I}$ are finite sets of indices, d, x and $\{a_i\}, i \in \mathcal{E} \bigcup \mathcal{I}$ are vectors with n elements.

a) Optimality conditions

We define the active set $\mathcal{A}(x)$ at any feasible solution x as the union of the set $\mathcal{E}$ with the sub-set of the indices of the active inequality constraints, or as:

$$\mathcal{A}(x) = \mathcal{E} \cup \left\{ i \in \mathcal{I} \mid a_i^T x = b_i \right\} \tag{1.28}$$

The optimal active set $\mathcal{A}(x^*)$ is defined as the active set at the optimal solution x^*.

Suppose that the solution x^* is a local solution of (1.25)-(1.27). Then there must be a Lagrange multiplier vector λ^* with component $\lambda_i^*, i \in \mathcal{E} \bigcup \mathcal{I}$, so that the following necessary and sufficient conditions are satisfied:

$$\nabla_x \mathcal{L}\left(x^*, \lambda^*\right) = 0 \tag{1.29}$$

$$a_i^T x^* = b_i, \quad \text{for all } i \in \mathcal{A}\left(x^*\right) \tag{1.30}$$

$$a_i^T x^* > b_i, \quad \text{for all } i \in \mathcal{I} \setminus \mathcal{A}\left(x^*\right) \tag{1.31}$$

$$\lambda^* \geq 0, \quad \text{for all } i \in \mathcal{I} \bigcap \mathcal{A}\left(x^*\right) \tag{1.32}$$

with:

$$\mathcal{L}\left(x, \lambda\right) = q(x) - \sum_{i \in \mathcal{E} \bigcup I} \lambda_i \left(a_i^T x - b_i \right) \tag{1.33}$$

Consequently:

$$\nabla_x \mathcal{L}\left(x^*, \lambda^*\right) = G x^* + d - \sum_{i \in \mathcal{A}\left(x^*\right)} \lambda_i^* a_i \tag{1.34}$$

These equations are obtained after simplifying the first order necessary conditions for optimality, often referred to as the Karush-Kuhn-Tucker conditions, or KKT conditions [67].

As a consequence of (1.25)-(1.27) being convex QP problem, any local solution is also the global solution, and the KKT conditions (1.29)-(1.32) are in fact necessary and sufficient conditions for the global solution.

The QP formulation given by (1.25)-(1.27) is the same as the one given by (1.24). The constant f_0 has been dropped out because it has no influence on the solution and the equality constraints (1.26) have been presented explicitly.

To solve the optimization problem (1.25)-(1.27), two major approaches are considered to offer the best performances: the active set methods successively used since 1970s, and the more recent interior point (IP) methods. Generally, the active set approaches are most effective for small to medium scale problems whereas the IP methods have been shown to be effective for large scale QP problems [67]. More details related to these approaches can be found in [67-69].

b) Active Set Methods

the active set methods start from an initial solution x_0 and an initial working set $\mathcal{W}_0$ that contains all the equality constraints $i \in \mathcal{E}$ in addition to some (but not necessary all) of the active constraints, then converge towards the optimal active set $\mathcal{A}(x^*)$ and the optimal solution x^* by iteratively solving equality QP sub-problems. Using the gradient and the Lagrange multipliers information, the solution x_k at the kth iteration and the working set $\mathcal{W}_k$ are continuously updated until the optimal active set $\mathcal{A}(x^*)$ and the optimal solution x^* are obtained.

We could distinguish three varieties for active set methods known as *primal*, *dual* and *primal-dual* [67]. In this section, the discussion is limited to the primal methods for which the generated solutions along the iterations remain feasible with respect to the primal problem (1.25)-(1.27). The general formulation of the algorithm is outlined in the next paragraphs.

Suppose we have at the kth iteration the working set $\mathcal{W}_k$ with the feasible solution x_k. The first step is to check, using the simplified KKT conditions, if this solution minimizes the QP (1.25)-(1.27) in the subspace defined by the working set. If not, a new candidate solution is generated by computing a step p obtained by solving a QP sub-problem in which the constraints in the working set $\mathcal{W}_k$ will be enforced as equality constraints while the inactive inequality constraints are disregarded. Let us define the following:

$$p = x - x_k, \qquad g_k = Gx_k + d$$

Replacing x by his expression in the objective function of (1.25) will result in:

$$\mathrm{q}(x) = \mathrm{q}(x_k + p) = \frac{1}{2} p^T G p + g_k^t p + c$$

Where $c = \frac{1}{2} x_k^T G x_k + d^T x_k$ is a constant term.

By dropping the constant term c, the QP sub-problem could be formulated as:

$$\min_p \quad \frac{1}{2} p^T G p + g_k^t p \tag{1.35}$$

Subject to

$$a_i^T p = 0_i, \quad \text{for all } i \in \mathcal{W}_k \tag{1.36}$$

Let p_k be the solution to this QP sub-problem.

Whatever the value of p_k, the constraints in $\mathcal{W}_k$ will also be satisfied with x for all $i \in \mathcal{W}_k$ because $a_i^T x_k = b_i \Leftrightarrow a_i^T (x_k + p_k) = b_i$ since $a_i^T p_k = 0,$ for all $i \in \mathcal{W}_k$. In fact, this remains true even when $x = x_k + \alpha p_k, \ \forall \ \alpha \in [0,1]$.

The step p_k can be computed using one of the standard approaches to solve the equality constraint QP (1.35)-(1.36) (see section 16.2 of [67] for an overview), like the one based on the direct solution of the corresponding KKT system:

$$\begin{bmatrix} G & -A_{\mathcal{W}}^T \\ A_{\mathcal{W}} & 0 \end{bmatrix} \begin{bmatrix} p_k \\ \lambda_k^* \end{bmatrix} = \begin{bmatrix} -g_k \\ 0 \end{bmatrix} \tag{1.37}$$

where $A = [a_i]_{i \in \mathcal{W}_k}^T$ is the Jacobian of constraints in the working set and λ_k^* is the vector of Lagrange multipliers for p_k.

If the step p_k is non zero, the next feasible solution x_{k+1} will be computed as:

$$x_{k+1} = x_k + \alpha p_k \tag{1.38}$$

where α_k is used to ensure that the new solution x_{k+1} satisfies the inactive inequality constraints of (1.27). Its value is given by:

$$\alpha_k \overset{\text{def}}{=} \min\left(1, \min_{i \notin \mathcal{W}_k, a_i^T p_k < 0} \frac{b_i - a_i^T x_k}{a_i^T p_k}\right) \tag{1.39}$$

If $\alpha_k < 1$, at least one of the inactive inequality constraints not in $\mathcal{W}_k$ is blocking the step. As a result, a new working set $\mathcal{W}_{k+1}$ is constructed by adding one of the blocking constraint to $\mathcal{W}_k$.

The iterations are continued in this manner, adding inactive constraints to the working set when necessary until the step $p = 0$, that is the current solution $\hat{x}$ minimizes the objective function (1.25) over the current working set $\mathcal{W}$. At this point, we examine the Lagrange multipliers corresponding to the inequality constraints in the working set. If they are all non-negative, the optimal solution to the original QP problem (1.25)-(1.27) is found to be the current solution $\hat{x}$. If one or more constraints have negative multipliers, the constraint corresponding to the most negative multiplier is generally removed from the working set to permit further decrease of the objective function. The algorithm continues iterating by solving the QP sub-problem (1.35)-(1.36).

The active set approach for convex QP could be summarized in the following algorithm [67]:

Algorithm 1.1

Compute a feasible starting point x_0;
Set $\mathcal{W}_0$ to be a subset of the active constraints at x_0;
for $k = 0, 1, 2, \ldots$
    Solve (1.35)-(1.36) to find p_k;
    if $p_k = 0$
        Compute Lagrange multipliers λ_i ($i \in \mathcal{W}_k$) that satisfy $\nabla_x \mathcal{L}(x_k, \lambda_k) = 0$;
        if $\lambda_i \geq 0$ for all $i \in \mathcal{W}_k \cap \mathcal{I}$;
            Stop with solution $x^* = x_k$;
        else
            set $j = \arg\min_{j \in \mathcal{W}_k \cap \mathcal{I}} \lambda_j$;
            $x_{k+1} = x_k$;
            $\mathcal{W}_{k+1} = \mathcal{W}_k \setminus \{j\}$;
    else // $(p_k \neq 0)$
        compute α_k from (1.39);

$x_{k+1} = x_k + \alpha_k p_k$;
if there are blocking constraints
obtain $\mathcal{W}_{k+1}$ by adding one of the blocking constraints to $\mathcal{W}_k$;
else
$\mathcal{W}_{k+1} = \mathcal{W}_k$
end for

The speed of the active set methods for a given QP problem depend greatly on the quality of the initial feasible solution and the method used to solve the KKT system (1.37). Within the context of linear MPC, the active set methods or any other optimization approach for that matter could be enhanced by exploiting the special structure of MPC problems. This special structure is essentially due to two features [12]:

4) The resulting QP in the formulation of the MPC problem could be sparse with a particular ordering of the variables.

5) A very good initial feasible solution for the current sampling time could be obtained from the previously calculated solution.

The active set algorithm start iterating from an initial feasible solution. If the proposed solution is of good quality, the algorithm will require only a few iterations to converge towards the optimal solution. A good initial solution in an MPC based QP problem could be generated using the solution found in the previous sampling time $t-1$, that is $x_0 = \hat{U}(t-1|t-1)$ or by shifting this solution by one sampling period and adding a zero in the last term as follows: $x_0 = \{\hat{u}(t|t-1), \hat{u}(t+1|t-1), \ldots, \hat{u}(t+N_u-1|t-1), 0\}$. This procedure remains accurate as long as the disturbances and the reference trajectory changes, if any, remain relatively small. Some other techniques to generate initial solutions could be found in [10, section 7.3]. In the case where no prior information about feasible solutions is available, specialized algorithms capable of generating either initial solutions or determining that the problem is infeasible exist. One of such approaches is the Phase I method in which the feasible solution will be generated by solving a linear optimization problem or the Big M approach in which a term that measures and penalizes the infeasibility, due to constraints violation, is added to

the objective function and the feasible solution is generated by solving the newly resulting QP problem [12, 70].

When formulating the linear MPC based algorithm, the resulting QP problem could be sparse. Wright [71] has pointed out that due to the structure of the predictive control problem, the use of a banded matrix when solving the KKT system (1.37) could be done and may be more advantageous, but at the cost of introducing many more variables into the problem. Maciejowski [12] has used this approach and proposed an approximate formula to determine whether the banded scheme is more beneficial than the original dense scheme. He has also pointed out that a careful comparison should be done for any particular application before adopting either one of the above mentioned approaches. Moreover, the algorithm could be *warm* started using solutions obtained in previous sampling periods to facilitate the optimization procedure.

The active set methods were extensively used to solve the MPC problems [29, 30, 72-74]. Nevertheless, they are not always preferred, especially for the large scale QP problems resulting from long prediction horizons formulation. In these cases, the IP methods are more appropriate.

c) Interior point methods

This approach represents the second major class of methods that are used to solve convex QP problems. It was developed as an extension to the Kamarkar's algorithm [75] for solving linear programming problems. IP methods have mainly two variants, the *primal barrier* and *primal-dual* methods. In this subsection, only a brief indication of how the *primal-dual* method works will be given.

For simplicity, we restrict our description to convex QP problems with only inequality constraints. The extension to the case of equality constraints can be done easily.

The convex quadratic programming is defined as:

$$\min_{x} \quad \mathrm{q}(x) = \frac{1}{2} x^T G x + x^T d \tag{1.40}$$

Subject to

$$Ax \geq b \tag{1.41}$$

where G is a positive semi-definite hessian matrix. The $m \times n$ matrix A and the right-hand-side vector b are defined as follows:

$$A = [a_i]_{i \in \mathcal{I}}, \quad b = [b_i]_{i \in \mathcal{I}}, \quad \mathcal{I} = \{1, 2, ..., m\}$$

Suppose that the solution x^* is a solution of (1.40)-(1.41). Then there must be a Lagrange multiplier vector λ^*, so that the following necessary and sufficient conditions, obtained after specializing the general KKT conditions for (1.40)-(1.41), are satisfied for $(x, \lambda) = (x^*, \lambda^*)$:

$$\begin{aligned} &Gx - A^T\lambda + d = 0 \\ &Ax - b \geq 0 \\ &(Ax - b)_i \lambda_i = 0, \quad i = 1, 2, ..., m \\ &\lambda \geq 0 \end{aligned}$$

By introducing the slack vector $y = Ax - b$, we can rewrite these conditions as:

$$Gx - A^T\lambda + d = 0 \tag{1.42}$$

$$Ax - y - b = 0 \tag{1.43}$$

$$y_i \lambda_i = 0, \quad i = 1, 2, ..., m \tag{1.44}$$

$$(y, \lambda) \geq 0 \tag{1.45}$$

The equations (1.42)-(1.45) could be rewritten as a constrained system of nonlinear equations and derive primal-dual IP algorithms by using a modified Newton based method to this system.

Let us define:

$$F(x, y, \lambda) = \begin{bmatrix} Gx - A^T\lambda + d \\ Ax - y - b \\ Y\Lambda e \end{bmatrix}, \quad (y, \lambda) \geq 0$$

where $Y = diag(y_1, y_2, ..., y_m)$, $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_m)$ and $e = (1, 1, ..., 1)^T$.

Let us also define a duality measure μ for a current iterate (x, y, λ) that satisfies $(y, \lambda) > 0$, as:

$$\mu = \frac{1}{m} \sum_{i=1}^{m} y_i \lambda_i = \frac{y^T \lambda}{m} \tag{1.46}$$

The *central path* C is the set of points $(x_\tau, y_\tau, \lambda_\tau)$ $(\tau > 0)$ where:

$$F(x_\tau, y_\tau, \lambda_\tau) = \begin{bmatrix} 0 \\ 0 \\ \psi e \end{bmatrix}, \qquad (y_\tau, \lambda_\tau) > 0$$

The algorithm will try to move the current point (x, y, λ) towards the point $(x_{\sigma\mu}, y_{\sigma\mu}, \lambda_{\sigma\mu})$ on the central path, where σ is a parameter of the algorithm chosen in $[0,1]$. The current point will be moved using the step obtained from:

$$\begin{bmatrix} G & -A^T & 0 \\ A & 0 & -I \\ 0 & Y & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_b \\ -\Lambda S e + \sigma\mu e \end{bmatrix} \tag{1.47}$$

where $r_d = Gx - A^T\lambda + d$ and $r_b = Ax - y - b$.

The new solution is obtained by:

$$(x^+, y^+, \lambda^+) = (x, y, \lambda) + \alpha(\Delta x, \Delta y, \Delta\lambda)$$

where α is mainly chosen to retain the inequality $(y^+, \lambda^+) > 0$.

Solving equation (1.47), at each iteration, is the most computing demanding operation within the IP methods. As the active set methods, efficient formulation resulting from the special structure of linear MPC problem could be used to reduce the general computing requirement of the algorithm.

Interior point methods require a smaller number of iterations than the active set methods [67, 76]. However, an iteration of the IP methods is more computing expensive than that of the active set methods. IP methods have also the great advantages of a relatively constant computing time per QP solution. The active set methods keep the feasibility of the solution at each iteration while the new variants of the IP methods do not ensure feasibility until the end of the search [12]. As a result, if the optimization has not been finished within the allotted period of time or was interrupted, the active set methods could generate a feasible solution while the IP methods could not. This property is very important because in many MPC

problems, the feasibility of the solution is more important than the exact optimality [12].

Starting an optimization process from feasible solutions in the vicinity of the optimum could help the algorithm become more computing efficient as less iterations are required to converge towards the optimum.

Active set methods are known to fully take advantage of this fact, while IP methods are known to have difficulties with warm starting [69, 77]. To overcome these difficulties, several warm starting strategies for the IP methods have been proposed [78-80]. Moreover, the IP methods are easy to implement compared to the active set methods.

The above mentioned advantages have made the IP methods very popular within the MPC framework especially for large scale problems [31, 32, 81-83].

4. Nonlinear model predictive control

Although linear MPC algorithms provide satisfactory performance in many applications, against highly nonlinear processes severe degradation of control performance can occur, unless the operating conditions are very close to the steady state around which the model is linearized [13]. To ensure higher control performance, nonlinear techniques that use a nonlinear prediction model must be investigated [84, 85]. In fact, in recent years, a lot of attention was given to the Nonlinear MPC (NMPC) strategies and several nonlinear control algorithms have been proposed [14-16]. Predicting the future behaviour of the system using nonlinear models, albeit more accurate, will result in a nonlinear and generally non-convex optimization problem that requires the use of complex and time consuming optimization algorithms. Seeing that it is not guaranteed to find the optimal solution when using these algorithms, the objective of most NMPC strategies is not to find the global optimal solution but is rather to find a suboptimal solution that can meet the control objectives.

Depending on how it is proceeded to obtain the suboptimal solution, Tatjewski [26] has classified the existing methods into the following main categories:

4.1. Nonlinear Model Predictive Control with Successive Linearization (NMPC-SL)

The aim is to use a more accurate nonlinear model of the system but still obtain a quadratic and convex optimization problem. This could be done by linearizing the nonlinear model, at each sampling period, around the current operating conditions, and use the resulting linear model with any LMPC strategy (DMC, GPC,…etc) [52, 86-89]. It is interesting to note that the obtained control sequence is the optimal solution of the resulting quadratic optimization problem, which is a linear approximation of the original nonlinear optimization problem. Thus, the NMPC-SL approach is a suboptimal one.

For systems that have slow dynamics or systems that operate close to certain equilibrium points for extended period of time, it is possible to further improve the NMPC-SL algorithm by not performing the linearization process at each sampling period, but rather after a predefined number of samples [12]. On the other hand, for systems that have high nonlinearities when the reference changes are quick or for dynamic transients after strong and rapid changes of disturbances, such approach could be insufficient even if the linearization process occurs at each sampling period [26].

4.2. Nonlinear Model Predictive Control with Nonlinear Predictions and Linearization (NMPC-NPL)

It is clear that the previous approach (NMPC-SL) has some limitations. Restricting the use of the nonlinear model to obtain linear approximations in each sampling period appears to be insufficient. The nonlinear model must be further incorporated within the control algorithm.

One key property of the LMPC formulation is the ability to use the superposition principle for which a system response (output) could be decomposed into separately computed free and forced components. This decomposition is necessary to facilitate solving the optimization problem analytically. Unfortunately, this principle is not applicable for nonlinear systems.

The NMPC-SL strategy uses this principle after evaluating both the free and the forced responses using the approximated linearized version of the original nonlinear system to compute the future predicted system outputs. More accurate predictions could be obtained by using the nonlinear system to evaluate the free

response component of the predictions and use the approximated linearized version for the forced component. The resulting optimization problem of the NMPC-NPL approach is the same as in the NMPC-SL case (quadratic and convex), but the way in which the predicted values of the free response component are computed differs. They are obtained by using the nonlinear model instead of the linearized system [52, 90-92].

From the computing requirement point of view, using a nonlinear model to compute the free component does not considerably affect the total amount of the required computing time in each sampling period; this component needs to be evaluated only once per a sampling period. Thus using the NMPC-NPL is generally more efficient and preferred than the NMPC-SL.

The efficacy of the NMPC-NPL is proportional to the values of the control input increments; small ones lead to a solution that is close to an optimal one even for a highly nonlinear system or when a transition to distant operating points is required as long as the control action trajectory is smooth with small increments. If quick reaction of the system that generates large control increments is necessary, the NMPC-NPL might quickly become insufficient. Several improvements that were proposed to the original NMPC-NPL are presented and explained in [26].

Although the NMPC-NPL control algorithm provides satisfactory performance in many applications, against highly nonlinear processes with a quick reaction time, significant loss of control performance can ensue. For this type of systems, nothing less than the complete use of the nonlinear model within the control algorithm appears to be sufficient. However, fully introducing the nonlinear model in the MPC formulation will also mean the loss of the convexity in the optimization problem, hence losing the well-established techniques for handling convex optimization problems and be forced to rely on nonlinear optimization techniques.

4.3. Nonlinear Model Predictive Control with Nonlinear Optimization (NMPC-NO)

The analytical solution to the nonlinear and non-convex optimization problem is the most convenient approach from the solution accuracy angle. It has the ability to solve the original optimization problem completely and determine the global optimum. However, analytical approaches are generally impossible to

derive and numerical approaches are used instead [26-28]. We could distinguish the following two major families:

4.3.1. Deterministic numerical approaches

These approaches solve the optimization problem numerically. Consequently, the resulting solutions are only an approximation of the actual solution.

Let us consider the discrete time dynamical systems augmented with algebraic equations expressed by [93]:

$$x_{t+1} = f\left(x_t, z_t, u_t\right) \tag{1.48}$$

$$g\left(x_t, z_t, u_t\right) = 0 \tag{1.49}$$

where x_t is a vector of differential states with n_x elements, z_t is a vector of algebraic states with n_z elements and $u_t \in \Re^{n_u}$ is the control vector (sequence). The two functions f and g are both assumed twice differentiable, the algebraic state z_t is uniquely determined by equation (1.49), we assume that $\frac{\partial g}{\partial z}$ is invertible everywhere, and both x_t and u_t have constant values.

Within the NMPC framework, an optimal control problem (OCP) will be formulated based on the discrete dynamic model (1.48)-(1.49) that has to be solved at each sampling period starting at the current initial state $\overline{x}_0$:

$$\min_{x,z,u} \sum_{i=t_0}^{t_0+t_p-1} L\left(x_i, \mathrm{z}_i, \mathrm{u}_i\right) \tag{1.50}$$

subject to

$$x_{t_0} - \overline{x}_0 = 0, \tag{1.51}$$

$$x_{i+1} - f\left(x_i, \mathrm{z}_i, \mathrm{u}_i\right) = 0, \quad i = t_0,...,t_0+t_p-1 \tag{1.52}$$

$$g\left(x_i, \mathrm{z}_i, \mathrm{u}_i\right) = 0, \quad i = t_0,...,t_0+t_p-1 \tag{1.53}$$

$$h\left(x_i, \mathrm{z}_i, \mathrm{u}_i\right) \le 0, \quad i = t_0,...,t_0+t_p-1 \tag{1.54}$$

The goal is to minimize the objective function (1.50) which contains a Lagrange term L (sometimes called running cost) along the prediction horizon $\left[t_0, t_0+t_p-1\right]$. The function h is assumed to be differentiable and to have the

appropriate dimension. The free parameters in this OCP are the differential state vector $x=\left(x_0^T,x_1^T,...,x_{t_0+t_p-1}^T,x_{t_0+t_p}^T\right)^T$, the algebraic vector $z=\left(z_0^T,z_1^T,...,z_{t_0+t_p-1}^T\right)^T$ and the control vector $u=\left(u_0^T,\mathrm{u}_1^T,...,\mathrm{u}_{t_0+t_p-1}^T\right)^T$. The constant $\bar{x}_0$ is not a variable of the OCP problem (1.50)-(1.54), but rather a parameter upon which the OCP depends via initial conditions. The differential state vector x is computed until sampling period t_0+t_p while z and u stop at the time period t_0+t_p-1.

The OCP problem (1.50)-(1.54) is a finite dimensional nonlinear programing optimization problem (NLP) that can be solved using the so called direct methods. When a continuous time dynamic system such as simple algebraic equation, ordinary differential equation (ODE) or differential algebraic equation (DAE), is used, the resulting OCP will constitute an infinite dimensional optimization problem over the function space to which u belongs. In this case, three basic classes to solve OCP problems of this form exist [94-96]:

a. Hamilton-Jacobi-Bellman partial differential equation / dynamic programming: this approach tries to compute recursively a feedback control $u^*(x,t)$, for all time t and all initial state $\bar{x}_0$, using the principle of optimality of subarcs instead of looking for the optimal control trajectory for a single given $\bar{x}_0$ at a time [95, 97]. This will lead to the Hamilton-Jacobi-Bellman (HJB) equation in the continuous time case and to a partial differential equation (PDE) in state space [95]. Since the complete solution for all t and $\bar{x}_0$ is considered at once, the HJB approach will suffer severely from the curse of dimensionality and will require a huge computing power. In practical situations, this approach could be useful only in situations where small systems are considered [95, 97] although their use, even in these situations, is limited. Nevertheless, NMPC algorithms, especially for small scale problem, were developed based on this approach [95, 98, 99].

b. Euler-Lagrange differential equations / calculus of variations / maximum principle (Indirect methods): using the necessary conditions for optimality of the infinite dimensional optimization problem and the classical calculus of variations, an explicit solution, which remains valid for the current given initial state $\bar{x}_0$ [97], of the input as a function of time $u(t)$ and not as a feedback

control law, is obtained. This solution is formulated as a boundary value problem (BVP) that has to be solved numerically [95]. The drawback with these methods is the fact that the solutions will be difficult to derive due to strong nonlinearity and instability. Moreover, the user must have significant knowledge and experience in optimal control in order to use these approaches [94].

c. Direct methods: the first two classes are often known as 'first optimize, then discretize' approaches. They solve the infinite dimensional optimization problem by formulating the control u as a continuous function and then discretize it to numerically obtain a solution. The direct methods transform the infinite OCP into a finite dimensional nonlinear programming problem (NLP) then solve this problem to obtain the discretized control u directly. Hence, this why this class is often known by the strategy 'first discretize, then optimize'.

For the online optimization of an NMPC problem, the approaches that belong to the last class are usually used [94, 95, 97, 100]. In the following sub-sections, only approaches belonging to this class are presented. More details about the first two classes and the general optimal control problem can be found in [37, 94, 95, 101, 102].

A) Solution of the NMPC problems using direct methods

Let us consider, in this subsection, the finite dimensional NMPC problem (1.50)-(1.54).

Equality constraints (1.51)-(1.53) uniquely define both variables x and z for a given control u. Expressing the variables x and z as functions $\tilde{x}(u)$ and $\tilde{z}(u)$ that satisfy (1.51)-(1.53) for all u will transform the optimization problem (1.50)-(1.54) to yield the following reduced NLP problem:

$$\min_{u} \sum_{i=t_0}^{t_0+t_p-1} L\left(\tilde{x}_i(u), \tilde{z}_i(u), u_i\right) \tag{1.55}$$

subject to

$$h\left(\tilde{x}_i(u), \tilde{z}_i(u), u_i\right) \le 0, \quad i = t_0,...,t_0+t_p-1 \tag{1.56}$$

This problem has interesting characteristics, given that its variable space has been severely reduced to include only the control variable u, this strategy is similar to the dense scheme in linear MPC. Solving this reduced problem instead

of the original problem could be more appealing. This approach is known as *sequential approach* to optimal control due to the fact that in each optimization iteration, both system simulation (to reduce the variable space) and optimization are performed sequentially, one after the other.

In contrast to the *sequential approach*, we have the *simultaneous method* in which both the system simulation and the optimization are performed simultaneously by directly considering the original NLP problem (1.50)-(1.54) and using a Newton type optimization algorithm.

Using the sequential approach will lead to an NLP problem which has a reduced variable space with less structure in the linear sub-problems than the simultaneous approach. In this method, often an off-the-shelf code for nonlinear optimization could be used [93]. This approach is used by many practitioners given that the practical implementation will be much more easy to accomplish [93]. Another advantage of the sequential approach is the continued feasibility of the solutions within the optimization iterations. Even if the optimization was interrupted or could not be finished in time, the intermediate solutions are guaranteed to be feasible and, thus, could be used at any moment. However, this is not true for the simultaneous method in which the feasibility of the solution is only ensured at the end of the optimization procedure [94, 97, 100].

When using the simultaneous approach, the full optimization problem (1.50) -(1.54) with the variables u, x and z will be considered. In this case, specially designed optimization algorithms could be used to exploit the resulting special structure in the optimization problem in order to efficiently compute the most crucial and often complex steps in Newton based algorithms [93, 97], mainly computing the derivatives and solving the subsequent sub-problems QP. As a result, even with more variables, the simultaneous approach could be more computing efficient than the sequential approach. It is also important to note that the simultaneous approach is able to deal with unstable nonlinear systems better than the sequential approach [100].

Once the NLP problem formulation has been chosen (sequential or simultaneous), the resulting NLP must be solved using dedicated optimization algorithms.

B) Optimization algorithms

The NMPC problems (1.50)-(1.54) and (1.55)-(1.56) are in fact specially structured cases of generic NLP problems that could be solved using any optimization algorithm that is dedicated to solving generic NLP problems. Although directly applying the optimization algorithms to the NMPC problems is possible, it is highly ill advised. The optimization algorithm should take advantage of the moving horizon formulation and fully exploit the special structure generally found in NMPC based NLP problems. Let us consider the following generic NLP problem:

$$\min_{X} \; F(X) \tag{1.57}$$
subject to:
$$G(X) = 0$$
$$H(X) \le 0$$

Under mild assumptions, we could state that for any local solution X^* to the NLP problem (1.57), there exist multiplier vectors λ^* and μ^* such that the following necessary and sufficient conditions (KKT conditions) hold:

$$\nabla_X \mathcal{L}(X^*, \lambda^*, \mu^*) = 0 \tag{1.58}$$
$$G(X^*) = 0 \tag{1.59}$$
$$H(X^*) \le 0 \tag{1.60}$$
$$\mu^* \ge 0 \tag{1.61}$$
$$H_i(X^*)\mu_i^* = 0, \qquad i = 1,\ldots,n_H \tag{1.62}$$

where:

$$\mathcal{L}(X, \lambda, \mu) = F(X) + G(X)^T \lambda + H(X)^T \mu$$

The Newton based optimization algorithms will then try to locate a local solution by using successive linearization of the problem functions in (1.58)-(1.62) and looking for points that satisfy these conditions. Two big families of Newton based algorithms could be distinguished depending on how to treat the inequality constraints (1.60)-(1.61); we have sequential quadratic programming (SQP) and the nonlinear interior point (NIP) methods.

In the sequential quadratic programming, all of the nonlinear functions within the resulting KKT system will be linearized. The resulting linearized KKT system is equivalent to the KKT system of the following QP:

$$\min_{X} \; F_{QP}^{k}\left(X\right) \tag{1.63}$$

subject to:

$$G\left(X^{k}\right)+\nabla G\left(X^{k}\right)^{T}\left(X-X^{k}\right)=0$$

$$H\left(X^{k}\right)+\nabla H\left(X^{k}\right)^{T}\left(X-X^{k}\right)\leq 0$$

where:

$$F_{QP}^{k}\left(X\right)=\nabla F\left(X^{k}\right)^{T}X+\frac{1}{2}\left(X-X^{k}\right)^{T}\nabla_{X}^{2}\mathcal{L}\left(X^{k},\lambda^{k},\mu^{k}\right)\left(X-X^{k}\right)$$

If the hessian matrix $\nabla_{X}^{2}\mathcal{L}\left(X^{k},\lambda^{k},\mu^{k}\right)$ of (1.63) is positive semi-definite, then this QP is convex and the global solution can reliably be found by using one of the optimization methods previously described in section 3.3.2. The solution to this QP problem will provide the required step ΔX^{k} to generate the point $X^{k+1}=X^{k}+\Delta X^{k}$ for the next iteration. In this SQP version, both hessian and Jacobian matrices are exactly obtained. However, the more widely used SQP variants are based on inexact approximation of them like the Powell's classical SQP methods [103] in which an approximation of the hessian matrix, updated in each optimization iteration, is used. Another variant, like the one given in [104], uses a reduced hessian approximation that approximates only the portion of the Hessian matrix relevant at the time, making the algorithm more computing efficient. More SQP variants could be found in [105].

In addition to the SQP algorithm, we could use a nonlinear interior point (NIP) algorithm to solve the KKT system (1.58)-(1.62) by replacing the last non-smooth condition by a nonlinear approximation and using one of Newton's methods. This approach uses the same concepts the IP method for convex QP does. The only difference is that a linearization of all problem functions is performed in each iteration. More details can be found in [69].

Several NMPC based algorithms were developed using the SQP methods [106-108] and the NIP approach [32, 83].

C) Classification of direct methods

According to the adopted strategy at each optimization step, several variants of Newton based optimization algorithms could be built. Mainly, the following strategies can be envisaged:

(a) NLP formulation: sequential or simultaneous

(b) Treatment of inequalities: SQP or nonlinear IP

(c) Derivative computation: full or reduced

(d) Linear algebra of linearized sub-problems: banded (sparse) or condensing (state elimination).

Based on these choices, we could distinguish between several algorithm variants like the classical single shooting method [109] which could be classified as (Sequential, SQP, Reduced) or as (Sequential, SQP, Full, Condensing) depending on whether the reduced derivatives were used, and the classical reduced SQP collocation methods, like the one used in [110], which could be classified as (Simultaneous, SQP, Full, Condensing). An exhaustive review of different optimization methods is given in [37].

Nonetheless, several problems have been reported concerning the implementation of deterministic numerical approaches. Numerous methods are initial conditions sensitive; suitable initial guesses must be provided to start the iterative process and to ensure converging toward the global optimum [33-35]. Of course, obtaining a suitable initial guess is not a trivial matter especially as the complexity of the system grows [111]. For this reason, these approaches are sometimes called local methods [33]. Moreover, these approaches could only be used when a mathematical based model of the process is available like in the case of fundamental models. These approaches could not be used with some empirical models. In fact, even if a mathematical based model is available, derivative information, nonetheless, is sometimes hard, impossible to get, or simply not available [36]. An additional drawback is the fact that these approaches could only be used with continuous variables, they cannot handle discrete problems [37].

4.3.2. Stochastic numerical approaches

Another approach to solve the nonlinear and generally non-convex optimization of the NMPC problem is the use of the stochastic meta-heuristics algorithms, abbreviated in this dissertation to meta-heuristics algorithms to simplify the notation. These algorithms are based on fundamental elements and procedures that produce evolution and intelligent behaviours in natural systems [38] or on physical principles [39], and are capable of handling the majority of optimization problems [39, 40]. These methods have known huge success, both in academia and the industry [39, 41]. This success could be attributed to the facts that the meta-heuristic algorithms are simple, flexible, derivative-free, generally able to handle local optima and generate high quality solutions in a reasonable period of time [39, 42-44]. In fact due to their heuristic and random natures, these algorithms have a better-searching capability than that of the classical analytical approach [45], are unaffected by initial conditions and can handle both discrete and continuous problems.

Numerous meta-heuristic variants exist. We could mention: genetic algorithm (GA) [112, 113], Differential Evolution (DE) [114, 115], Particle Swarm Optimization [40, 116], Ant Colony Optimization (ACO) [117, 118]. Some of the more recent variants are the: Artificial Bee Colony (ABC) [119, 120], Gravitational Search Algorithm (GSA) [121, 122], Kinetic Gas Molecule Optimization (KGMO) [123] and Grey Wolf Optimizer (GWO) [42]. In fact, meta-heuristic is still an active research field where enhancements of current variants and completely new algorithms are continuously proposed. A good review about the different meta-heuristic approaches could be found in [39, 124].

These algorithms were successfully applied to solve multiple engineering problems in diverse fields including the predictive control where at each sampling period, a control action is computed by solving an NLP optimization problem.

Let us reconsider the generic NLP problem of (1.57). If X has a limited number of possible values, then the ideal approach to solve this problem, from the accuracy and simplicity angles, is to evaluate the objective function $F(.)$ for all possible X and determine the value(s) that give(s) the minimum value of the objective function and fulfils the imposed constraints. However, in most realistic

problems, the admissible set of (1.57) tend to be at the least large enough to make this approach irrelevant except for the most trivial problems. The meta-heuristic algorithms are based on this concept, but instead of evaluating all possible values of X, only a limited number of strategic evaluations will be performed with the hope of locating a solution of reasonable quality.

More details about how to use the meta-heuristic algorithms to solve the NMPC problem will be given in chapters 2 and 3.

Although their success, the meta-heuristic algorithms are not perfect optimization techniques. In fact, several drawbacks could be enumerated like the necessity to tune several parameters of the algorithm before its use [124, 125]. These parameters, which have not universal optimal values, have great influence on the efficiency and the effectiveness of the algorithm [39]. The meta-heuristic based algorithms are generally able to generate high quality solutions in a reasonable amount of time, but this does not mean that they will guarantee this fact or the fact that the generated solutions are in the neighbourhood of the global optimum.

5. Efficient ways to reduce the NMPC on-line computing requirement

Implementing on-line NMPC algorithms requires a huge computing power which can limit their use in fast-sampling systems and real time applications. To reduce the computing requirement of these algorithms, several approaches taking advantages of the special structure of NMPC formulation were proposed. Bemporad, Morari [126] have proposed the explicit MPC for linear models where the optimization problem is pre-computed off-line for a given range of operating conditions of interest. In their work, they exploited the multi-parametric programming techniques to express the optimal control as an explicit function of the state and the reference vectors reducing the online computations to a simple function evaluation. This function is usually piecewise affine (PWA) and the controller is mapped into a lookup table of linear gains [127]. This approach has attracted the interest of the research community and was extended to nonlinear MPC [16, 128-130] where the computing requirements are even more significant. The major challenge that has limited the applicability of the explicit MPC is that the numbers of entries in the lookup table increase exponentially with the number of

decision variables [15, 34]. Moreover, the nonlinear constraints are represented using piecewise affine approximation. Consequently, the entries in the lookup table will also increase if more approximation accuracy is required [15]. Hence, the explicit MPC use is limited to small problems with low dimensions [15].

Another approach is based on the work of Zheng and Zhang [131] where the authors have proposed a control algorithm in which the first control move was exactly calculated while the rest of the control moves were approximated, given that only the first control action is to be applied to the system. Their approach should significantly reduce the online computing requirement as the considered NMPC optimization problem will always have a control horizon of one move regardless of the original control horizon.

The computing requirement needed to solve an optimization problem depends on the number of constraints. However, only constraints that can become active can influence the optimization result. The other remaining superfluous constraints, which can never be violated, have no influence on the optimization result; therefore, they can be eliminated from the optimization problem without any repercussion. The resulting simpler equivalent optimization problem will have fewer constraints and could be solved using less computing power. Several methods that can determine the superfluous constraints in an optimization problem were proposed. However, their computing requirement is quite important. For more details about constraints reduction, see [10].

Patrinos, Sopasakis [76] have proposed a reformulation of the strictly convex QP arising in constrained LMPC as a system of piecewise affine equations. They have shown that the resulting linear system that needs to be solved at each iteration is positive semi-definite and has a significantly smaller dimension than that of the original problem. This system showed considerable merit when applied to MPC over standard active set or interior point algorithms. In fact, they have claimed that the proposed algorithm is orders of magnitudes faster than the state-of-the-art QP solvers especially for large-scale problems and long horizons. Several implementations on different systems of various dimensions and prediction horizons have been given in their paper.

This is by no means a complete survey or a thorough investigation of techniques that reduce the on-line computing requirement of NMPC optimization problems. It is a simple presentation of some techniques. In this thesis, we are only interested in approaches that tackle and solve the original nonlinear and generally non-convex optimization problem completely on line.

6. Feasibility issues

Solving a constrained optimization problem does not necessarily mean that a solution to this problem exists. In fact, in the presence of certain conditions, it will be impossible to satisfy simultaneously all of the imposed constraints, especially those on the output. In this case, the optimization problem is said to be infeasible.

Within the NMPC framework, the infeasibility of the optimization problem means that no control action has been calculated for the next sampling time. This situation is unacceptable for an on-line controller; a control action has to be applied at each sampling period. Therefore, the infeasibility issue should be either avoided by making sure that the original optimization problem (or a more relaxed form) remains feasible regardless of the current situation or by introducing a back-up control strategy to take over until the feasibility has been recovered.

Infeasibilities within the NMPC framework, are due to multiple causes [10]. Instances where the desired set points cannot be reached while maintaining the constraints on the inputs are one example of unobtainable control objectives. These problems arise from bad formulation and contradictory constraints and should be corrected at the design stage by putting reasonable control objectives. In other instances where perturbations or large reference changes may force variables out of their permissible sets without having the ability to return these variables to their admissible sets is another reason for infeasibilities. In addition, if the operator changes the operational variable limits during operation, current variables could be outside of the new limits and, as a consequence, the optimization problem will be unfeasible. Another reason is due to system/model mismatch which can, erroneously, let the controller thinks that it cannot satisfy the constraints if the modelling errors keep growing [12].

The infeasibility issue is more pronounced in instances where the operating objectives force the system to operate in the vicinity of the constraints in the presence of perturbations. In general, the infeasibility of the optimization problem is difficult to anticipate [12].

To avoid the situation where no control action is computed, several strategies were proposed [10]:

a. **Ad hoc measures:** when the infeasibility is detected, the controller could apply the same control output used for the previous sampling time, that is $u(t+1)=\hat{u}(t\mid t)$, or use instead the control signal $u(t+1)=\hat{u}(t+1\mid t)$. However, this approach could lead to unpredictable closed-loop behaviour [132].

b. **Disconnection of the controller:** this strategy is based on the idea of replacing the NMPC controller when the infeasibility occurred by a backup control strategy, and returning to automatic operation once the feasibility is recovered. However, the infeasibility occurs when the system is in critical operating conditions and special care should be used to re-establish normal operation to avoid catastrophic failure due to violation of safety or economic based constraints. This strategy is usually used where the infeasibility problems are not frequent.

c. **Constraints elimination:** the reason of infeasibility is the inability to satisfy, simultaneously, all the imposed constraints. So, one way to avoid this situation is to eliminate the constraint(s) causing the infeasibility and solve a more relaxed version of the optimization problem.

During normal operation, if the optimization problem becomes infeasible, the controller removes the constraints one by one starting by the relatively less important ones until the feasibility is recovered. In the upcoming sampling times, the feasibility is continuously analysed in order to reintroduce the removed constraints. This is a widely accepted approach [12] although it may pose some computing problems when used. Indeed, the optimization problem will be solved after each time a constraint is removed until the feasibility is regained. Another problem that may arise is to which level, the controller is allowed to remove constraints in order to recover feasibility. Does the controller have the authority to remove the safety based constraints? Or should it be limited to non-critical

constraints. If so, what to do in case the infeasibility will not be recovered by removing the non-critical constraints?

d. Constraints relaxation: another frequently practical approach is to soften some or all the output constraints [26]; this means that a violation of the constraints is allowed but under severe and special conditions. The easy way to soften the outputs constraints is to add new variables, called slack variables, to the cost function. These variables will heavily penalize any violation of the constraints in order to force the optimizer to violate these constraints only when the infeasibility issue arises. This approach could not be used on systems that do not tolerate any violation of the imposed hard output constraints.

e. Changing the constraints horizons: most of constraints violations occur at the first part of the control horizon due to sudden perturbations that could force the system variables outside of their permissible sets [10]. The idea is to ignore the constraints when solving the optimization problem during this first part. This approach of constraint window is adopted by some commercial MPC [10].

7. Stability

It has been shown that using a finite horizon criterion does not guarantee the closed-loop stability and that the only way to achieve it in practical implementation is to use a suitable tuning of the design parameters such as the prediction, the control horizons and the weighting matrices [133]. However, no ‘controller design procedure’ that allows determining the stabilizing prediction and control horizons for an NMPC setup based on the process model and the chosen cost function exists [134].

Using the infinite horizon approach to achieve the stability is computationally impracticable [134, 135]. Hence, the attention was directed to more practical methods with finite horizon. Most of these methods change the OCP (NLP) formulation by introducing suitable equality and inequality constraints or even adding new term to the cost function. The sole purpose of these modifications is to achieve closed loop stability of the controller regardless of the design parameters or the considered system. The introduced constraints are called stability

constraints [19, 136] and NMPC strategies based on these modifications are called NMPC approaches with guaranteed stability [134].

It is also interesting to note that optimality does not imply stability [137]. However, under mild assumptions, the feasibility of the optimization problem alone could suffice for the stability [138].

In the following paragraphs, four NMPC approaches with guaranteed stability are briefly discussed. A thorough and complete review can be found in [134].

First, let us reconsider the OCP problem (1.50)-(1.54) without the algebraic state for simplification (constraint (1.53) is removed) and with prediction and control horizons of the same length. At each sampling period, full measurement of the state is assumed to be available, no disturbances (persistent or not) act on the system and there is no model/process mismatch between the dynamical model and the physical process. The set $\Re^{n_u}$ of admissible control u is compact and convex while $\Re^{n_x}$ the set of admissible differential state x is closed and simply connected with $(0,0)$ in the interior of $\Re^{n_x} \times \Re^{n_u}$. The function L is quadratic in x and u, while f is twice continuously differentiable on $\Re^{n_x} \times \Re^{n_u}$, with the assumption that $f(0,0) = 0$ (without loss of generality). The objective of the control is to bring the system to the assumed equilibrium point of $(0,0)$.

7.1. State terminal equality constraint

This approach widely used to design NMPC strategy with guaranteed stability introduces a 'zero state terminal equality constraint' into the original OCP problem to force the state to zero at the end of the prediction horizon:

$$\tilde{x}_{t_0+t_p} = 0 \tag{1.64}$$

Introducing a single constraint to the original OCP is one of the easiest ways to establish closed-loop stability. Its theoretical framework is clear and no changes to the optimization algorithm are required (in the case when the original OCP is constrained). These facts have made this approach one of the more popular ways in establishing closed-loop stability [134]. However, adding a new equality constraint to the original OCP problem could be too restrictive and result in a

significantly reduced feasibility set especially if relatively short horizons are used. In this case, the controller does not have enough time to bring the state back to the origin. Another drawback is the computing requirement of the new constraint; it is a well-known fact that constraints increase the computing requirement to solve OCP problems.

7.2. Dual mode (Terminal constraints set)

Instead of using the restrictive constraint (1.64) to establish the stability of the closed loop, a two-steps approach is used. In the first step, the states are driven inside a terminal region Ω in the neighbourhood of the origin using a NMPC controller with variable horizons. Then, in the second step, the state is forced to the origin using a linear state feedback controller $u_t = Kx_t$ [139]. The term *dual* is used to point out the fact that the two controllers are used.

The terminal region Ω and the state feedback K are computed offline to ensure that the terminal region is a positive invariant region of attraction for the nonlinear system controller with the linear state feedback law while the input and state constraints are satisfied within this region.

The original OCP formulation is slightly changed to accommodate the dual-mode approach. The original cost function is replaced by

$$\min_{x_t, u_t, \mathrm{N}_p} J\left(x_t, \mathrm{u}_t, \mathrm{N}_p\right) \tag{1.65}$$

where N_p has been taken as an additional variable decision. The second modification concerns the following additional constraint:

$$x_{N_p|t} \in \Omega_B \tag{1.66}$$

This constraint is used to ensure that the states at the end of the prediction horizon are at the boundary Ω_B of the terminal region Ω. It is interesting to note that this constraint is less restrictive than the one given by (1.64).

Under mild assumptions, the feasibility of the new OCP problem will ensure the closed-loop stability [19, 134].

This approach is more attractive than the previous one; it has less restrictive constraint. However, its implementation is more complex due to its dual-mode control strategy.

7.3. Contractive NMPC

In this strategy [140], the following stability constraint is added to the original OCP problem:

$$\left\| x_{N_p|t} \right\|^2 \le \alpha \left\| x_t \right\|^2, \qquad \alpha \in [0,1] \tag{1.67}$$

This constraint, known as contraction stability constraint, will ensure that the magnitude of the state will be at least decreasing (contracting) by a pre-specified factor at the end of the prediction horizon.

In this approach, the predictive formulation is different. Once the input vector u_t is calculated, the entire elements of this vector will be applied to the system contrary to the conventional approach of only the first element. Thus, the next optimization problem will be solved at time $t + N_p$.

This approach is not practical, given that feasibility issues could arise in the intermediate sampling periods where the optimization problem is not solved.

7.4. Quasi-infinite horizon NMPC

In this scheme, an inequality stability constraint:

$$x_{N_p|t} \in \Omega \tag{1.68}$$

and a quadratic terminal penalty term described by:

$$\Phi\left(x_{N_p|t} \right) = x_{N_p|t}^T P x_{N_p|t} \tag{1.69}$$

are both added to the original OCP. The reasoning behind this approach was to choose the matrix P in (1.69) off line in such a way to make the objective function of the new OCP problem approximates that of an infinite horizon. In this way, the closed-loop stability can be achieved while numerically solving an optimization problem over a finite horizon [134].

The terminal region Ω and a linear state feedback controller are chosen to make Ω positively invariant for the closed loop and the nonlinear system asymptotically stable in this region.

This approach was established based on the dual-mode scheme, but instead of switching between two control strategies, the proposed approach uses only a NMPC strategy [135].

The aforementioned methods remain valid as long as the proposed assumptions are verified. However, assumptions such as no perturbation acts on the system or no model/process mismatch exists do not represent practical situations where both perturbations and model/process mismatch are actual facts. To this end, a robustness analysis of the NMPC controller is necessary to determine to which extent the previous result about stability remains valid in the presence of uncertainties.

8. Conclusion

This chapter has covered the main aspects and properties of model predictive control. In the next chapter, the meta-heuristic algorithms will be described in details.

CHAPTER 2:

META-HEURISTIC ALGORITHMS

1. Introduction

In this chapter, the basic concepts of meta-heuristic optimization algorithms and the formulation of the corresponding generic optimization problem are introduced. Mainly, Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Artificial Bees Colony (ABC) are considered. The last section of this chapter gives the details of two proposed variants of the ABC algorithm.

2. Meta-heuristic algorithms: Basics

Heuristic or approximate algorithms, used to solve optimization problems, have been present for several decades. In fact, this concept has been originally introduced in 1945 by Polya [141] and has known, since then, rapid dissemination within the research community. Several algorithms have been developed to tackle the increasingly difficult and more complex optimization problems. Variants were proposed to enhance the already existing versions, conferences and other scientific events dedicated to heuristic have started to appear. A more concise definition, as stated by Caserta and Voß [142], is that heuristic is a technique which seeks good solutions (approximates) at a reasonable computational cost.

In 1986, Glover [143] has introduced the term ‘meta-heuristic’ to designate the newly introduced, more efficient and general purpose optimizations algorithms, that are based on classical heuristic algorithms, artificial intelligence, biological evolution, neural systems and statistical mechanics [144]. The suffix ‘meta’ is a Greek word that means ‘upper level methodology’. In fact, meta-heuristics are considered to be ‘an iterative generation process which guides a subordinate heuristic, by intelligently combining different concepts, for exploring and exploiting the search spaces using learning strategies to structure information in order to find efficiently near-optimal solutions’ [144].

2.1. Classification

Meta-heuristic algorithms could be classified based on several criteria [39]:

a. Nature inspired versus non-nature inspired: the meta-heuristic algorithms could be classified depending on whether they were designed based on natural processes or on non-natural processes: Artificial immune system algorithm from biology; particle swarm, bee and ant colony optimization are inspired from swarm intelligence whereas simulated annealing for example is inspired from physics.

b. Memory usage versus memoryless: in some algorithms, the evolution of the search does not require information already collected about the search space. This approach is adopted by the simulated annealing algorithm. In contrast, tabu search and PSO, for example, use memory to save dynamically extracted information for future exploitation.

c. Deterministic versus stochastic: stochastic algorithms use some randomness during the search in contrast to the deterministic algorithms which use deterministic decisions. As such, in deterministic algorithms: starting from the same initial solution will always lead to the same final solution. However, due to the introduced randomness within the stochastic algorithms, starting from the same initial solution will typically lead to different final solutions.

d. Iterative versus greedy: in iterative algorithms, the search starts with an initial solution(s), then they iteratively try to enhance the solution(s) using some search operators in contrast to the greedy algorithms in which the search starts from an empty solution. The solution is built step by step by assigning a single decision variable of the problem in each step. Most meta-heuristic algorithms are iterative [39].

e. Population-based search versus single solution based search: population-based algorithms work with a population (numerous) of solutions. In each iteration, all of the current solutions will be manipulated to generate the next population of solutions (e.g., PSO, GA). In the single solution based approaches, only one solution will be evolved during the search process (e.g., simulated annealing). Population based algorithm are more exploration-oriented as they allow to explore the search space more effectively while single solution based

approaches are more exploitation oriented as they permit an in-depth search of local regions.

In this dissertation, only stochastic iterative population-based meta-heuristic algorithms are exclusively used. Henceforth, when the meta-heuristic algorithms are mentioned, it is implied that we are talking about the stochastic iterative population-based meta-heuristic variants.

2.2. population based meta-heuristics

In this family, the search algorithm, starting from a population of initial solutions, will iteratively try to improve the solutions using some search operators on the current population. In order to obtain a population of expectantly better solutions, a replacement strategy is carried out to replace part (or all) of the current population from the new generated population. These processes of generation and replacement will continue until a given condition is satisfied (stopping criteria). The population based meta-heuristic algorithms encompasses a very large number of algorithms, most of which are nature inspired algorithms [39]. We could mention: genetic algorithm, particle swarm optimization, Artificial Immune Systems (AIS), artificial bee colony and many other variants. These algorithms are identified based on their adopted generation/replacement strategies.

The general formulation standard to any population based meta-heuristic algorithm is given by Algorithm 2.1 [39].

Algorithm 2.1

Set the initial population P_0
$t = 0$
repeat
  Generate the new population $P_t^{'}$ // Generation process
  $P_{t+1} = \text{select-population}\left(P_t \cup P_t^{'}\right)$ // Replacement process
  $t++$
until (stopping criteria is satisfied)
output: best solution(s) found.

2.2.1. Initial population

Although the distribution of the initial population plays a critical role in the effectiveness and efficiency of the meta-heuristic algorithms, this step is often disregarded in their design [39]. Nevertheless, when considering implementing an optimization algorithm, a special care on how to generate the initial population should be taken [145].

If the initial population does not efficiently cover the search space, the optimization algorithm may not be able to locate the appropriate solution points, thereby missing the global optimum [146] and converging toward a local optimum (premature convergence).

Regardless of the used meta-heuristic algorithm, we could distinguish four strategies to generate the initial population [39]: random generation, sequential diversification, parallel diversification and heuristic initialization. Their evaluation is done based on the diversity of the initial population, the computing requirement, and the quality of the solutions. More details about these approaches could be found in [39].

2.2.2. Population size

The population size is an extremely important parameter in any population based meta-heuristic algorithms. The larger a population is chosen, the better its solutions will be, and the more computing requirement it will need. A compromise must be reached between the contradictory objectives of having good quality solutions and the low computing requirement. No definitive rule exists on how to choose the population size, although some practical design rules have been extracted depending on the used algorithms (PSO,GA, DE …etc) and the problem dimension. In the PSO case, Clerc [40] has found that a population of 20 to 30 could handle almost all of the classical optimization problems. For the evolutionary algorithms, Talbi [39] has indicated that a population between 20 and 100 is usually sufficient. Storn and Price [147] have recommended, for the DE algorithms, a population size of 5 to 10 times the dimension of the problem. Generally, the size of the population is chosen based on the problem dimension [148].

Other researchers have considered the problem from another angle. Instead of using the design rules to determine the best population size, a dynamical strategy could be used to regulate the population size on-line during the run. The algorithm could increase or decrease its population based on the current situation. This approach was used by Michalewicz [149] who introduced the concept of 'age' of a chromosome (GA). Fit chromosomes stay alive longer than the less fit ones. Using such approach, the population size will vary from an iteration to the next. Clerc [40] has described an adaptive PSO where the swarm population size is obtained through different strategies that regulate the population by removing badly performing particles and creating new ones. In [150], an adaptive DE in which the population size is adjusted based on the current search status, was used.

2.2.3. Exploitation versus exploration

Another important factor that can heavily influence the performance of the meta-heuristic algorithms is its ability to both explore (global search) the search space looking for regions of interest and to exploit (local search) these regions in order to locate optimum solutions. Ideally, an optimization algorithm that has both of these characteristics fully integrated should be designed. However, the exploration and the exploitation are somewhat exclusive characteristics.

The computing power available in the majority of practical situations is quite limited. Therefore, it is necessary to judiciously use this power. When solving an optimization problem, a certain balance between exploring the search space and exploiting its prominent regions must be established. If the exploration has not been thorough, the optimizer may miss a prominent region(s) of the search space. If this region contains the optimum solution(s), the algorithm will not be able to find this optimum or even a solution in its neighbourhood. Thus, the undesirable phenomenon of premature convergence will occur; the algorithm will converge toward a local optimum and be trapped in it. The convergence speed of the optimization algorithm could also be affected if the exploration has not been sufficiently performed as prominent regions of the search space will take more time to be, if ever, discovered.

On the other hand, once the exploration has determined a prominent region where an optimum or at least a solution of a good quality may be found, the optimization algorithm will start the exploitation process of this region by concentrating the search in its immediate neighbourhood. If this process is prematurely interrupted, the algorithm may overlook a good solution or even an optimum to converge toward a lesser solution. Consequently, and in addition to the fact that the convergence speed of the algorithm depends greatly on this process, it is necessary to give the exploitation process the required time to complete its task.

Generally, all meta-heuristic algorithms use several mechanisms to balance between these two abilities. A good balance between them is necessary to obtain an efficient optimizer [151-153].

2.2.4. Stopping criteria

Depending on the optimization problem at hand, several criteria could be used to stop the optimization procedure. We could mention [39]:

- **Static procedure**: In this method, the end of the optimization process is known a priori. This could be implemented with a fixed number of iterations, a limit on computation resources or a maximum number of objective function evaluations. This procedure is usually used when a time limit is imposed.
- **Adaptive procedure**: In this method, the end of the optimization process cannot be known a priori. This could be implemented with a fixed number of non-improving iterations or when an optimum or a satisfactory solution is obtained with a predefined error tolerance.

Some other procedures are specific to population based meta-heuristic algorithms. They generally depend on some statistics of the current population or its evolution [39]. For example, in some situations, the algorithm will be stopped if its diversity drops below a given threshold where the population will be considered stagnant. When this happens, continuing the optimization process is meaningless [39].

3. Solving optimization problems

3.1. General approach

Let us reconsider the following generic NLP problem:

$$\min_{X} \; F(X) \tag{2.1}$$

subject to:

$$G(X) = 0 \tag{2.2}$$

$$H(X) \le 0 \tag{2.3}$$

where $X \in \Re^{n_x}$ is the decision variable, F is the cost (objective) function, while G and H are the constraints functions.

Let $S \subset \Re^{n_x}$ be the set of admissible solution X that satisfy constraints (2.2) and (2.3) and n_{pop} be the population size. The basic steps needed to solve the optimization problem (OP) (2.1)-(2.3) using a population based meta-heuristic algorithm are summarized in algorithm 2.2.

Algorithm 2.2

for $h = 1 : n_{pop}$ *// Initial population*

Choose an initial solution for X_h from S

end for

Randomly choose one of the initial solutions as the best solution X_{Best}

$iter = 1$ *// Set the current number of iteration*

for $h = 1 : n_{pop}$ *// Find the best solution in current population*

if $F(X_h) < F(X_{Best})$

$X_{Best} = X_h$

end if

end for

Repeat *// Iterative process*

for $h = 1 : n_{pop}$ *// Generation/Replacement processes*

Generate a new solution X_{hNew}

Apply a replacement strategy

if replacement is necessary

$X_h = X_{hNew}$

end if

end for

for $h = 1 : n_{pop}$ *// Find the best solution in current population*

if $F(X_h) < F(X_{Best})$

$X_{Best} = X_h$

end if

end for

$iter++$ *// Set the current number of iteration*

until (stopping criteria satisfied)

At the end of the process, the optimization solution will be stored in X_{Best}.

This is a general layout of population based algorithms; it cannot accurately describe all existing population based algorithms as each algorithm has its own peculiarities.

3.2. Constraints handling

In the previous section, we have presented how to solve optimization problem using meta-heuristic algorithms with the assumption that the admissible set named S is known a priori. In practical situation, building the admissible set from constraints is not always easy or possible. In fact, even if this set could be known, several difficulties could emerge within the generation process as this set could not be invariant with respect to the generation process. As such, the strategy adopted to handle the constraints is of capital importance for the design of a good and efficient optimization algorithm. These approaches can be grouped into the following categories [39]:

3.2.1. Reject strategies

In these methods, also known as death penalty approaches, only feasible solutions are used during the search. If an infeasible solution is generated, this solution will be automatically discarded.

These approaches are only attractive when the majority of the search space is feasible. These approaches do not use infeasible solutions to gather information about global optimal solutions that can be either on the boundary between feasible and infeasible solutions or on another independent feasible region if the admissible set contains discontinuous regions.

3.2.2. Penalizing strategies

In these approaches, both feasible and unfeasible solutions could be considered. However, the original cost function will be modified to include a new term that will heavily penalize the unfeasible solutions. These are the most popular and widely used approaches to handle constraints.

3.2.3. Repairing strategies

In these strategies, custom built heuristic algorithms are used on the generated unfeasible solutions in order to transform them or 'repair them' into feasible solutions. The success of the strategy depends greatly on the efficiency of the constructed algorithm.

The problem with these methods is the extra variant computing requirement needed to repair all the generated infeasible solutions.

3.2.4. Preserving strategies

In these methods, the optimization algorithm will be slightly modified in order to ensure that each generated solution will always be feasible by incorporating problem-specific knowledge. Of course, the tailored algorithm cannot be used to solve any given optimization problem. Furthermore, feasible initial solutions have to be generated to start the algorithm which could be problematic.

The hybrid approach that combines more than one strategy can also be used to handle the constraints.

4. variants of meta-heuristic algorithm

In the hope of producing efficient algorithms, several meta-heuristic optimization algorithms have been proposed. given this huge numbers of algorithms, the question that will instinctively arise is which variant must we use to

efficiently solve our optimization problems? What algorithm has the best performances?

Contrary to some beliefs, a simple answer to this question exists. There does not exist and will not exist an optimization algorithm that outperforms all the remaining algorithms on all possible optimization problems. This statement was given based on the 'no free lunch theorem' introduced by Wolpert and Macready in [154, 155]. The authors have stated in [154]:

> «All algorithms that search for an extremum of a cost function perform exactly the same, according to any performance measure, when averaged over all possible cost functions. In particular, if algorithm A outperforms algorithm B on some cost functions, then loosely speaking there must exist exactly as many other functions where B outperforms A.»

However our interest does not include all possible cost functions. In fact, in practical situations, the number of objective functions of interest is quite limited. The idea is to evaluate the best optimization algorithms against these functions. If A outperforms B in these functions, it does not matter that B will outperforms A in other, loosely speaking, non-important functions.

4.1. Genetic algorithm

The genetic algorithm is one of the most famous and successful meta-heuristic optimization algorithms that have made a big impact within the research community. This algorithm has been originally developed by Holland [156] and his collaborators during the 1960s and 1970s. Since then, the algorithm has been the object of intensive study, exploitation and developments. The algorithm is a model of biological evolution based on the theory of natural selection [157]. Starting from an initial population, the algorithm will begin the optimization process by generating new more fit populations (generations) of chromosomes using genetic operators like crossover, recombination and mutation. The main steps of the algorithm are given by the flowchart of figure 2.1.

Each chromosome $x_h=\{x_{h1},\ldots,x_{hD}\}$ $(h=1,\ldots,n_{pop})$ is a D-dimensional vector of variables that represents a possible solution to the optimization problem with a fitness value $F_{itness}(x_h)$. Once the initial population has been generated and its

chromosomes' fitness evaluated, the algorithm start iterating through the following steps [113]:

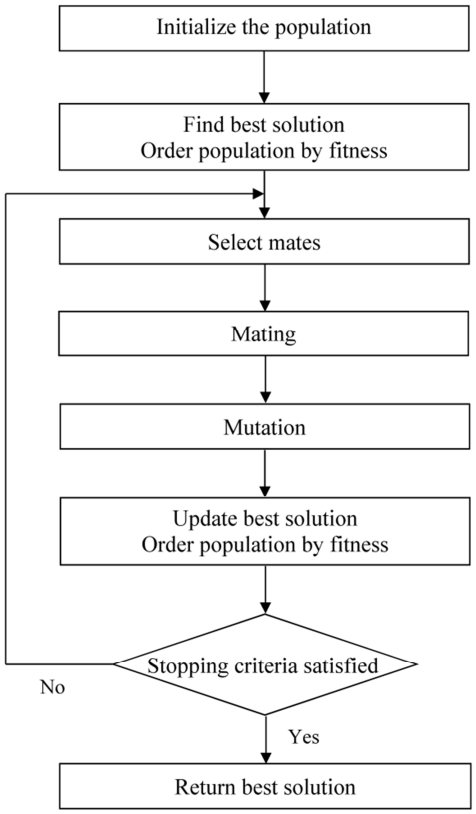

Figure : 2.1 Flowchart of the standard GA.

4.1.1. Natural selection

Based on the fitness information gathered in the last phase, the chromosomes will be sorted in a descending manner according to their fitness. Chromosomes that are fit enough will be selected to survive and possibly reproduce offspring for the next generation while the rest will die off. Of the n_{pop} chromosomes in the population, only the best N_{keep} members will be kept for mating. The remaining chromosomes will be removed to make place for the new offspring. This process will permit the population to evolve over the generations.

4.1.2. Pairing

In this stage, two parents are chosen from the surviving population to produce two offspring that contain traits from each parent. Several pairing

mechanisms exit: we could randomly choose the parents, use the roulette wheel based on their fitness or pair them from top to bottom, as the chromosomes will be ordered starting from the top of the (fitness orderly) population. More details about pairing approaches could be found in [113]. The chosen parent will also be added to the new population.

This procedure is repeated until the new population has been completely regenerated.

4.1.3. Mating

The offspring will be generated by merging the parents to pass on genetic material. The simplest method consists of choosing randomly a single or multiple crossover points in the chromosome. The first offspring will be built by copying the first parent until the crossover point, after which the second parent will be used. This procedure is inversed for the second offspring.

Let $x_f = \{x_{f1}, \ldots, x_{fD}\}$ and $x_m = \{x_{m1}, \ldots, x_{mD}\}$ be the parent, then the offspring x_1 and x_2 are given:

$$x_1 = \{x_{f1}, x_{f2}, x_{f3}, \uparrow x_{m4}, x_{m5}, \ldots, x_{mD}\}$$
$$x_2 = \{x_{m1}, x_{m2}, x_{m3}, \uparrow x_{f4}, x_{f5}, \ldots, x_{fD}\}$$

This approach of generating offspring is not attractive since no new genetic material is introduced once an initial population has been chosen. We are merely interchanging variables between chromosomes; no new variables will be added to the chromosomes in this stage.

Another more interesting method is the 'blending methods' in which the offspring are built by combining variables values of the parents as follows:

$$x_{1i} = \beta x_{fi} + (1-\beta) x_{mi}, \quad i \in \{1, \ldots, D\}$$
$$x_{2i} = \beta x_{mi} + (1-\beta) x_{fi}, \quad i \in \{1, \ldots, D\}$$

where β is a random number in $[0,1]$

This blending could be done to all variables or only to a limited number. More details can be found in [39, 113, 158]

4.1.4. Mutation

To allow the algorithm to explore other regions of the search space and escape local optima, a change or a mutation in some of the variables is randomly introduced. A parameter called mutation rate is used to determine the probability of a variable being mutated. For example, a mutation rate of 20% indicates that 1/5 of the variables in all of the chromosomes will be replaced by randomly generated values. The variables to mutate are also chosen randomly.

The algorithm will continue iterating by repeating the previous four phases until the stopping criterion has been satisfied.

Originally, GA algorithm used a binary representation as chromosomes were represented by binary strings of 0 and 1. However, this discrete representation worked well only for problems requiring solutions of low dimensionality and precision. To overcome this limitation, the concept of real coded GA was introduced [159] where a vector of real-valued genes was used to represent a chromosome. The remaining phases of the algorithm are the same as in the binary representation.

Genetic algorithm is one of the most widely used optimization algorithm in modern nonlinear optimization [157], nonetheless, it has several known deficiencies [157, 160]. Namely, its tendency to converge toward local optimum if the fitness function has not been correctly formulated, its slow convergence rate [159] and the huge computing requirement needed to find a solution. In fact, given the same problem and computation time, simpler optimization algorithms may find better solutions.

In order to overcome these issues, the balance between exploration and exploitation must be enhanced. Within the GA, the crossover operation affects decisively the exploration capability of the algorithm [159]. As such, a lot of research has been conducted on how to produce more efficient crossover operators [161-164].

4.2. Particle swarm optimization

Particle Swarm Optimization algorithm is a meta-heuristic optimization algorithm that was introduced by Kennedy and Eberhart in 1995 as a solution for

the optimization problem of a single objective continuous problem. It is based on the observation of the collective behaviour of social insects such as ants, termites and bees as well as the behaviour of other animal societies such as flocks of birds or schools of fish [165]. Basically, a number of randomly distributed particles will start surfing the search space looking for a solution to an optimization problem. Every particle represents a potential solution, and by means of a fitness function $\left(F_{itness}(.)\right)$, the suitability of the particle will be assessed and evaluated. Using these data, the particles will change their locations to try to enhance their fitness to provide more accurate solutions. This change is influenced by the history of the particle or its best position, its neighbourhood history, mainly the best position found by the whole swarm, in addition to some random behaviour.

The first step is to generate the initial population, evaluate the particles finesses and determine the global best solution P_g. Then the algorithm will start iterating by updating the positions of the particles.

The particles movements are bound by the following equations:

$$v(m+1) = v(m) + a(m+1) \tag{2.4}$$

$$X(m+1) = x(m) + v(m+1) \tag{2.5}$$

where v, a, x and m are velocity, acceleration, position and iteration index respectively. For the algorithm proposed by Clerc-Kennedy [166], the acceleration of a given particle i is given by the following expression:

$$a_i = \chi\left[c\epsilon_1\left(P_g - X_i\right) + c\epsilon_2\left(P_i - X_i\right)\right] - (1-\chi)v_i \tag{2.6}$$

where $\chi = 0.729843788$, $c = 2.05$, ϵ_1 and ϵ_2 are random numbers from the uniform distribution $U[0,1]$, x_i and v_i are the current position and velocity of particle i respectively, P_i is the personal best position of particle i, while P_g is the global best position of all particles. Note that a_i could take negative values as well as positive ones. The parameter χ guarantees a decreasing velocity for each particle as the number of iteration increases. This property will improve the convergence quality; as the particles approach the solution, their movement will be more and more limited which help fine tune the found solution.

The PSO algorithm is summarized in algorithm 2.3.

Algorithm 2.3

for each particle h
Randomly initialize $x_h,\ v_h$
$P_h = x_h$ // Initialize personal best position
end for

repeat
for each particle h
update particle position x_h using (2.4)-(2.6)
if $F_{itness}(x_h) > F_{itness}(P_h)$ // Update personal best
$P_h = x_h$
end if
if $F_{itness}(x_h) > F_{itness}(P_g)$ // Update Global best
$P_g = x_h$
end if
until (stopping criteria satisfied)
output: best solution stored in P_g .

PSO algorithm has a serious weakness that could seriously reduce its performances if no actions are to be taken. Mainly, the PSO algorithm has a problem keeping a healthy balance between exploring the search space and exploiting prominent regions of the search space [167-169]. As a direct result, the optimization algorithm would be susceptible to getting trapped within local optimum especially over multimodal, rugged, and non-separable fitness landscapes [45]. Thus, the algorithm will be vulnerable to the premature convergence problem.

A lot of research has been conducted in order to address this problem and make the PSO algorithm more efficient. Indeed, various variants of the PSO algorithm have been proposed. For example, Silva, Neves [169] have proposed a predator prey strategy in order to maintain diversity. While He, Wu [170] have introduced a passive congregation PSO (PSOPC) in which information can be transferred among particles to avoid misjudging information and becoming trapped by poor local minima, Sun, Fang [171] have proposed a quantum behaved particle

swarm optimization with Gaussian distributed local attractor point. More variants of the PSO algorithm can be found in [168, 172-175].

4.3. Artificial bee colony

The ABC is a meta-heuristic optimization algorithm that was introduced by Karaboga in 2005 [119] to solve the optimization problem in multivariable functions. It is based on the observation made on the social behaviour of the honey bee swarm.

The colony of artificial bees contains three groups of bees: employed bees, onlooker bees and scouts. The first half of the colony consists of the artificial employed bees and the second one includes the onlookers.

- **Employed bees**: the employed bee exploits a food source and advertises its position to the onlookers by dancing in the nearby hive. There is one employed bee per source.
- **Onlooker bees**: the onlookers tend to choose the best food sources to further exploit based on information communicated by employed bees. Therefore, the good food sources attract more onlooker bees compared, to the bad ones.
- **Scout bees**: When the food source is considered exhausted, it will be abandoned and its employed bee will be converted to a scout which will randomly choose a new food source to replace the old one.

The number of food sources is equal to the number of employed bees and also equal to the number of onlooker bees.

In this algorithm, the position of the food source represents a possible solution to the optimization problem while the nectar amount (quality of the food source) corresponds to the fitness of the associated solution. This fitness is usually evaluated using a given cost function. The size of the population designated by PN is the sum of the number of employed bees (SN) and the number of onlooker bees (LN). Each solution $x_h=\{x_{h1},\ldots,x_{hD}\}$ $(h=1,\ldots,SN)$ is a D-dimensional vector where D is the number of optimization parameters.

4.3.1. Basic algorithm

The main phases of the basic (original) ABC algorithm are as follows:

a. Initialization phase: The employed bees are placed on random initial food sources around the hive within the boundary of the allowed search space. The positions of these initial food sources are generated using the following expression:

$$x_{hj} = x_{\min j} + \alpha_{hj}\left(x_{\max j} - x_{\min j}\right) \tag{2.7}$$

where: $h=1,\ldots,SN$, $j=1,\ldots,\mathrm{D}$, $x_{\min j}$ and $x_{\max j}$ are respectively, the lower and the upper bound of the dimension j and $\alpha_{hj} \in [0\ 1]$ is a uniformly distributed random number. Each food source x_h is assigned to only one employed bee.

The food sources are subjected to repeated iterations of the search processes of the employed, onlooker and scout bees. The Termination criterion is chosen to be either a maximal number of iterations or achieving a specific error tolerance.

b. Employed bee phase: In this stage, each employed bee will generate a new candidate food source position v_h in the neighbourhood of its old food source position x_h according to the following expression:

$$v_{hj} = x_{hj} + \phi_{hj}\left(x_{hj} - x_{kj}\right) \tag{2.8}$$

where $k \in \{1,\ldots,SN / k \neq h\}$ and $j \in \{1,\ldots,D\}$ are uniform randomly chosen indexes, $\phi_{hj} \in [-1\ 1]$ is a uniformly distributed random number.

If a new generated position exceeds its predetermined boundary values, it is adapted to remain within the search space boundary. The fitness of the generated candidate food source v_h is evaluated and compared to the fitness of x_h. After that, a greedy selection is applied to decide which one of them to keep.

Only one optimization parameter is updated when generating the new candidate food source.

c. Probabilistic selection phase: Once all the employed bees complete their update process, they will share the nectar amount (fitness) of their food sources in the hive with the onlookers. Using this information, each onlooker bee

will randomly choose a food source to exploit with a probability value p_h. The p_h value is given by the following expression:

$$p_h = \frac{f_{itness}(x_h)}{\sum_{h=1}^{SN} f_{itness}(x_h)} \tag{2.9}$$

where $f_{itness}(x_h)$ is the fitness value of the food source x_h. Clearly, the higher the nectar amount of a given food source is, the higher its probability to be chosen by an onlooker.

d. Onlookers bee phase: Once all the onlooker bees have chosen their food sources, each one will produce a new candidate food source position v_h in the neighbourhood of the selected food source x_h using equation (2.8). Then, the greedy selection between x_h and v_h is used.

e. Scout bee phase: A food source x_h is abandoned after a given number of unsuccessful trials to produce a better food source in its neighbourhood, is exhausted; this number is denoted by *limit*. The food source which is abandoned by the employed bee is replaced with a new random food source that is generated by the scout bee. The position of this food source is obtained using equation (2.7).

The value of the *limit* parameter is given by the following expression:

$$limit = SN \times D \tag{2.10}$$

The *limit* parameter allows keeping the diversity within the ABC population by regulating the generation of the scout bees.

The location of the best ever food source discovered in the whole search space by any artificial bee is stored in x_{best}. If a given artificial bee produces a better food source, x_{best} will be replaced by its position.

The different steps of the ABC algorithm are given by the flowchart of figure 2.2.

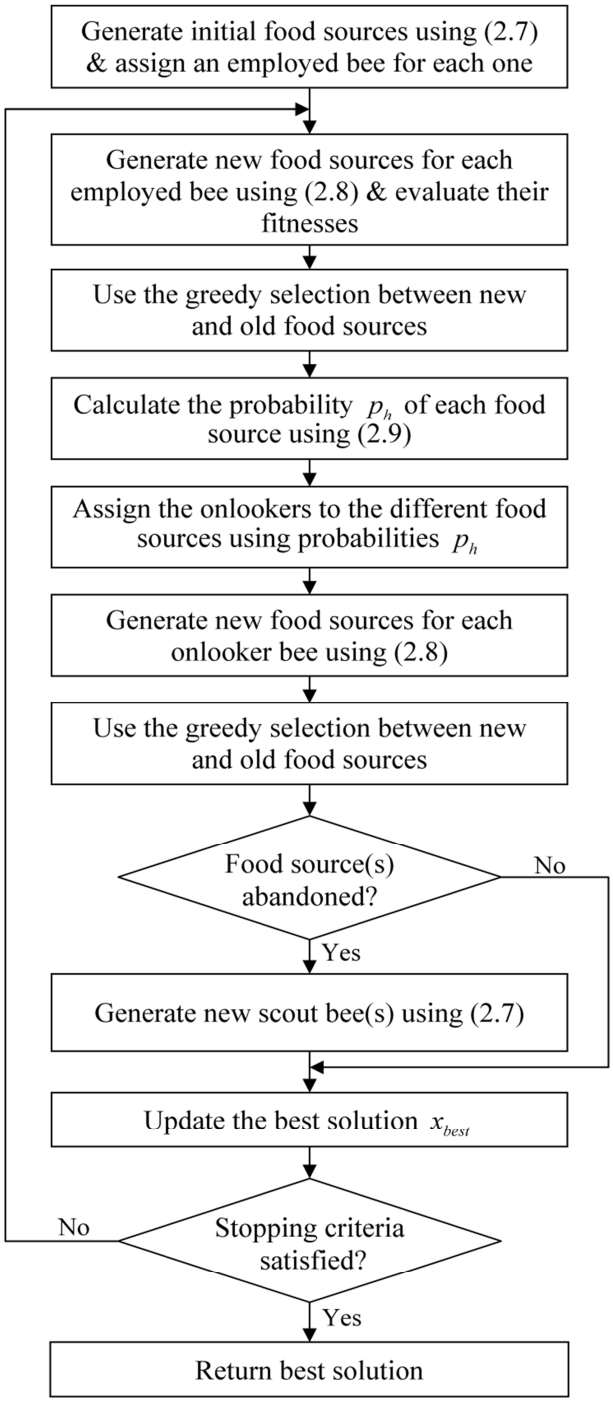

Figure 2.2 : Flowchart of the ABC algorithm.

5. Proposed variants of the ABC algorithm

Two improved variants of the ABC algorithm were proposed. They are designated by ABCEV (ABC Enhanced Version) and EEABC (Equal Exploitation ABC).

5.1. ABC Enhanced Version (ABCEV)

To enhance the overall performance of the ABC algorithm, several modifications were introduced to the original ABC algorithm. These modifications are described in details as follows:

5.1.1. Initialization phase

The distribution of the initial population in any evolutionary based algorithm is a critical task especially if no information about the solution is available. If the population does not cover the search space efficiently, it may not be able to locate the appropriate solution points, thereby missing the global optimum [146]. This difficulty may be minimized to a great extent by using a well-organized distribution mechanism rather than the more conventional random initialization.

Chaotic map has been successfully used to distribute the initial population of different evolutionary algorithms. Using such initialization will increase the diversity within the population and help to generate high quality solutions. Several versions of chaotic initialization have been considered. The 'Chaotic initialization based on logistic equation' used in [152] was found to be more efficient in both initial solutions quality and computing requirement. Hence, in this proposed version, the position of the initial food sources is generated using the following expression:

$$x_{hj} = x_{\min j} + \zeta_{hj}\left(x_{\max j} - x_{\min j}\right) \tag{2.11}$$

where h=1,…,SN, j=1,…,D and $\zeta_{h+1\,j} = 4\zeta_{hj}\left(1-\zeta_{hj}\right)$ with $\zeta_{1j} \in [0\ \ 1]$ is a uniformly distributed random number not a multiple of 0.25.

5.1.2. Update equation

Equation (2.8) encompasses a large portion of the ABC philosophy. It is responsible, in large, for both the exploratory and especially the exploitation behaviour of the algorithm. Based on a mutation strategy of the DE algorithm, Gao and Liu [176] have proposed the following equation as a replacement for equation (2.8):

$$v_{hj} = x_{best\,j} + \phi_{hj}\left(x_{r_1 j} - x_{r_2 j}\right) \tag{2.12}$$

where r_1 and $r_2 \in \{1,\ldots,SN / r_1 \neq r_2 \neq h\}$ are uniform randomly chosen indexes.

From equation (2.12), the best ever food source x_{best} is biasing the newly generated food sources towards itself, increasing thus the exploitation capabilities of the algorithm especially around the current best food source, and also increasing the convergence speed. But, as the exploitation capability of the algorithm increases, its exploration capability on the other hand decreases. To address this issue, the *limit* value which controls the scout bees was also modified.

5.1.3. Scout bee

The exploratory behaviour of the algorithm can be increased by generating more scout bees. This can be done by decreasing the value of the control parameter *limit*. Its new expression is then given by:

$$limit = 0.6 \times SN \times D \tag{2.13}$$

This expression was inspired from [177], and has proven to be efficient compared to other expressions.

5.2. Equal Exploitation ABC (EEABC)

In the previous version (ABCEV), a new update equation was used. This equation will increase the exploitation capabilities of the algorithm especially around the best ever food source x_{best}, increasing, thus the convergence speed. But, on the other hand, it will make all the artificial bees in the population converge towards the same region of the search space. Consequently, both the diversity within the population and the exploratory behaviour of the algorithm will be decreased, which will make the algorithm more vulnerable to premature convergence and to get trapped in local optima even when the *limit* parameter value is decreased.

To address this issue and further increase the convergence speed of the algorithm, the following modifications have been introduced to the ABCEV:

5.2.1. New probability equation

The original ABC uses equation (2.9) to assign onlooker bees to further exploit the different prominent regions of the search space found so far. But, as

the optimization process progresses, it will converge towards a single solution around which all the artificial bees (Employed and onlooker) will be clustered. This behaviour is extremely desirable if this solution is the global optimum as all the population will be exploiting the most prominent region of the search space. However, if this solution is not the global optimum, the population diversity is low and the algorithm will not be able to explore the remaining search space. Hence, it will converge toward this solution making the algorithm premature convergent. Scout bees, introduced to avoid converging toward local optima, are generally not able to introduce sufficient diversity within the population to escape a sub optimal solution. In fact, when a scout bee, having a low probability for attracting any onlooker bees to exploit its new source, is generated, it will be attracted to the other artificial bees clustered around the sub optimal solution. This problem will be further amplified if equation (2.12) is used.

In order to address the diversity issue and the premature convergence problem, we propose the following new probability equation as a replacement for equation (2.9):

$$p_h = (1-\alpha)\frac{f_{itness}(x_h)}{\sum_{h=1}^{SN} f_{itness}(x_h)} + \alpha \frac{\frac{1}{ExplInd(x_h)}}{\sum_{h=1}^{SN} \frac{1}{ExplInd(x_h)}} \tag{2.14}$$

where: α is a constant called *Exploration rate* and chosen in the interval $[0,1]$, $ExplInd(x_h)$ is called *Exploitation Index* of food source x_h, which represents the number of times this food source was exploited by either an employed or an onlooker bee. It is different from the number of unsuccessful trials used with the *limit* parameter. In this case, both successful and unsuccessful trials are counted. Using this parameter, we can keep track of how much the different food sources are being exploited.

Contrary to the original probability equation where the onlookers chose a food source solely based on the fitness values, the probability of an onlooker to choose a food source depends, according to (2.14), on the fitness of the food sources and their exploitation indexes. The second term in the right side of (2.14) will favour the exploitation by the onlookers of the food sources that are not being

thoroughly exploited whereas the first term will favour the exploitation of the food sources with higher fitness. Using both terms, the onlookers will not only exploit the possible prominent regions of the search space, but also the less exploited food sources maintaining a higher diversity within the population which tends to increase the exploration capabilities of the algorithm.

The *exploration rate* α is used to balance between exploiting prominent regions of the search space and exploring this space by maintaining diversity within the population. The value of α should be chosen carefully, a small one tends to make (2.14) more close to (2.9) while a value approaching 1 will increase the exploration of the algorithm but decreases severely its exploratory capabilities.

This algorithm is called *Equal Exploitation* ABC because it tends to equally exploit all the food sources not only the fittest ones. As the algorithm starts iterating, a given food source $x_{LowIndex}$ that has the lowest *exploitation index* will be more and more exploited by onlookers, which tends to increase its *exploitation index* compared to the other food sources. When the *exploitation index* of a given food source increases, the probability to further exploit it will decrease giving rise to the exploitation of other food sources with lower *exploitation index*, which ensures that all food sources get a chance to be thoroughly exploited.

The scout bees' role will also be strengthened. In fact, seeing that the food source generated by a new scout bee has not been previously exploited, it has a high probability to attract onlookers to its position which ensures that this position will get an adequate chance to be exploited. When a scout bee generates new food source, its *Exploitation Index* is immediately given the value of the lowest *Exploitation Index* found within the population.

5.2.2. Adaptive exploration rate

The value of the exploration rate α of equation (2.14) is of crucial importance; it makes a good balance between exploring the search space and exploiting its prominent regions. Determining a fixed single value for α in order to ensure good performances for any optimization problem is somewhat difficult if not outright impossible as each optimization problem has its own characteristics. Some problems require more exploration than exploitation or vice versa. In other

situations, depending on the current status of the optimization, more exploration (exploitation) is momentarily required.

The ideal situation would be to design an adaptive strategy where the exploration rate α is automatically adjusted based on details about the current status of the optimization. This is done by adopting Rechenberg's 1/5 mutation rule that states that: the ratio of successful mutation to all mutation should be equal to 1/5 in any efficient optimization process [178]. Using the value of this ratio, we could determine if more exploration (exploitation) is required.

The value of the exploration rate α is adjusted using the following formula:

$$\alpha = \begin{cases} \alpha \times 0.85 & \text{if } \varphi < 1/5 \\ \alpha / 0.85 & \text{if } \varphi > 1/5 \\ \alpha & \text{if } \varphi = 1/5 \end{cases} \tag{2.15}$$

where φ is the ratio of successful mutation to all mutations computed every $10 \times D$ iterations. The value of α is adjusted every D iterations.

The initial value for α is chosen equal to 0.1.

5.2.3. Adaptive update mechanism

In the basic ABC, each new solution is generated by updating only one optimization parameter at a time. This approach helps explore the search space more vigorously and avoid premature convergence. However, the algorithm will require more time to converge toward good solutions. In order to increase the convergence speed of the algorithm without compromising its ability to explore and escape local optima, an adaptive update strategy where one or more optimization parameters could simultaneously be updated, is developed.

The idea revolves around the fact that at the start of the optimization, the diversity within the population is relatively high, so, instead of updating only one optimization parameter at a time, multiple (not necessarily all) parameters could be updated to increase the convergence speed without worrying about the premature convergence problem. However, as the algorithm start iterating, the number of parameters to be updated will be constantly reduced as a response to the typically

diminishing diversity, until having only one optimization parameter to update at the end of the optimization process.

The number of dimensions to update, in each iteration, is given by:

$$N_{update} = 0.1 \times D \times \left(1 - \left(m / Max\_iter\right)\right) \quad (2.16)$$

where m denotes the current number of iteration.

The N_{update} updated dimensions are randomly chosen each time a new candidate food source is generated.

5.3. Experimental study on numerical benchmark functions

In this section, the performances of the proposed algorithms are evaluated and compared to those of the ABC algorithm [119] and its well-known variants (GABC [153], '*the best-so-far selection in ABC*' dubbed here *'best-so-far ABC'* [179], MABC [180]). To highlight the main similarities and differences between these algorithms and the proposed ones, the different operations involved in each algorithm are gathered in table 2.1.

Table 2.1 : Characteristics of the different considered variants of the ABC algorithm.

Approach	Initial population	Update equation for employed bees	Update equation for onlookers bees	The *Limit* parameter
Basic ABC	Random	$v_{hj} = x_{hj} + \phi_{hj}\left(x_{hj} - x_{kj}\right)$	The same as that of the employed bees	$limit = SN \times D$
GABC	Random	$v_{hj} = x_{hj} + \phi_{hj}\left(x_{hj} - x_{kj}\right) + \psi_{hj}\left(y_j - x_{hj}\right)$	The same as that of the employed bees	$limit = SN \times D$
Best-so-far ABC	Random	$v_{hj} = x_{hj} + \phi_{hj}\left(x_{hj} - x_{kj}\right)$	$v_{hj} = x_{hj} + \phi_{hj} f_b\left(x_{hj} - x_{kj}\right)$	$limit = SN \times D$
MABC	Random	$v_{hj} = x_{hj} + \phi_{hj}\left(x_{hj} - x_{kj}\right)$, $\phi_{hj} = rand\left]-SF, SF\right[$	The same as that of the employed bees	$limit = SN \times D$
ABCEV	Chaotic	$v_{hj} = x_{best\,j} + \phi_{hj}\left(x_{r_1 j} - x_{r_2 j}\right)$	The same as that of the employed bees	$limit = 0.6 \times SN \times D$
EEABC	Chaotic	$v_{hj} = x_{best\,j} + \phi_{hj}\left(x_{r_1 j} - x_{r_2 j}\right)$	The same as that of the employed bees	$limit = 0.6 \times SN \times D$

In these algorithms, except for the best-so-far, the MABC and the EEABC algorithms, the update equation is applied to only one dimension of the solution. In the case of the MABC algorithm, the number of dimensions to be updated is a

control parameter of the algorithm whereas all dimensions of the onlooker bees' position are updated in the case of the best-so-far algorithm. In the EEABC algorithm, the number of dimensions to update is given by (2.16).

The probability equation for all algorithms is given by (2.9), except for the EEABC algorithm which uses that given by (2.14).

Some newer enhanced version of the ABC algorithm such as the works given in [152, 181] were not considered in this comparison. In these works more than one equation are used to update the onlooker's positions at the same time. In this work, only algorithms with one update equation are considered.

The comparative study is exclusively limited to ABC variants due to the fact that previous papers have already compared the performance of the ABC algorithm against existing conventional approaches such as GA, PSO and many other optimization algorithms [49, 120, 182].

5.3.1. Benchmark functions

Tables 2.2 and 2.3 give twelve standard benchmark functions frequently used to evaluate optimization algorithms [42]. While table 2.2 contains six unimodal benchmark functions, table 2.3 contains 6 multimodal benchmark functions where D denote the dimension of the solution. The global minimum of all these functions, except for f_7, is equal to 0. The other test bed we have also chosen is the fifteen benchmark functions proposed in the CEC2015 special session on 'Bound Constrained Single-Objective Computationally Expensive Numerical Optimization' [183]. These functions, which include hybrid and composition functions, are computationally expensive optimization problems that offer great complexity compared to the standard benchmark functions. Table 2.4 [183] lists a summary of these benchmark functions. A more detailed description of the CEC 2015 benchmark function could be found in annex C and in the CEC 2015 technical report [183]. The CEC 2015 benchmark functions were slightly modified in order to shift their minimums to 0.

Table 2.2 : Unimodal benchmark functions.

Function	Search range	Min
$f_1(\vec{x}) = \sum_{i=1}^{D} x_i^2$	$[-100,100]^D$	0
$f_2(\vec{x}) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10,10]^D$	0
$f_3(\vec{x}) = \max_i \{ \lvert x_i \rvert, 1 \le i \le D \}$	$[-100,100]^D$	0
$f_4(\vec{x}) = \sum_{i=1}^{D-1} 100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2$	$[-30,30]^D$	0
$f_5(\vec{x}) = \sum_{i=1}^{D} \left(x_i + 0.5\right)^2$	$[-100,100]^D$	0
$f_6(\vec{x}) = \sum_{i=1}^{D} i x_i^4 + random(0,1)$	$[-1.28,1.28]^D$	0

Table 2.3 : Multimodal benchmark functions.

Function	Search range	Min
$f_7(\vec{x}) = \sum_{i=1}^{D} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	$[-100,100]^D$	$-418.9829 * D$
$f_8(\vec{x}) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	$[-10,10]^D$	0
$f_9(\vec{x}) = 20 + \mathrm{e} - 20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)$	$[-100,100]^D$	0
$f_{10}(\vec{x}) = \frac{1}{4000}\left(\sum_{i=1}^{D}(x_i - 100)^2\right) - \left(\prod_{i=1}^{D}\cos\left(\frac{x_i - 100}{\sqrt{i}}\right)\right) + 1$	$[-30,30]^D$	0
$f_{11}(\vec{x}) = \frac{\pi}{D}\left(10\sin(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2\left(1 + 10\sin^2(\pi y_{i+1})\right) + (y_D - 1)^2\right) + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$	$[-100,100]^D$	0
$y_i = 1 + \frac{x_i + 1}{4}$		
$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$		
$f_{12}(\vec{x}) = 0.1\left(\sin^2(3\pi x_1) + \sum_{i=1}^{D}(x_i - 1)^2\left(1 + \sin^2(3\pi x_i + 1)\right) + (x_D - 1)^2\left(1 + \sin^2(2\pi x_D)\right)\right) + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	$[-1.28,1.28]^D$	0

Table 2.4 : Summary of the CEC2015 expensive optimization test problems.

Categories	Function	Description	Related basic functions
Unimodal functions	F_1	Rotated Bent Cigar function	Bent Cigar function
	F_2	Rotated Discus function	Discus function
Simple multimodal functions	F_3	Shifted and rotated Weierstrass function	Weierstrass function
	F_4	Shifted and rotated Schwefel's function	Schwefel's function
	F_5	Shifted and rotated Katsuura function	Katsuura function
	F_6	Shifted and rotated HappyCat function	HappyCat function
	F_7	Shifted and rotated HGBat function	HGBat function
	F_8	Shifted and rotated Expanded Griewank's plus Rosenbrock's function	Griewank's function Rosenbrock's function
	F_9	Shifted and rotated Expanded Scaffer's F6 function	Expanded Scaffer's F6 function
Hybrid functions	F_{10}	Hybrid function 1	Schwefel's function Rastrigin's function High Conditioned Elliptic function
	F_{11}	Hybrid function 2	Griewank's function Weierstrass function Rosenbrock's function Scaffer's F6 function
	F_{12}	Hybrid function 3	Katsuura function HappyCat function Griewank's function Rosenbrock's function Schwefel's function Ackley's function
Composition functions	F_{13}	Composition function 1	Rosenbrock's function High Conditioned Elliptic function Bent Cigar function Discus function High Conditioned Elliptic function
	F_{14}	Composition function 2	Schwefel's function Rastrigin's function High Conditioned Elliptic function
	F_{15}	Composition function 3	HGBat function Rastrigin's function Schwefel's function Weierstrass function High Conditioned Elliptic function

5.3.2. Experimental setup

The aim of this analysis is to compare the minimization quality of the proposed algorithms against other variants of the ABC algorithm by considering the numerical optimization (minimization) of the benchmark functions of table 2.2, 2.3 and 2.4.

For the first twelve benchmark functions, a population of 20 employed bees and 20 onlooker bees and a maximum number of 1500 iterations is chosen. The results of each algorithm are averaged over 150 runs. For the CEC 2015

benchmark functions, a population of 50 employed bees and 50 onlooker bees and a maximum number of 3000 iterations is used while the number of parameters to be updated in each iteration will be calculated using:

$$N_{update} = 0.2 \times D \times \left(1 - \left(m / Max\_iter\right)\right) \quad (2.17)$$

The results of each algorithm are averaged over 25 runs.

All the simulations are executed on the same Intel Core i5 3.10 GHz (TM) based machine. If the numerical function evaluation drops below 2.22e-16, it is reported as 0.

We are interested in recording the mean (found in column named *Mean*) of objectives values, its standard deviations (found in column named *SD*) and the medians (found in column named *median*) of the best solution in the last iteration. We are also interested in recording the number of iterations required to converge toward the solutions (only for the twelve first benchmark functions that have converged to zero within 7500 iterations) and the convergence speed of each algorithm.

5.3.3. Comparative results

The results of the comparison between the proposed algorithms and the other variants of the ABC algorithm are presented in tables 2.5 and 2.6 for the standard benchmark functions and in table 2.7 for the CEC 2015 benchmark functions. The bold font is used to indicate the minimum mean values in each row, whereas the grey background is used to indicate better optimization result than both of the proposed algorithms.

It can be observed from table 2.5 that both of the proposed algorithms give the best general results compared to the rest of the algorithms. From the 36 considered cases, both of these algorithms give the best solutions in 26 cases compared to all the other remaining algorithms. The GABC gives best solutions in 7 cases, the MABC is the best in 5 cases and the best-so-far is the best in only 2 cases. The basic ABC algorithm is outperformed by all of the other algorithms.

Nonetheless, the proposed EEABC algorithm seems to be having some problems with the functions f_9, f_{10}, f_{11} and f_{12}. A big difference is observed when

comparing the median values with the average values. Let us take the function f_{10} with 30 dimensions as an example. In this case, the median value is equal to 0 which means that in at least half of the 150 performed runs, the algorithm converged to zero. However, the average value of all 150 runs equals 1.381E-0.3 which indicates that in some of the runs, the optimization results were big enough to shift the average from 0 to 1.381E-0.3. The more reasonable conclusion is that the proposed algorithm is sometimes getting trapped within some local optimum instead of converging toward the global optimum. This problem is less apparent in the proposed ABCEV algorithm.

In table 2.6, the average convergence number of iterations of each algorithm is recorded. These results were obtained by increasing the maximum number of iterations to 7500. If the algorithm could converge within this number, the average number of iterations required to converge toward the optimum will be recorded. If for a given function, no algorithm could converge toward the optimum, the convergence information of this function is omitted.

Of the 24 cases, the EEABC requires the least number of iterations to converge in 17 cases, 4 cases for the ABCEV, 2 cases for the best-so-far and only one case for the MABC.

Comparing the convergence quality of the ABCEV and the EEABC, the EEABC is slightly better than the ABCEV. However, when comparing their convergence speed, the EEABC clearly outperforms the ABCEV.

The CEC 2015 benchmark results are gathered in table 2.7. From the 30 cases presented in this table, the EEABC algorithm outperforms its counterparts in half of them. The MABC has better results in 8 cases, the basic ABC in 5 cases, the Best-so-far in 2 cases while the ABCEV in only 1 case.

Comparing the obtained results with those of the standard benchmark functions, it appears that the ABCEV along with the GABC are not suitable for optimization problems of great complexity. This does not hold true, however, for the MABC which appears to achieve good performances and the basic algorithm whose performances have increased.

Table 2.5 : Comparative results of the convergence quality for the standard benchmark functions.

F	D	Basic ABC (best in 0 case)			GABC (best in 7 cases)			Best-so-far ABC (best in 2 cases)			MABC (best in 5 cases)			ABCEV (best in 14 cases)			EEABC (Best in 14 cases)		
		Average	STD	Median	Average	STD	Median	Average	STD	Median	Average	STD	Median	Average	STD	Median	Average	STD	Median
f_1	30	3.167E-14	9.537E-14	1.065E-14	**0.000E+00**	0.000E+00	0.000E+00	4.529E-08	3.619E-07	2.714E-13	**0.000E+00**	0.000E+00	0.000E+00	**0.000E+00**	0.000E+00	0.000E+00	**0.00E+00**	0.00E+00	0.00E+00
	60	1.174E-07	1.855E-07	6.139E-08	5.707E-10	9.422E-10	3.361E-10	9.971E-03	9.071E-03	7.494E-03	1.035E-08	5.890E-09	9.096E-09	1.680E-14	2.129E-14	9.945E-15	**0.00E+00**	0.00E+00	0.00E+00
	100	1.103E-04	1.900E-04	7.890E-05	3.520E-04	3.960E-04	2.104E-04	7.360E+02	2.641E+02	7.367E+02	5.181E-03	1.580E-03	4.971E-03	1.808E-07	1.298E-07	1.536E-07	**4.09E-09**	2.29E-09	3.54E-09
f_2	30	5.857E-08	2.765E-08	5.210E-08	1.702E-12	1.131E-12	1.419E-12	6.297E-07	7.710E-06	1.543E-10	1.040E-13	6.539E-14	8.475E-14	1.548E-16	3.181E-16	0.000E+00	**0.00E+00**	0.00E+00	0.00E+00
	60	2.524E-04	6.567E-05	2.411E-04	9.991E-06	7.151E-06	8.068E-06	3.024E-03	8.331E-04	2.892E-03	4.971E-06	1.537E-06	4.736E-06	2.140E-08	8.380E-09	2.087E-08	**9.56E-11**	3.76E-11	9.01E-11
	100	1.275E-02	2.138E-03	1.250E-02	1.113E-02	1.207E-02	4.936E-03	4.418E+00	7.502E-01	4.451E+00	4.362E-02	1.361E-02	4.068E-02	**7.185E-05**	5.167E-05	5.718E-05	7.33E-01	2.62E+00	4.56E-06
f_3	30	1.298E+01	3.701E+00	1.296E+01	8.622E+00	1.391E+00	8.622E+00	2.441E+01	2.839E+00	2.473E+01	5.900E+00	3.037E+00	6.091E+00	4.465E+00	9.912E-01	4.508E+00	**9.669E-01**	2.562E-01	9.251E-01
	60	5.527E+01	3.833E+00	5.517E+01	5.156E+01	3.136E+00	5.173E+01	5.977E+01	2.067E+00	5.978E+01	2.008E+01	3.035E+00	2.011E+01	5.476E+01	3.693E+00	5.485E+01	**1.967E+01**	2.735E+00	1.968E+01
	100	7.757E+01	2.928E+00	7.791E+01	7.633E+01	2.485E+00	7.679E+01	7.492E+01	1.420E+00	7.492E+01	**4.668E+01**	3.993E+00	4.688E+01	8.440E+01	3.314E+00	8.506E+01	5.352E+01	4.083E+00	5.366E+01
f_4	2	7.842E-03	9.184E-03	5.147E-03	3.606E-03	3.735E-03	2.281E-03	**5.881E-11**	3.350E-10	4.609E-13	4.894E-03	4.951E-03	3.570E-03	1.385E-04	1.933E-04	7.277E-05	2.061E-05	3.299E-05	8.817E-06
	3	1.106E-01	1.114E-01	8.147E-02	3.012E-02	2.511E-02	2.258E-02	4.782E-04	3.713E-03	2.267E-06	4.574E-02	4.539E-02	3.209E-02	2.365E-03	2.859E-03	1.189E-03	**3.473E-04**	5.845E-04	1.653E-04
	10	2.275E-01	3.335E-01	1.023E-01	**2.098E-01**	2.907E-01	9.135E-02	4.207E-01	9.850E-01	1.085E-01	1.565E+00	1.914E+00	6.589E-01	3.682E-01	1.172E+00	8.833E-03	2.891E-01	1.046E+00	7.752E-03
f_5	30	4.100E-14	1.421E-13	1.216E-14	**0.000E+00**	0.000E+00	0.000E+00	1.471E-08	1.727E-07	3.482E-13	**0.000E+00**	0.000E+00	0.000E+00	**0.000E+00**	0.000E+00	0.000E+00	**0.000E+00**	0.000E+00	0.000E+00
	60	8.621E-08	8.932E-08	6.014E-08	4.666E-10	5.142E-10	2.936E-10	9.382E-03	7.023E-03	7.252E-03	9.998E-09	4.576E-09	9.269E-09	1.706E-14	1.997E-14	1.178E-14	**0.000E+00**	0.000E+00	0.000E+00
	100	9.177E-05	5.774E-05	7.622E-05	3.821E-04	7.200E-04	2.035E-04	7.312E+02	2.696E+02	7.051E+02	5.219E-03	1.447E-03	4.905E-03	1.548E-07	9.510E-08	1.341E-07	**3.995E-09**	2.147E-09	3.795E-09
f_6	30	7.825E-02	2.020E-02	7.880E-02	5.399E-02	1.340E-02	5.338E-02	**1.794E-02**	6.893E-03	1.685E-02	1.089E-01	2.846E-02	1.060E-01	4.098E-02	1.019E-02	4.193E-02	2.801E-02	7.776E-03	2.706E-02
	60	2.843E-01	9.218E-02	2.847E-01	2.777E-01	4.100E-02	2.784E-01	1.626E-01	4.630E-02	1.573E-01	2.418E-01	4.679E-02	2.442E-01	1.703E-01	3.236E-02	1.721E-01	**8.521E-02**	1.777E-02	8.301E-02
	100	6.893E-01	2.441E-01	6.861E-01	9.391E-01	1.178E-01	9.439E-01	8.697E-01	1.685E-01	8.477E-01	4.773E-01	8.177E-02	4.717E-01	6.069E-01	7.966E-02	6.029E-01	**2.525E-01**	3.116E-01	2.237E-01
f_7	30	-1.24E+04	8.578E+01	-1.25E+04	**-1.25E+04**	6.891E+01	-1.26E+04	-1.19E+04	1.812E+02	-1.19E+04	-8.78E+03	3.424E+02	-8.77E+03	-1.25E+04	7.948E+01	-1.26E+04	-1.24E+04	1.68E+02	-1.25E+04
	60	-2.35E+04	2.315E+02	-2.35E+04	-2.37E+04	2.433E+02	-2.37E+04	-2.17E+04	3.087E+02	-2.17E+04	-1.56E+04	6.132E+02	-1.55E+04	**-2.49E+04**	1.747E+02	-2.49E+04	-2.3E+04	5.24E+02	-2.30E+04
	100	-3.66E+04	3.811E+02	-3.66E+04	-3.63E+04	4.052E+02	-3.63E+04	-3.21E+04	4.799E+02	-3.21E+04	-1.73E+04	6.475E+02	-1.73E+04	**-4.11E+04**	2.743E+02	-4.10E+04	-3.69E+04	8.99E+02	-3.69E+04
f_8	30	3.064E-09	2.916E-08	8.583E-12	**4.257E-12**	1.959E-11	2.274E-13	3.073E+00	1.023E+00	3.123E+00	4.145E+01	7.918E+00	4.129E+01	1.305E-08	8.599E-08	5.684E-14	9.984E-02	3.599E-01	1.376E-09
	60	6.335E+00	2.413E+00	6.332E+00	4.036E+00	1.168E+00	4.158E+00	3.532E+01	3.363E+00	3.523E+01	1.346E+02	1.735E+01	1.343E+02	**2.560E-01**	4.924E-01	1.327E-08	7.353E+00	3.611E+00	6.965E+00
	100	4.346E+01	6.645E+00	4.434E+01	4.150E+01	3.814E+00	4.087E+01	1.368E+02	8.790E+00	1.378E+02	6.386E+02	3.641E+01	6.429E+02	**6.659E+00**	2.326E+00	6.633E+00	4.179E+01	1.237E+01	3.956E+01
f_9	30	6.938E-08	9.298E-08	4.655E-08	1.444E-11	1.241E-11	1.052E-11	5.272E-05	3.475E-04	2.543E-06	3.885E-02	1.857E-01	2.180E-03	**5.503E-14**	1.074E-14	5.418E-14	5.086E-01	2.712E+00	3.286E-14
	60	1.080E-04	5.513E-05	9.591E-05	1.028E-04	5.794E-05	8.725E-05	1.682E+00	5.634E-01	1.788E+00	6.130E-01	4.739E-01	5.559E-01	**1.581E-07**	8.615E-08	1.405E-07	4.720E+00	7.160E+00	4.870E-09
	100	1.460E+00	2.859E-01	1.508E+00	3.600E-01	1.142E-01	3.583E-01	7.744E+00	6.564E-01	7.695E+00	2.884E+00	9.173E-01	2.739E+00	**5.734E-03**	3.361E-03	4.631E-03	8.880E+00	8.656E+00	9.554E+00
f_{10}	30	9.704E-04	3.574E-03	6.967E-13	4.471E-04	2.057E-03	4.772E-13	6.285E-04	2.519E-03	9.207E-13	**8.688E-06**	4.859E-05	8.915E-08	3.531E-03	9.950E-03	1.937E-14	1.381E-03	3.373E-03	0.000E+00
	60	4.424E-03	8.443E-03	2.905E-07	5.517E-04	2.726E-03	1.374E-08	3.574E-03	6.214E-03	9.381E-04	**9.569E-05**	3.027E-04	2.791E-05	2.112E-03	7.000E-03	2.943E-13	1.035E-03	3.233E-03	0.000E+00
	100	1.537E-02	2.808E-02	2.154E-04	2.770E-02	3.019E-02	1.720E-02	1.215E+00	7.561E-02	1.202E+00	1.459E-02	9.615E-03	1.222E-02	1.168E-03	3.896E-03	2.007E-07	**1.165E-03**	4.526E-03	1.939E-09
f_{11}	30	1.487E-12	1.074E-11	2.413E-15	**0.000E+00**	0.000E+00	0.000E+00	2.201E-10	2.063E-09	8.477E-15	1.294E-01	3.064E-01	3.410E-06	2.561E-16	3.136E-15	0.000E+00	6.911E-04	8.465E-03	0.000E+00
	60	1.820E-08	6.892E-08	2.323E-09	1.261E-10	9.159E-10	6.181E-12	2.197E-04	3.310E-04	1.400E-04	1.607E+00	1.010E+00	1.598E+00	**3.396E-15**	2.166E-14	0.000E+00	1.590E-02	5.303E-02	0.000E+00
	100	1.004E-06	1.213E-06	7.084E-07	2.146E-06	1.031E-05	2.144E-07	2.164E+00	5.237E-01	2.136E+00	5.308E+00	1.820E+00	5.132E+00	**6.098E-09**	6.687E-08	2.401E-10	1.075E-01	2.027E-01	2.875E-06
f_{12}	30	2.255E-11	2.572E-10	2.211E-14	**0.000E+00**	0.000E+00	0.000E+00	6.088E-10	5.875E-09	3.140E-14	1.217E-03	3.371E-03	5.544E-07	1.048E-09	1.284E-08	0.000E+00	3.649E-14	3.444E-13	0.000E+00
	60	1.671E-07	5.576E-07	5.133E-08	3.282E-09	1.423E-08	2.112E-10	1.651E-03	2.485E-03	8.860E-04	9.465E-03	5.452E-03	7.018E-03	**3.372E-14**	1.464E-13	5.887E-15	1.094E-02	1.304E-01	0.000E+00
	100	6.083E-05	9.151E-05	3.875E-05	1.904E-04	1.118E-03	9.569E-06	9.447E+03	2.108E+04	1.112E+03	1.118E+00	1.878E-01	1.097E+00	**4.007E-05**	4.887E-04	1.075E-08	9.123E-02	4.941E-01	2.653E-08

Table 2.6 : Comparative results of the convergence iteration.

F	D	Basic ABC (best in 0 cases)	GABC (best in 0 cases)	Best-so-far ABC (best in 2 cases)	MABC (best in 1 cases)	ABCEV (best in 4 cases)	EEABC (best in 17 cases)
f_1	30	1699.167	1127.62	2050.907	1230.013	813.8333	**610.98**
	60	3478.78	2317.333	5159.513	2336.173	1672.013	**1166.2**
	100	5862.527	3915.393	7500	3943.72	2849.26	**1899.653**
f_2	30	3419.22	1960.067	2231.433	1749.173	1477.567	**954.42**
	60	7043.107	4048.687	5035.72	3376.42	3069.093	**1745.84**
	100	7500	6905.613	7500	5785.087	5233.307	**3025.04**
f_3	30	7500	7500	7500	**7498.707**	7500	7500
f_4	2	7500	7500	**4175.1**	7500	7500	7500
	3	7500	7500	**7461.487**	7500	7500	7500
f_5	30	1706.44	1127	1982.173	1234.767	820.4933	**611.9067**
	60	3459.767	2310.753	5193.213	2339.02	1674.273	**1166.04**
	100	5876.127	3912.953	7498.78	3942.913	2846.227	**1897.193**
f_8	30	3348.093	2457.813	3271.407	7500	**2068.947**	2164.927
	60	6900.487	6620.707	7477.147	7500	**6565.127**	7407.893
	100	7489.14	7475.4	7500	7500	**7332.727**	7500
f_{10}	30	4539.533	3765.58	2227.007	6869.927	2538.54	**1183.653**
	60	5078.867	4675.28	4337.633	7368.767	2965.44	**1865.387**
	100	6425.513	5526.747	6321.64	7450	3682.173	**2609.493**
f_{11}	30	1649.753	1063.367	1911.947	7500	753.9	**575.88**
	60	3240.733	2130.573	4517.793	7500	1501.713	**1361.58**
	100	5415.113	3543.073	7329.28	7500	**2510.387**	2800.313
f_{12}	30	1813.22	1147.02	1894.7	7500	812.1	**600.7267**
	60	3713.173	2313.053	4700.773	7500	1656.473	**1209.54**
	100	6092.047	3928.413	7482.413	7500	2800.12	**2396.7**

Figures 2.3 - 2.6 show the convergence speed results of the different algorithms (only 4 cases are considered). In figures 2.3, 2.4 and 2.6, the MABC starts as the fastest algorithm; however the proposed EEABC rapidly reaches and surpasses it. Figure 2.5 shows that the proposed ABCEV is the fastest algorithm, although at the start of the optimization process, the EEABC was faster.

Table 2.7 : Comparative results of the convergence quality for the CEC 2015 benchmark functions.

F	D	Basic ABC (best in 5 case)			GABC (best in 0 cases)			Best-so-far ABC (best in 2 cases)			MABC (best in 8 cases)			ABCEV (best in 1 cases)			EEABC (Best in 15 cases)		
		Average	STD	Median	Average	STD	Median	Average	STD	Median	Average	STD	Median	Average	STD	Median	Average	STD	Median
F_1	10	1.710E+03	1.136E+03	1.496E+03	7.512E+03	4.802E+03	6.434E+03	**6.81E+02**	5.530E+02	5.612E+02	7.687E+02	1.326E+03	1.163E+02	1.219E+04	9.303E+03	1.109E+04	1.032E+04	1.009E+04	6.034E+03
	30	**8.30E+02**	5.914E+02	5.838E+02	9.316E+03	4.642E+03	8.700E+03	1.001E+04	8.496E+03	7.585E+03	1.436E+03	1.137E+03	1.150E+03	3.874E+03	3.417E+03	2.826E+03	5.259E+03	3.934E+03	5.596E+03
F_2	10	9.628E+03	2.502E+03	9.874E+03	8.284E+03	2.138E+03	8.334E+03	1.387E+04	4.365E+03	1.363E+04	1.357E+04	4.088E+03	1.319E+04	5.842E+03	1.759E+03	5.806E+03	**5.341E+03**	1.898E+03	4.941E+03
	30	8.783E+04	1.208E+04	8.971E+04	8.511E+04	1.350E+04	8.656E+04	9.123E+04	1.220E+04	9.358E+04	7.263E+04	1.358E+04	7.493E+04	7.659E+04	1.122E+04	7.674E+04	**3.845E+04**	6.794E+03	3.850E+04
F_3	10	4.432E+00	8.264E-01	4.551E+00	3.393E+00	9.429E-01	3.508E+00	3.673E+00	7.387E-01	3.886E+00	2.837E+00	9.606E-01	2.924E+00	2.707E+00	1.096E+00	2.414E+00	**2.099E+00**	1.434E+00	1.991E+00
	30	3.048E+01	1.471E+00	3.035E+01	2.759E+01	2.056E+00	2.777E+01	3.180E+01	2.104E+00	3.161E+01	1.740E+01	1.790E+00	1.744E+01	2.738E+01	2.104E+00	2.766E+01	**1.729E+01**	2.418E+00	1.772E+01
F_4	10	2.000E-02	3.203E-02	3.001E-11	4.965E-05	2.464E-04	0.000E+00	9.346E-02	4.378E-02	8.474E-02	7.444E+01	5.702E+01	5.835E+01	1.819E-14	1.286E-13	0.000E+00	**0.000E+00**	0.000E+00	0.000E+00
	30	**3.400E-01**	4.878E-01	1.770E-01	5.868E-01	1.163E+00	2.302E-01	5.989E+00	2.051E+00	5.763E+00	1.456E+03	2.826E+02	1.478E+03	5.270E-01	9.966E-01	1.458E-01	2.633E+01	4.473E+01	9.088E+00
F_5	10	2.471E-01	5.898E-02	2.469E-01	4.456E-01	1.074E-01	4.494E-01	3.754E-01	7.473E-02	3.827E-01	**9.341E-02**	3.695E-02	8.883E-02	4.052E-01	1.326E-01	4.011E-01	4.207E-01	1.309E-01	3.971E-01
	30	6.960E-01	1.401E-01	6.871E-01	8.462E-01	1.623E-01	8.549E-01	8.537E-01	1.666E-01	8.610E-01	**1.633E-01**	4.829E-02	1.587E-01	9.163E-01	2.040E-01	9.246E-01	1.032E+00	2.348E-01	1.009E+00
F_6	10	1.454E-01	2.292E-02	1.444E-01	1.437E-01	3.052E-02	1.477E-01	**8.794E-02**	1.963E-02	8.387E-02	1.159E-01	3.456E-02	1.127E-01	1.192E-01	2.258E-02	1.211E-01	1.126E-01	1.882E-02	1.134E-01
	30	**1.822E-01**	2.873E-02	1.810E-01	2.401E-01	3.441E-02	2.421E-01	2.076E-01	3.019E-02	2.110E-01	2.733E-01	3.544E-02	2.789E-01	1.829E-01	3.137E-02	1.819E-01	2.292E-01	4.391E-02	2.285E-01
F_7	10	1.336E-01	3.019E-02	1.359E-01	7.976E-02	1.705E-02	7.955E-02	6.663E-02	1.992E-02	6.718E-02	1.644E-01	5.796E-02	1.569E-01	5.769E-02	1.739E-02	5.483E-02	**5.352E-02**	1.762E-02	5.056E-02
	30	2.258E-01	3.504E-02	2.261E-01	2.117E-01	2.967E-02	2.102E-01	2.265E-01	2.417E-02	2.261E-01	2.693E-01	2.779E-02	2.714E-01	**1.853E-01**	1.994E-02	1.789E-01	1.934E-01	3.040E-02	1.875E-01
F_8	10	8.301E-01	2.769E-01	8.179E-01	7.307E-01	2.046E-01	7.229E-01	8.638E-01	1.940E-01	8.712E-01	7.206E-01	2.276E-01	7.144E-01	6.413E-01	1.642E-01	6.207E-01	**5.757E-01**	1.092E-01	5.875E-01
	30	1.918E+01	3.772E+00	1.958E+01	1.400E+01	2.353E+00	1.408E+01	2.727E+01	3.994E+00	2.750E+01	6.567E+00	1.732E+00	6.531E+00	1.107E+01	2.309E+00	1.179E+01	**5.650E+00**	1.433E+00	5.800E+00
F_9	10	2.784E+00	2.478E-01	2.813E+00	2.579E+00	2.970E-01	2.566E+00	2.652E+00	2.161E-01	2.673E+00	3.082E+00	2.347E-01	3.140E+00	2.577E+00	2.798E-01	2.574E+00	**2.397E+00**	2.219E-01	2.372E+00
	30	1.270E+01	2.823E-01	1.274E+01	1.251E+01	3.245E-01	1.254E+01	1.277E+01	2.692E-01	1.278E+01	**1.216E+01**	3.850E-01	1.228E+01	1.248E+01	2.997E-01	1.251E+01	1.229E+01	3.071E-01	1.225E+01
F_{10}	10	3.992E+03	2.748E+03	2.843E+03	3.869E+03	1.737E+03	3.585E+03	1.346E+04	1.314E+04	8.778E+03	**9.675E+02**	4.345E+02	9.361E+02	2.660E+03	1.321E+03	2.308E+03	3.263E+03	1.772E+03	3.003E+03
	30	3.856E+05	2.240E+05	3.373E+05	6.405E+05	2.809E+05	5.692E+05	9.260E+05	2.909E+05	8.975E+05	**4.231E+04**	3.108E+04	3.327E+04	6.406E+05	5.197E+05	4.219E+05	5.384E+05	3.081E+05	4.045E+05
F_{11}	10	3.230E+00	5.660E-01	3.306E+00	2.882E+00	4.821E-01	2.821E+00	2.376E+00	4.075E-01	2.390E+00	2.907E+00	5.953E-01	2.967E+00	2.620E+00	4.567E-01	2.572E+00	**2.246E+00**	3.766E-01	2.142E+00
	30	1.909E+01	1.308E+00	1.923E+01	1.891E+01	1.602E+00	1.928E+01	1.758E+01	1.453E+00	1.785E+01	**1.554E+01**	1.278E+00	1.579E+01	1.928E+01	1.480E+00	1.939E+01	1.735E+01	1.741E+00	1.719E+01
F_{12}	10	2.863E+01	3.691E+00	2.785E+01	2.804E+01	2.100E+00	2.793E+01	3.044E+01	4.920E+00	2.920E+01	3.057E+01	9.616E+00	2.786E+01	2.792E+01	2.612E+00	2.696E+01	**2.708E+01**	2.038E+00	2.668E+01
	30	3.129E+02	1.086E+02	2.844E+02	2.808E+02	9.185E+01	2.865E+02	3.515E+02	1.075E+02	3.615E+02	**1.489E+02**	6.317E+01	1.345E+02	2.967E+02	1.006E+02	3.001E+02	1.652E+02	6.945E+01	1.482E+02
F_{13}	10	**2.45E+02**	6.665E+01	2.263E+02	2.974E+02	3.910E+01	3.152E+02	2.601E+02	5.664E+01	2.385E+02	3.059E+02	3.090E+01	3.149E+02	3.156E+02	7.581E-01	3.151E+02	3.153E+02	3.633E-01	3.157E+02
	30	3.277E+02	4.276E-02	3.277E+02	3.278E+02	2.322E-01	3.278E+02	3.279E+02	1.017E-01	3.279E+02	**3.276E+02**	3.298E-13	3.276E+02	3.279E+02	8.221E-01	3.278E+02	**3.276E+02**	2.760E-13	3.276E+02
F_{14}	10	1.917E+02	3.438E+00	1.917E+02	1.914E+02	3.957E+00	1.909E+02	1.913E+02	3.531E+00	1.921E+02	1.898E+02	3.399E+00	1.896E+02	1.880E+02	3.494E+00	1.881E+02	**1.870E+02**	4.494E+00	1.865E+02
	30	2.223E+02	3.349E+00	2.223E+02	2.224E+02	3.431E+00	2.222E+02	2.250E+02	3.779E+00	2.250E+02	2.182E+02	2.683E+00	2.181E+02	2.228E+02	4.899E+00	2.228E+02	**2.156E+02**	3.266E+00	2.161E+02
F_{15}	10	1.235E+01	2.463E+00	1.182E+01	1.228E+01	1.995E+00	1.209E+01	1.845E+01	5.492E+01	1.010E+01	8.760E+01	1.541E+02	1.163E+01	8.977E+00	1.037E+00	9.099E+00	**7.524E+00**	1.994E+00	7.275E+00
	30	**4.31E+02**	7.436E+00	4.307E+02	4.522E+02	1.651E+01	4.475E+02	4.365E+02	8.704E+00	4.357E+02	4.386E+02	1.959E+01	4.346E+02	4.499E+02	2.317E+01	4.465E+02	6.774E+02	9.249E+01	6.917E+02

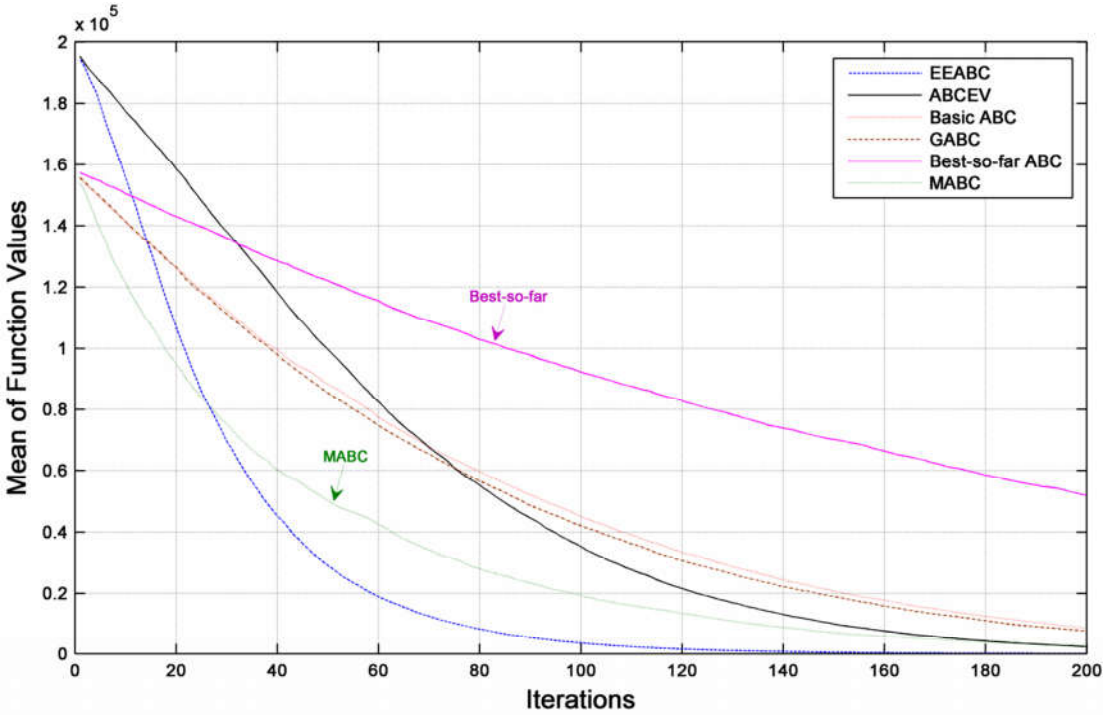

Figure 2.3 : Convergence speed for function f1 with D=60.

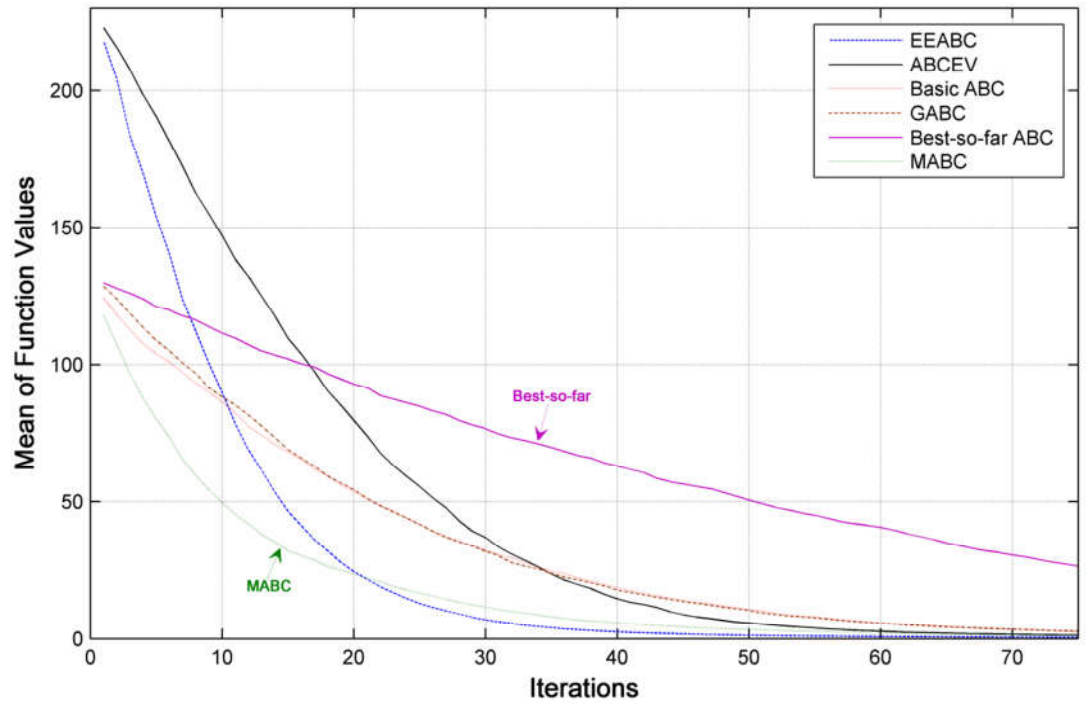

Figure 2.4 : Convergence speed for function f6 with D=30.

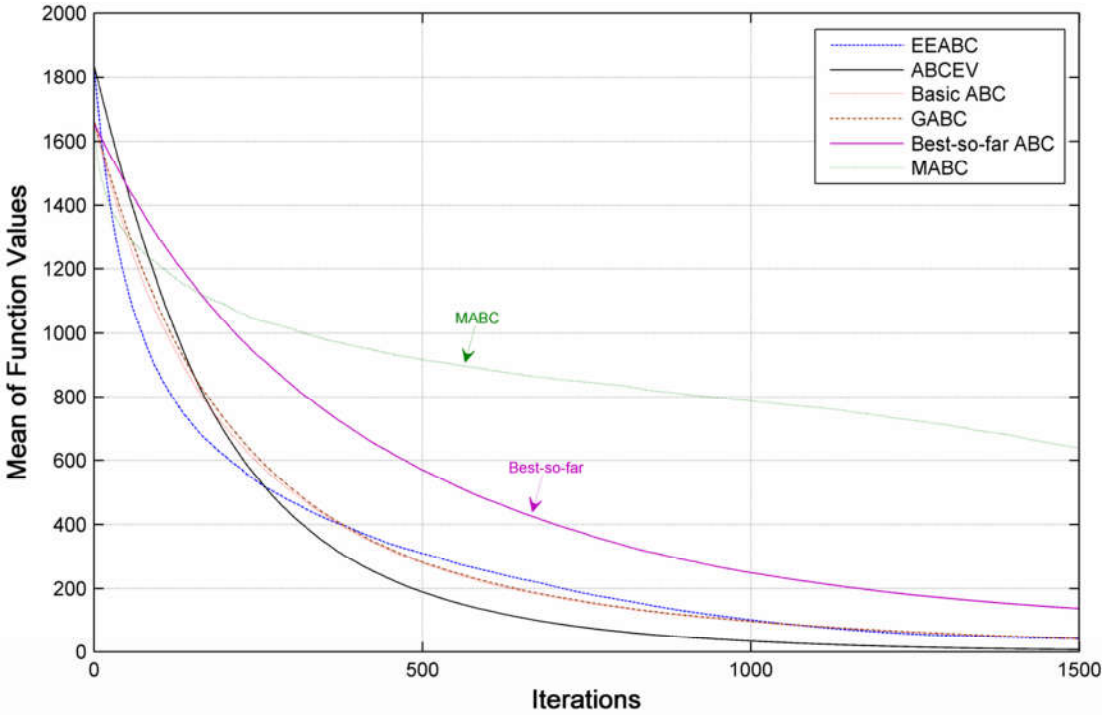

Figure 2.5 : Convergence speed for function f8 with D=100.

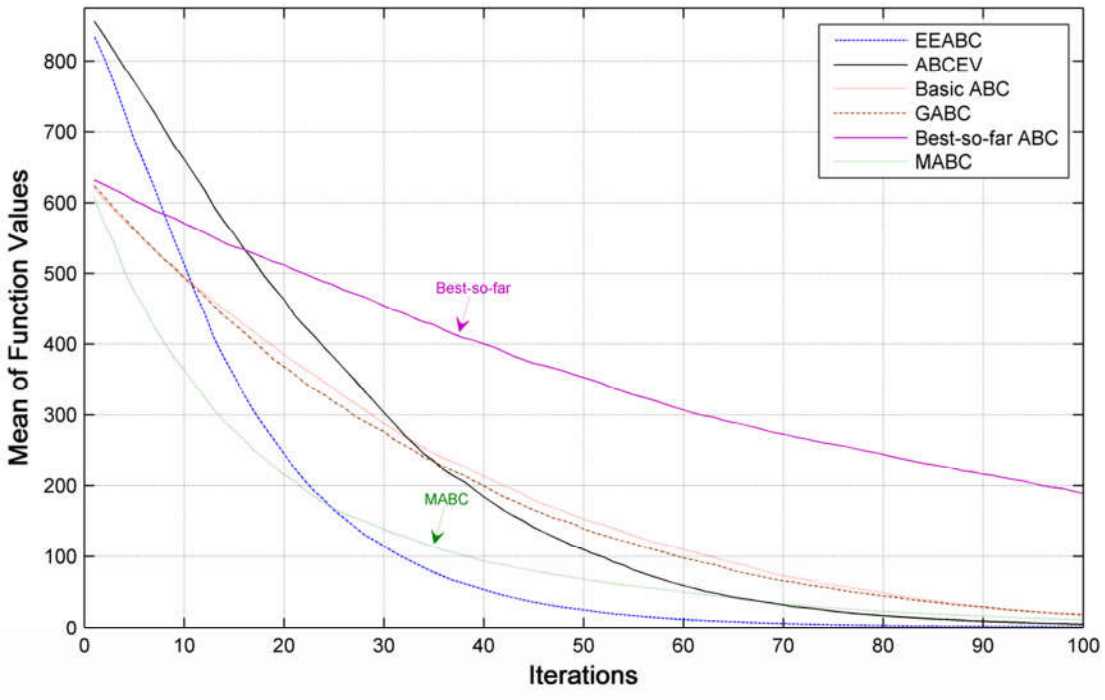

Figure 2.6 : Convergence speed for function f10 with D=30.

6. conclusion

In this chapter, two novel variants of the ABC algorithm were proposed. A comparative study against several existing variants of the ABC algorithm has indicated the good performances of the proposed algorithms.

Although the EEABC has generated relatively good performances, in some situations, some deficiencies has been observed. As such, further future improvements should be carried out in order to overcome these deficiencies.

CHAPTER3:

FUZZY MODEL PREDICTIVE CONTROL BASED ON META-HEURISTIC ALGORITHMS

1. Introduction

After evaluating the proposed optimization algorithms using numerical benchmark functions in chapter 2. The next phase would be to implement these optimization algorithms within nonlinear control strategies, and evaluate the performances of the entire controllers.

The current chapter starts by formulating the complete fuzzy predictive control strategy. TS fuzzy technique, used to construct nonlinear prediction models, is initially given. The developed control algorithm around meta-heuristic algorithms is then presented along with the adopted techniques used to handle the constraints.

The next section of the chapter contains a detailed description of the proposed NMPC control algorithms; two are based on the previously developed ABC algorithms while the last one is based on an efficient PSO algorithm. These three control algorithms are evaluated and compared to a number of linear and non-linear control strategies by considering the control of a simulated CSTR process, a model of an industrial boiler and a DSP based experimental setup.

2. Fuzzy based model predictive control

2.1. Takagi-Sugeno dynamic fuzzy modelling

The first step in the fuzzy based nonlinear model predictive control (FNMPC) is the construction of the fuzzy nonlinear model of the system.

Fuzzy Inference Systems (FIS) are capable of approximating any continuous function with a certain level of accuracy; they are universal approximators [22]. In addition, they have the ability to either extract or incorporate human knowledge directly via linguistic data [13, 184]. Takagi-Sugeno (TS) models, a subdivision of fuzzy models, are particularly suitable for NMPC

algorithms [23]. These models are able to express the dynamic nature of systems with characteristics of randomness, large delay time and strong nonlinearity [185, 186].

The Takagi-Sugeno FIS are based on the so called IF-THEN rules that have the following form:

$$R_j: \quad \text{IF} \quad \mathcal{F} \ \text{is} \ A_j \qquad \text{THEN} \qquad y_j = \theta_j^T \mathcal{X} \tag{3.1}$$

where $\mathcal{F}=\{y(t),\ y(t-1)\ldots\ y(t-n_a+1),\ u(t),\ u(t-1)\ldots\ u(t-m_a+1)\}$ is the set of n_a+m_a premise values, $\mathcal{X}^T=\{y(t),y(t-1)\ldots y(t-n_r+1),u(t),u(t-1)\ldots u(t-m_r+1),1\}$ is the set of m_r inputs and n_r outputs values used in the consequent regressors, $A_j=\{A_{1,j},A_{2,j}\ldots A_{n_a,j},B_{1,j},B_{2,j}\ldots B_{m_a,j}\}$ is the set of membership functions associated with the antecedents of the j^{th} rule, $\theta_j^T=\{a_{j,1},a_{j,2}\ldots a_{j,n_r},b_{j,1},b_{j,2}\ldots b_{j,m_r},c_j\}$ is the parameter vector of the j^{th} sub-model while y_j is its output ($1\le j\le r$; r is the number of fuzzy rules).

It is clear, from equation (3.1), that the Takagi-Sugeno fuzzy models only have fuzzy propositions in their antecedents while their consequences are linear functions of the antecedents or their variables.

Using Input/output data extracted from a nonlinear process, a fuzzy model can be constructed to mimic the process behaviour by defining the parameter matrix as:

$$\Theta=[\theta_1,\theta_2\ldots\ \theta_r], \tag{3.2}$$

and the normalized membership grade vector as:

$$\mu^T=[\mu_1,\mu_2\ldots\ \mu_r], \tag{3.3}$$

The inferred output of the TS fuzzy model is given by:

$$\hat{y}(t+1)=\left(\Theta\mu\right)^{\mathrm{T}}\chi \tag{3.4}$$

Using the input/output representation, the TS model given by (3.4) can be rewritten as follows:

$$\hat{y}(t+1)=\sum_{p=1}^{n_a}\bar{a}_p(t)\,y(t-p+1)+\sum_{p=1}^{m_a}\bar{b}_p(t)u(t-p+1)+\bar{c}(t) \tag{3.5}$$

where $\bar{a}_p(t)=\sum_{l=1}^{r}\mu_l(t)a_{l,p}$, $\bar{b}_p(t)=\sum_{l=1}^{r}\mu_l(t)b_{l,p}$ and $\bar{c}(t)=\sum_{l=1}^{r}\mu_l(t)c_l$.

The global output of the fuzzy model can be written as:

$$\begin{aligned}\hat{y}(t+1)&=\sum_{p=1}^{r}\mu_p(t)\,y_p(t+1)\\&=\left[\mu_1(t)y(t),\ldots,\mu_1(t)u(t-m_r+1),\ldots,\mu_r(t)\right]\tilde{\theta}\\&=\varphi^T(t)\tilde{\theta}\end{aligned} \tag{3.6}$$

where $y_p(t+1)$ is the output of the p^{th} sub-model and:

$$\tilde{\theta}^T=\left[\theta_1^T\ \ \theta_2^T\ldots\theta_r^T\right] \tag{3.7}$$

Assuming that a set of N input-output data pairs $\left(u(t),y(t)\right)$ is available and defining $L=\max\left\{m_r,n_r\right\}$, a regression matrix which has the following expression can be constructed:

$$\Phi=\begin{bmatrix}\varphi^T(L)\\\varphi^T(L+1)\\\cdots\\\varphi^T(N-1)\end{bmatrix} \tag{3.8}$$

The vector $\tilde{\theta}$ can be calculated by solving the following least square problem:

$$Y=\Phi\tilde{\theta} \tag{3.9}$$

where $Y=\left[y(L+1),\ldots,y(N)\right]^T$.

The resolution of equation (3.9) which is the last step in TS fuzzy model construction deals with the definition of the consequences part of the rules. However, the antecedent which involves in part the number, the position, the shape and the distribution of the membership functions must be first selected. A lot

of techniques for the construction of fuzzy models from Input-Output data are described in the literature [184, 187].

2.2. Notation

First, let us define the adopted notation in the formulation of the proposed control algorithm. We consider a MIMO system with m inputs and n outputs and for which a TS fuzzy model [185] is used as the explicit model of the system. The process output at the future instant $t+j$ is given by:

$$\hat{y}(t+j\,|\,t) = \hat{y}_{model}(t+j\,|\,t) + d(t+j) \tag{3.10}$$

where $N_1 \le j \le N_2$, $\hat{y}_{model}(t+j\,|\,t)$ is the TS fuzzy model estimated using measured and estimated past values of the system output and $d(t)$ is a disturbance (model/process mismatch).

Since $d(t+j)$ cannot be measured, an estimate value will be used. The predicted disturbance over the future sampling time will be considered equal to that of the current sampling time [13, 188]. Therefore:

$$d(t+j) = d(t) = y(t) - \hat{y}_{model}(t) \tag{3.11}$$

The optimal control sequence is denoted by $U(t) = \{u(t), u(t+1), \ldots, u(t+N_u-1)\}$, where $u(t) = [u_1(t), u_2(t), \cdots, u_m(t)]^T$ is the system input vector. The cost function J is used to calculate the fitness function $f_{itness}(.) = \dfrac{1}{J(.)}$.

The position of the h^{th} element (gene, particle, artificial bee…etc.) at the sampling time t, is represented by:

$$X_h(t) = \{u_h(t), u_h(t+1), \ldots, u_h(t+N_u-1)\} \tag{3.12}$$

where $u_h(t+j) = [u_1^h(t+j), \cdots, u_m^h(t+j)]^T$ j=0,…,N_u-1.

The control structure is shown in figure 3.1.

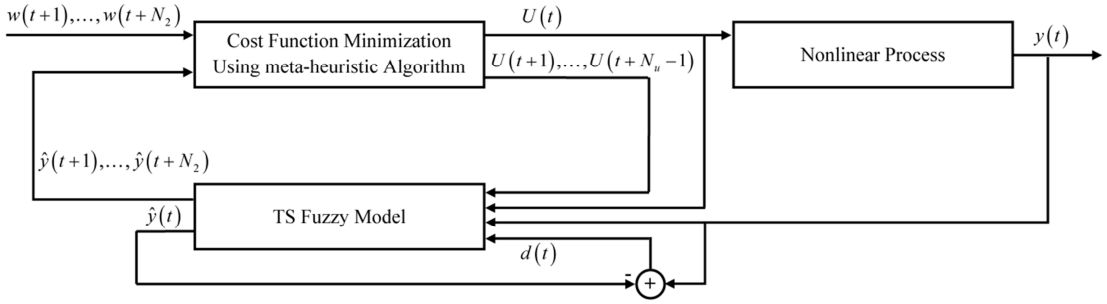

Figure 3.1 : Bloc diagram of the proposed NMPC algorithm.

2.3. Solving the fuzzy NMPC optimization problem

Let us consider the following typical NMPC optimal control problem:

$$\min_{\Delta\hat{u}(t)} J\left(\hat{u}(t),\hat{y}(t),w(t)\right)=\min_{\Delta\hat{u}(t)}\sum_{j=N_1}^{N_2}\left\|\hat{y}(t+j\,|\,t)-w(t+j)\right\|^2_{Q(j)}+\sum_{j=1}^{N_u}\left\|\Delta\hat{u}(t+j-1)\right\|^2_{R(j)} \tag{3.13}$$

Subject to:

$$\Delta\hat{u}(t+j-1)=0 \quad \text{for} \quad j>N_u \tag{3.14}$$

$$\Delta u_{min}<\Delta\hat{u}(t+j-1)<\Delta u_{max} \quad \text{for} \quad j\le N_u \tag{3.15}$$

$$u_{min}<\hat{u}(t)<u_{max} \tag{3.16}$$

$$y_{min}<\hat{y}(t)<y_{max} \tag{3.17}$$

$y_{min}=\left[y_{\min 1},\ldots,y_{\min n}\right]^T$ and $y_{max}=\left[y_{\max 1},\ldots,y_{\max n}\right]^T$ limit the range where the outputs are allowed to exist, $u_{min}=\left[u_{\min 1},\ldots,u_{\min m}\right]^T$ and $u_{max}=\left[u_{\max 1},\ldots,u_{\max m}\right]^T$ limit the range of the inputs while $\Delta u_{min}=\left[\Delta u_{\min 1},\ldots,\Delta u_{\min m}\right]^T$ and $\Delta u_{max}=\left[\Delta u_{\max 1},\ldots,\Delta u_{\max m}\right]^T$ limit the range of the inputs increments. The predictions $\hat{y}(t+j\,|\,t)$ are obtained using a TS fuzzy model of the system.

The first step toward solving the OCP (3.13)-(3.17) consists of choosing a suitable strategy to handle its constraints.

2.3.1. Constraints handling

From chapter 1, we could consider the constraints (3.14)-(3.16) on the inputs as hard constraints that cannot be violated whereas the constraint (3.17) on the outputs as soft constraint that can be violated. As such, a hybrid approach in handling the constraints (3.14)-(3.17) is adopted. The input constraints are

handled using a preserving strategy while the output constraints are handled using a penalizing strategy.

a) Outputs constraints

The easy way to soften the outputs constraints is to add new variables, called slack variables, in the cost function to heavily penalize any deviation or violation of the constraints. The output-dependent weight function $\Gamma_y(y)$ (Figure 3.2) was chosen to soften the output constraints. It has the following expression [9]:

$$\Gamma_y(\hat{y}) = \begin{bmatrix} \Gamma_{\hat{y}_1}(\hat{y}_1) & 0 & \dots & 0 \\ 0 & \Gamma_{\hat{y}_2}(\hat{y}_2) & \dots & 0 \\ 0 & \vdots & \ddots & 0 \\ 0 & \dots & 0 & \Gamma_{\hat{y}_n}(\hat{y}_n) \end{bmatrix}, \text{ with} \tag{3.18}$$

$$\Gamma_{y_i}(y_i) = \begin{cases} \Gamma_{y_i}(0)\left[1+\varepsilon_i\left(y_i - y_{\min i}\right)^2\right] & \text{if } y_i \le y_{\min i} \\ \Gamma_{y_i}(0) & \text{if } y_{\min i} \le y_i \le y_{\max i} \\ \Gamma_{y_i}(0)\left[1+\varepsilon_i\left(y_i - y_{\max i}\right)^2\right] & \text{if } y_i \ge y_{\max i} \end{cases}$$

where $i = 1,\dots,n$, $\hat{y}(t) = \left[\hat{y}_1(t), \hat{y}_2(t), \cdots, \hat{y}_n(t)\right]^T$ and ε_i is used to define the degree of softening: $\varepsilon_i = \infty$ indicates hard constraint while $\varepsilon_i = 0$ indicates no constraint.

The function Γ_{y_i} can also be used on the unconstrained outputs y_i by setting $y_{\min i}$ and $y_{\max i}$ respectively to $-\infty$ and $+\infty$.

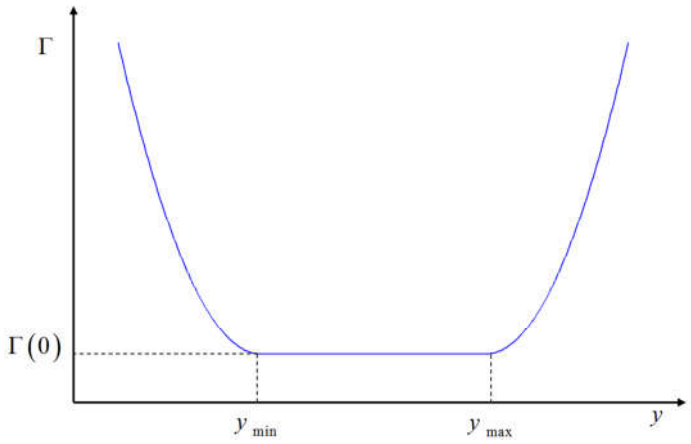

Figure 3.2 : Weight function $\Gamma_y(y)$.

By introducing the weight function $\Gamma_y(y)$, the OCP problem (3.13)-(3.17) is reformulated to have the following expression:

$$\min_{\Delta\hat{u}(t)} J\left(\Delta\hat{u}(t),\hat{y}(t),w(t)\right)=\min_{\Delta\hat{u}(t)}\sum_{j=N_1}^{N_2}\left\|\hat{y}(t+j\,|\,t)-w(t+j)\right\|_{\Gamma_y}^2+\sum_{j=1}^{N_u}\left\|\Delta\hat{u}(t+j-1)\right\|_{R(j)}^2 \tag{3.19}$$

Subject to:

$$\Delta\hat{u}(t+j-1)=0 \quad \text{for} \quad j>N_u \tag{3.20}$$

$$\Delta u_{min}<\Delta\hat{u}(t+j-1)<\Delta u_{max} \quad \text{for} \quad j\le N_u \tag{3.21}$$

$$u_{min}<\hat{u}(t)<u_{max} \tag{3.22}$$

b) Inputs constraints

The constraints (3.21) and (3.22) on the inputs could be combined into a single constraint on the inputs as follow:

For a given feasible solution ΔU_h associated with X_h ($h=1,...,n_{pop}$), we can write from (3.21):

$$\begin{gathered}\Delta u_{\min i}\le\Delta\hat{u}_i^h(t+j\,|\,t)\le\Delta u_{\max i}\ \Rightarrow\\ \Delta u_{\min i}+\hat{u}_i^h(t+j-1\,|\,t)\le\hat{u}_i^h(t+j\,|\,t)=\hat{u}_i^h(t+j-1\,|\,t)+\Delta\hat{u}_i^h(t+j\,|\,t)\le\Delta u_{\max i}+\hat{u}_i^h(t+j-1\,|\,t),\\ (i=1,..,m\,;j=0,..,N_u-1)\end{gathered} \tag{3.23}$$

If $j=0$, then $\hat{u}_i^h(t-1\,|\,t)$ is the already calculated control action $u_i(t-1)$.

On the other hand, the constraint (3.22) on the inputs magnitudes could also be written as:

$$u_{\min i}\le\hat{u}_i^h(t+j\,|\,t)\le u_{\max i}, \qquad (i=1,..,m\ ;j=0,..,N_u-1) \tag{3.24}$$

Hence, (3.23) and (3.24) could be combined into a single constraint as follows:

$$L_{\min i}^h(t+j)\le\hat{u}_i^h(t+j\,|\,t)\le L_{\max i}^h(t+j), \qquad (i=1,..,m\ ;j=0,..,N_u-1) \tag{3.25}$$

where

$$L^h_{\min i}(t+j)=\begin{cases}\Delta u_{\min i}+\hat{u}^h_i(t+j-1\,|\,t) & \text{if } \Delta u_{\min i}+\hat{u}^h_i(t+j-1\,|\,t)>u_{\min i}\\ u_{\min i} & \text{Otherwise}\end{cases}$$

$$L^h_{\max i}(t+j)=\begin{cases}\Delta u_{\max i}+\hat{u}^h_i(t+j-1\,|\,t) & \text{if } \Delta u_{\max i}+\hat{u}^h_i(t+j-1\,|\,t)<u_{\max i}\\ u_{\max i} & \text{Otherwise}\end{cases}$$

The inequality (3.25) defines the boundary of the feasible search space of the following OCP problem:

$$\min_{\Delta\hat{u}(t)} J\left(\Delta\hat{u}(t),\hat{y}(t),w(t)\right)=\min_{\Delta\hat{u}(t)}\sum_{j=N_1}^{N_2}\left\|\hat{y}(t+j\,|\,t)-w(t+j)\right\|^2_{\Gamma_y}+\sum_{j=1}^{N_u}\left\|\Delta\hat{u}(t+j-1)\right\|^2_{R(j)} \quad (3.26)$$

Subject to:

$$L^h_{\min i}(t+j)\le\hat{u}^h_i(t+j\,|\,t)\le L^h_{\max i}(t+j),\qquad (i=1,..,m\ ;j=0,..,N_u-1) \quad (3.27)$$

$$\Delta\hat{u}(t+j-1)=0\quad \text{for}\quad j>N_u \quad (3.28)$$

2.3.2. Why using the artificial bee colony ?

Although the *no free lunch theorem* [154] states that all optimization algorithms are somewhat equivalents, restricting their use within the NMPC framework clearly changes the circumstances. In fact, huge performance differences are observed when using different meta-heuristic algorithms or even when using different variants of the same algorithm.

Regardless of the used modelling approach, several meta-heuristic algorithms have been used to find high quality solutions to the NMPC optimization problem using as little computing requirements as possible. Among these algorithms, genetic algorithms [13, 14, 34, 189-191] and particle swarm optimization [46, 58, 186, 192-195] have been extensively used in developing NMPC algorithms. Recently, some other meta-heuristic algorithms, such as: ant colony optimization (ACO) [47], Bacterial Foraging Optimization (BFO) [48] and simulated annealing (SA) [189], were applied in solving the NMPC optimization problem. It is true that this shift to use other optimization algorithms than the GA and the PSO algorithm was partially motivated by the limitations of these algorithms. Nevertheless, the main reason could be attributed to the huge advances made in the optimization field. Indeed, an increasingly large number of new and mostly more efficient meta-heuristic optimization algorithms are continuously being proposed

The ABC, a recently introduced algorithm, has been a distinctive meta-heuristic algorithm. In fact, many comparative studies [49, 50] between the ABC algorithm and several other meta-heuristic algorithms such as GA, PSO and DE (Differential Evolution) have shown that the ABC performances are better or at least similar to the performances of these algorithms. Moreover, it was found that the ABC algorithm is more computing efficient, has better solutions accuracy, and is simpler than the other algorithms. A recent survey [51], related to the advances to the ABC algorithm and its applications, has indicated that more than 330 research papers were published within the scope of merely seven years of its creation. Given that more scholars adopt this algorithm, this numbers is expected to be increasing exponentially in the near future.

The ABC has been used in many fields; however its presence within the control engineering framework is quite limited. In fact, in the above mentioned survey, only nine research papers have been classified within the systems control field. Moreover, two-thirds of these papers deal solely with the design of the proportional integral differential (PID) controller and its parameters tuning.

Given the good performances of the ABC algorithm and its successful implementation in many engineering fields [196-201], and the fact that the control community has not yet fully exploited the ABC algorithm to solve the different control engineering problems, especially the nonlinear predictive control one, it is expected that its use in predictive control will provide good results.

3. Proposed control algorithms

Assuming the MPC design parameters $\left(N_1, N_2, N_u, \varepsilon \text{ and } \Gamma_y(0)\right)$ have been chosen and the fuzzy prediction model has been obtained, the basic steps of the proposed control algorithms are given in the following sub-sections:

3.1. Efficient PSO based controller

The proposed algorithm is based on the idea that takes advantage of both prior knowledge about the search space landscape and the fact that in most practical applications the dynamic optimization problem changes are gradual [165]. So it is, in most cases, logical to assume that once a solution was found, it is better to track it rather than to look for it every time the optimization problem

changes. In the used PSO algorithm, a group of candidate solutions is generated using the Gaussian distribution and the fittest one is chosen as the initial global best position of the PSO population. Then the quality of this initial solution is enhanced by favouring the search within an adaptive immediate neighbourhood around this initial global best position. This increases the efficiency of the algorithm in regard of the execution time, by using a small population size, and the quality of the solutions. It is known that the conventional PSO algorithm involves, in order to find satisfactory solutions, the use of a large population size to effectively probe the whole search space. The downside, of course, will be the huge computing requirement necessary to manage this population.

The different steps of the algorithm are drawn based on the following points:

1) The solutions obtained at the sampling time t $X_1^0 = \{u(t),\, u(t+1),\ldots,u(t+N_u-1)\}$ and $X_2^0 = \{u(t+1),\, u(t+2),\ldots,u(t+N_u)\}$ where $(u(t+N_u) = u(t+N_u-1))$ are both evaluated and the best one is chosen as the initial global best position $X_{best}^0(t+1) = \{u^0(t+1),\, u^0(t+2),\ldots,\, u^0(t+N_u)\}$ of the entire PSO population.

2) Using the Gaussian distribution, the particles $X_h\ (h = 1,...,n_{pop})$ are distributed around X_{best}^0 within the radius $r_d = [r_{d1}\ \cdots\ r_{dm}]^T$ according to the following expression:

$$u_i^h(t+j) = u_i^0(t+j) + r_{di}d_j\frac{\sqrt[N_u]{k_m}}{dist_d} \qquad (j = 0,..,N_u-1\ ;\ i = 1,..,m\ ;\ h = 1,...,n_{pop}) \tag{3.29}$$

where d_j is a random value from the normal distribution $N\ [0,1]$, k_m is a random value from the normal distribution $N\ [0,\frac{1}{3}]$ and $dist_d = \sqrt{\sum_{j=1}^{N_u} d_j^2}$.

The value of the radius r_d at the sampling time $t+1$ is given by:

$$r_d = |u(t+1) - u(t)| \tag{3.30}$$

To explore the whole search space, the particles positions are updated according to (2.4), (2.5) and (2.6).

Using a Gaussian distribution will ensure that the density of the particles gets higher around the chosen initial global best position; thus the optimal solution can be reached in few iterations and using a small number of particles. This is very interesting in the cases when the system outputs changes are small, the system is operating in the steady state or if we want to fine tune the solution. The use of a uniform distribution favours the diversity of the particles and increases the probability to find better solutions relatively far from the previous one. Since the number of PSO particles and the allowed iterations number should not be large, their initial positions carry a lot of influence in the quality of the solution.

To keep the algorithm from converging toward a local optimum, a minimal value $r_{dmin} = \left[r_{d1\min} \quad \cdots \quad r_{dm\min} \right]^T$ given by the following expression is imposed to r_d:

$$r_{dmin} = \alpha_1 \left| w(t) - y(t) \right| + \alpha_2 \tag{3.31}$$

where α_1 is an $n \times m$ matrix of scaling parameters and α_2 $(m \times 1)$ is used to impose a minimum value to r_{dmin} regardless of the tracking error (the value of r_{dmin} can have a fixed value by setting α_1 to zero).

If the current solution is far from the global optimum, the corresponding tracking error will be large and consequently the values of r_{dmin} will also be large. When $r_{di\ min} > r_{di}$ $(i = 1, \ldots, m)$, the particles are distributed around X^0_{gbest} within $r_{di\ min}$ instead of r_{di}. This will allow the algorithm to escape the current local optimum by looking for a better solution far from the actual one.

The resulting control algorithm is summarized in Algorithm 3.1:

Algorithm 3.1

$Sample\_ID = 0$

Do *// start a new sampling period*

$Sample\_ID{+}{+}$ *// Increment sampling period identifier*

Specify the desired reference trajectory between $t + N_1$ and $t + N_2$

Make the system measurements

$iter = 1$ *// Set the current number of iteration*

Compute the initial global best position X_{best}^{0} according to point 1)

Compute the value of r_d using (3.30) and the value of $r_{d\min}$ using (3.31)

If $r_{di} < r_{di\min}$

$$r_{di} = r_{di\min} \qquad (i = 1,...,\mathrm{m})$$

end if

Set $X_{Best} = X_{gbest}^{0}$

for $h = 1,...,n_{pop}$ *// Generate initial population*

Randomly initialize the velocity v_h

Using (3.29), distribute the particle X_h around X_{Best} within the radius r_d

end for

for $h = 1,...,n_{pop}$ *// Evaluate initial population*

Compute predictions between $t+N_1$ and $t+N_2$ for solution X_h

Compute the cost $F(X_h)$ associated with the solution X_h

Set $P_h = X_h$ *// Update personal best solution*

If $F(H_h) < F(X_{Best})$ *// Update the global best solution*

$$X_{Best} = X_h$$

end if

end for

Repeat *// Iterative process of the algorithm*

for $h = 1,...,n_{pop}$

Update the solution X_h using (2.4), (2.5) and (2.6)

Compute the predictions between $t+N_1$ and $t+N_2$ for solution X_h

Compute the cost $F(X_h)$

If $F(X_h) < F(P_h)$ *// Update personal best solution*

$P_h = X_h$

If $F(P_h) < F(X_{Best})$ *// Update the global best solution*

$X_{Best} = P_h$

end if

end if

end for

$iter++$ *// Update current number of iteration*

until ($iter \geq Max\_Iter$) *// Criterion for stopping the optimization*

$U_{opt}(t) = X_{Best}$ *// Collect best solution*

Send the first element $u(t)$ of $U_{opt}(t)$ to the system

wait *// Wait for next sampling period*

3.2. The ABCEV based controller

The predictive controller based on the proposed ABCEV algorithm is summarized in Algorithm 3.2.

Algorithm 3.2

$Sample\_ID = 0$

for $h = 1 : SN$ *// Initial population*

Choose an initial solution for X_h using (2.11)

end for

Do *// start a new sampling period*

$Sample\_ID++$ *// Increment sampling period identifier*

Specify the desired reference trajectory between $t + N_1$ and $t + N_2$

Make the system measurements

$iter = 1$ *// Set the current number of iteration*

Randomly choose one of the initial solutions as the best solution X_{Best}

for $h = 1 : SN$ *// Evaluate initial population*

Compute the predictions between $t + N_1$ and $t + N_2$ for the solution X_h

Compute the cost $F(X_h)$ associated with solution X_h

if $F(X_h) < F(X_{Best})$ *// Find the best solution in current population*

$X_{Best} = X_h$

end if

$Trial(h) = 1$ *// Number of unsuccessful attempts to enhance a food source*

end for

Repeat *// Iterative process of the ABCEV*

for $h = 1 : SN$ *// Employed bee phase*

Generate a new solution X_{hNew} using (2.12)

Compute the predictions between $t + N_1$ and $t + N_2$ for X_{hNew}

Compute the cost $F(X_{hNew})$

if $F(X_{hNew}) < F(X_h)$ *// Use greedy selection*

$X_h = X_{hNew}$

$Trial(h) = 1$

if $F(X_h) < F(X_{Best})$ *// Update the best solution*

$X_{Best} = X_h$

end if

else

$Trial(h) = Trial(h) + 1$

end if

end for

for $h = 1 : SN$ *// Probabilistic selection phase*

Evaluate the probability p_h using (2.9)

end for

for $h = 1 : SN$ *// Onlookers bee phase*

Using p_h, select a food source $X_r\left(r \in \{1, \ldots, SN\}\right)$ to exploit using the roulette wheel selection

Generates a new solution V_h in the neighbourhood of X_r using (2.12)

Compute predictions between $t+N_1$ and $t+N_2$ for V_h

Compute the cost $F(V_h)$

if $F(V_h) < F(X_r)$ *// Use greedy selection*

$X_r = V_h$

$Trial(r) = 1$

if $F(X_r) < F(X_{Best})$ *// Update the best solution*

$X_{Best} = X_r$

end if

else

$Trial(r) = Trial(r) + 1$

end if

end for

for $h = 1 : SN$ *// Scout bee phase*

if $Trial(h) > limit$ *// Randomly generate a new solution*

Generate a new solution X_h using (2.7)

$Trial(h) = 0$

end if

end for

$iter++$ *// Update current number of iteration*

until ($iter \geq Max\_Iter$) *// Criterion for stopping the optimization*

$U_{opt}(t) = X_{Best}$ *// Collect best solution*

Send the first element $u(t)$ of $U_{opt}(t)$ to the system

Sort the food sources X_h $(h = 1,\ldots,SN)$ according to their cost in ascendant manner $\left(F(x_{i1}) \leq F(x_{i2}) \leq \ldots \leq F(x_{iSN})\right)$

for $h = 3 : SN$ *// Keep best two food sources for the next sampling period*

Generate a new solution X_{hNew} using (2.11)

end for

wait *// Wait for next sampling period*

3.3. The EEABC based controller

The predictive controller based on the proposed EEABC algorithm is summarized in Algorithm 3.3.

Algorithm 3.3

$Sample\_ID = 0$

for $h = 1 : SN$ *// Initial population*

Choose an initial solution for X_h using (2.11)

end for

Do *// start a new sampling period*

$Sample\_ID{+}{+}$ *// Increment sampling period identifier*

Specify the desired reference trajectory between $t + N_1$ and $t + N_2$

Make the system measurements

$iter = 1$ *// Set the current number of iteration*

Randomly choose one of the initial solutions as the best solution X_{Best}

for $h = 1 : SN$ *// Evaluate initial population*

Compute the predictions between $t + N_1$ and $t + N_2$ for solution X_h

Compute the cost $F(X_h)$ associated with the solution X_h

if $F(X_h) < F(X_{Best})$ *// Find the best solution in current population*

$X_{Best} = X_h$

end if

$Trial(h) = 1$ *// Number of unsuccessful attempts to enhance a food source*

$ExplInd(h) = 0$ *// Exploitation index*

end for

Repeat *// Iterative process of the EEABC*

if $\mathrm{mod}(iter, 10 \times N_u) = 0$

Compute φ the ratio of successful mutation to all mutations

end if

if $\mathrm{mod}(iter, N_u) = 0$ *// Update the exploration rate*

Compute the new exploration rate α using (2.15)

end if

for $h = 1 : SN$ *// Employed bee phase*

Generate a new solution X_{hNew} using (2.16) and (2.12)

$ExplInd(h) = ExplInd(h) + 1$

Compute the predictions between $t + N_1$ and $t + N_2$ for X_{hNew}

Compute the cost $F(X_{hNew})$

if $F(X_{hNew}) < F(X_h)$ *// Use greedy selection*

$X_h = X_{hNew}$

$Trial(h) = 1$

if $F(X_h) < F(X_{Best})$ *// Update the best solution*

$X_{Best} = X_h$

end if

else

$Trial(h) = Trial(h) + 1$

end if

end for

for $h = 1 : SN$ *// Probabilistic selection phase*

Evaluate the probability p_h using (2.14)

end for

for $h = 1 : SN$ *// Onlookers bee phase*

Using p_h, select a food source $X_r\left(r \in \{1, \ldots, SN\}\right)$ to exploit using the roulette wheel selection

Generates a new solution V_h in the neighbourhood of X_r using (2.16) and (2.12)

$ExplInd(r) = ExplInd(r) + 1$

Compute the predictions between $t+N_1$ and $t+N_2$ for V_h

Compute the cost $F(V_h)$

if $F(V_h) < F(X_r)$ *// Use greedy selection*

$X_r = V_h$

$Trial(r) = 1$

if $F(X_r) < F(X_{Best})$ *// Update the best solution*

$X_{Best} = X_r$

end if

else

$Trial(r) = Trial(r) + 1$

end if

end for

for $h = 1 : SN$ *// Scout bee phase*

if $Trial(h) > limit$ *// Randomly generate a new solution*

Generate a new solution X_h using (2.7)

$Trial(h) = 0$

$ExplInd(h) = Min\{ExplInd(1), ..., ExplInd(SN)\}$

end if

end for

$iter++$ *// Update current number of iteration*

until ($iter \geq Max\_Iter$) *// Criterion for stopping the optimization*

$U_{opt}(t) = X_{Best}$ *// Collect best solution*

Send the first element $u(t)$ of $U_{opt}(t)$ to the system

Sort the food sources X_h $(h = 1, \ldots, SN)$ according to their cost in ascendant manner $(F(x_{i1}) \leq F(x_{i2}) \leq \ldots \leq F(x_{iSN}))$

for $h = 3 : SN$ *// Keep best two food sources for the next sampling period*

Generate a new solution X_{hNew} using (2.11)

end for

wait *// Wait for next sampling period*

4. Applications

In this section, the proposed control algorithms are implemented and their performances are evaluated against several other controllers. To this end, two highly nonlinear processes models have been considered: a SISO continuous stirred tank reactor (CSTR) and 4x4 MIMO industrial boiler.

4.1. CSTR

4.1.1. Process description

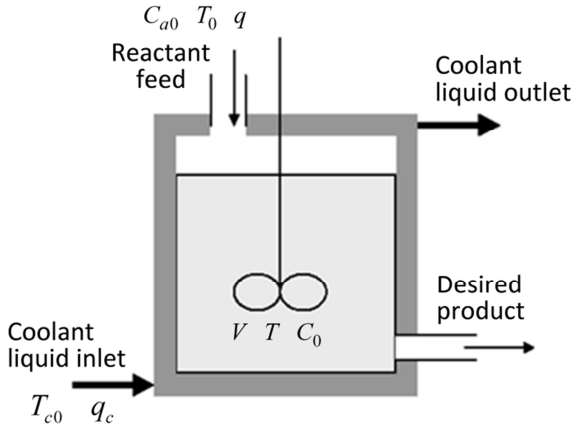

Figure 3.3 : Continuous stirred tank reactor (CSTR).

The process considered is the continuous stirred tank reactor (figure 3.3) within it a given product A will be converted into another product B via an exothermic chemical reaction. A coolant flow q_c (the control input) controls the reactor temperature which controls in its turn the concentration of the resulting product C_a (The process output). The process is described by the following equations:

$$\dot{C}_a(t) = \frac{q}{v}\left(C_{a0} - C_a(t)\right) - k_0 C_a(t)\, e^{-\frac{E}{RT(t)}} \tag{3.32}$$

$$\dot{T}(t) = \frac{q}{v}\left(T_0 - T(t)\right) + k_1 C_a(t)\, e^{-\frac{E}{RT(t)}} + k_2 q_c(t)\left(1 - e^{-\frac{k_3}{q_c(t)}}\right)\left(T_{c0} - T(t)\right) \tag{3.33}$$

The constant k_1, k_2 and k_3 are given by: $k_1 = \frac{\Delta H k_0}{\rho C_p}$, $k_2 = \frac{\rho_{cC_{pc}}}{\rho C_p v}$ and $k_3 = \frac{h_a}{\rho_c C_{pc}}$.

C_{a0}, T_0 represent respectively the inlet feed concentration and the temperature while T_{c0} is the coolant temperature and T the mixture temperature.v, E/R, k_0, k_1, k_2 and k_3 are thermodynamic and chemical constants. The numerical values for these constants can be found in [184].

4.1.2. Fuzzy identification

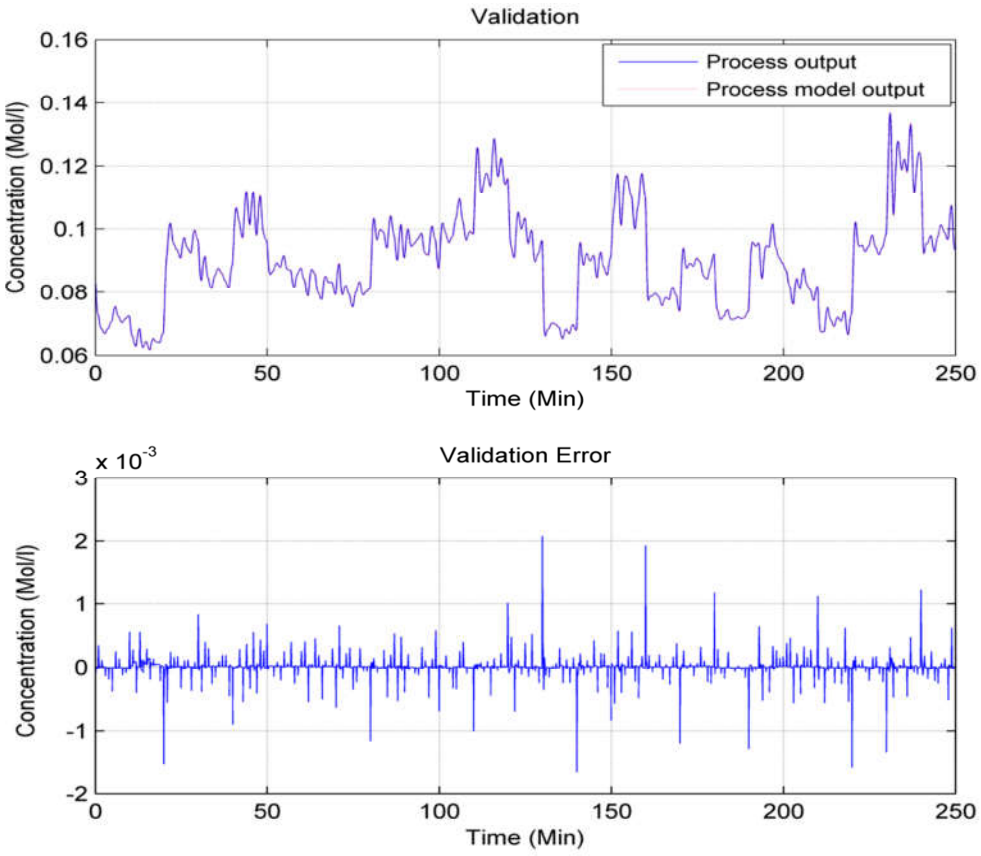

Figure 3.4 : Model and process validation for the CSTR.

A TS fuzzy model of the CSTR process is constructed using a dataset [202] containing 7500 input/output samples measurements of the CSTR process taken with a 6s interval. The following model structure is considered:

$$\hat{C}_a(t+1) = f\left(\hat{C}_a(t), \hat{C}_a(t-1), \hat{C}_a(t-2), q_c(t-1)\right) \tag{3.34}$$

where f represents the fuzzy relationship between the input and the output.

A two triangular membership functions per input have been adopted to construct the TS fuzzy model while a set of 5000 input/output samples are used to train it. The model validation is carried out using the remaining 2500 samples. Figure 3.4 illustrates the accuracy of the obtained model. It can be seen that the process and the model responses are superposed and the prediction error is very small.

4.1.3. Controllers implementation

The controlled output is the concentration $c_a(t)$ of the desired product and the control input is the coolant flow rate $q_c(t)$. The product concentration $c_a(t)$ must follow, as faithfully as possible, the reference trajectory given in figure 3.5.

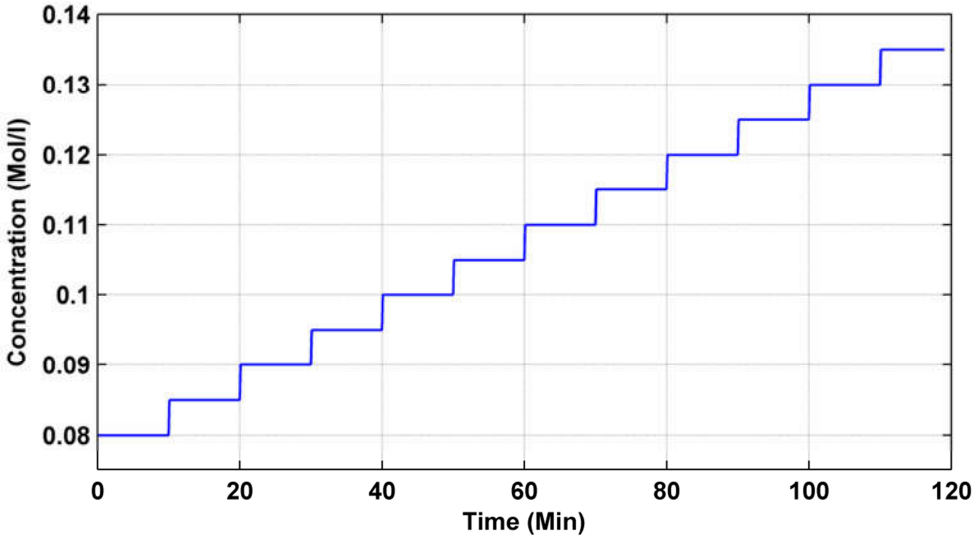

Figure 3.5 : Reference trajectory for the CSTR.

A sampling period of 6s and the following values of the design parameters are used to determine the control law of each controller:

$$N_1 = 1,\ N_2 = 10,\ N_u = 2,\ R = 2 \times 10^{-4} I^{Nu}$$

No constraints are imposed on the output, as such, the parameters $y_{\min}$ and $y_{\max}$ are respectively set to $-\infty$ and $+\infty$ while $\Gamma_{y_i}(0)$ has been set to 1. A

limitation on the possible coolant flowrate (mol/l) has imposed the following constraint on the computed values of the control signal:

$$89 \le q_c \le 111$$

The system evolves starting from the following initial conditions:

$$q_c = 97.227 \text{ l/min}, \quad C_a = 0.080 \text{ mol/l} \quad \text{and} \quad T = 443.339 \text{ K}$$

The following linear and nonlinear controllers are implemented:

- **Efficient PSO based controller** (control algorithm given by Algorithm 3.1).
- **Proposed ABCEV based controller** (control algorithm given by Algorithm 3.2).
- **Proposed EEABC based controller** (control algorithm given by Algorithm 3.3). The number of solutions to be updated in each iteration is computed using:

$$N_{update} = D \times \left(1 - \left(m / Max\_iter\right)\right) \tag{3.35}$$

- **Basic ABC based controller** (the original version of the ABC algorithm proposed in [119] is used).
- **Best-so-far ABC based controller** (the enhanced version of the basic ABC algorithm in [179] is used).
- **PSO algorithm based controller** (the PSO version described in section 4.2 of chapter 2 is used).
- **GA algorithm based controller** (the GA, with a mutation rate of 20%, described in section 4.1 of chapter 2 is used).
- **GPC based controller**: the control low is carried out using the following output error linear model of the considered system [184]:

$$\hat{C}_a\left(k+1\right) = \frac{1.653 \times 10^{-4} z^{-1}}{1 - 2.43z^{-1} + 2.4z^{-2} - 1.189z^{-3} + 0.269z^{-4}} q_c(k) \tag{3.36}$$

- **Analytical nonlinear approach**: it is based on the control algorithm found in [184]. The idea is to use an approximation of the free and forced

responses of the process using a TS fuzzy prediction model in order to obtain the control law analytically.

The population size is fixed to have 16 elements (bees, particles or individuals) and a number of 10 iterations is used. In the PSO and the GA based controllers, the solution of the previous sampling time is taken as a possible solution for the current one, while in the Basic ABC, the best-so-far ABC and the proposed ABC based controllers, the best two food sources from the previous sampling time are always taken as a possible solution for the current one.

Figures 3.6 and 3.7 give, respectively, the system output and the control signals for the proposed controllers while figures 3.8 and 3.9 give the system output for all considered controllers. The obtained results clearly show that the proposed controllers provide good control performance. It is shown in figure 3.9 that the performances of the EEABC based controller are slightly better than those of the ABCEV and the efficient PSO based controllers.

The Mean Cost Value (MCV) given by equation (3.37) is used to further compare the performance of the considered algorithms. The MCV is evaluated for several values of the population size (number of: bees, particles, individuals) and the number of iterations. The design parameters values are the same and each control algorithm is executed 10 times using the reference trajectory given in figure 3.5. The average values of the MCV are depicted in figure 3.10.

$$MCV = \frac{1}{N}\sum_{t=1}^{N} J\left(U\left(t\right)\right) \quad , \; N \text{ is the number of samples} \tag{3.37}$$

The three proposed controllers give the smallest average values of the MCV. The EEABC based controller has smaller values than the ABCEV and the efficient PSO based controller especially for the configurations with a relatively small population size. For the same population size, the basic ABC, the best-so-far ABC and the PSO based controllers require more iterations to converge toward the same MCV average values as in the proposed controllers. GA based controller has the poorest performances.

Another interesting observation is the fact that the MCV values of all ABC based controllers are smaller than those of the PSO and GA based controllers. The ABC variants based controllers are more efficient than those based on the

PSO and the GA algorithms with the exception of the proposed efficient PSO based controller.

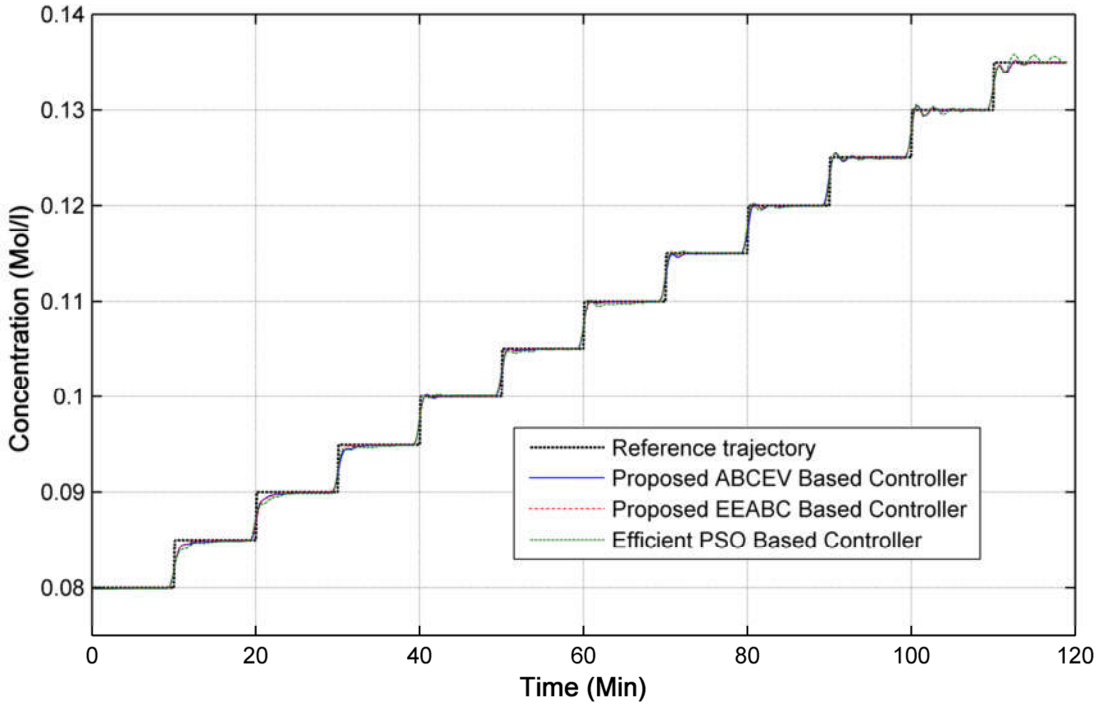

Figure 3.6 : Control performances of the proposed control algorithms for the CSTR.

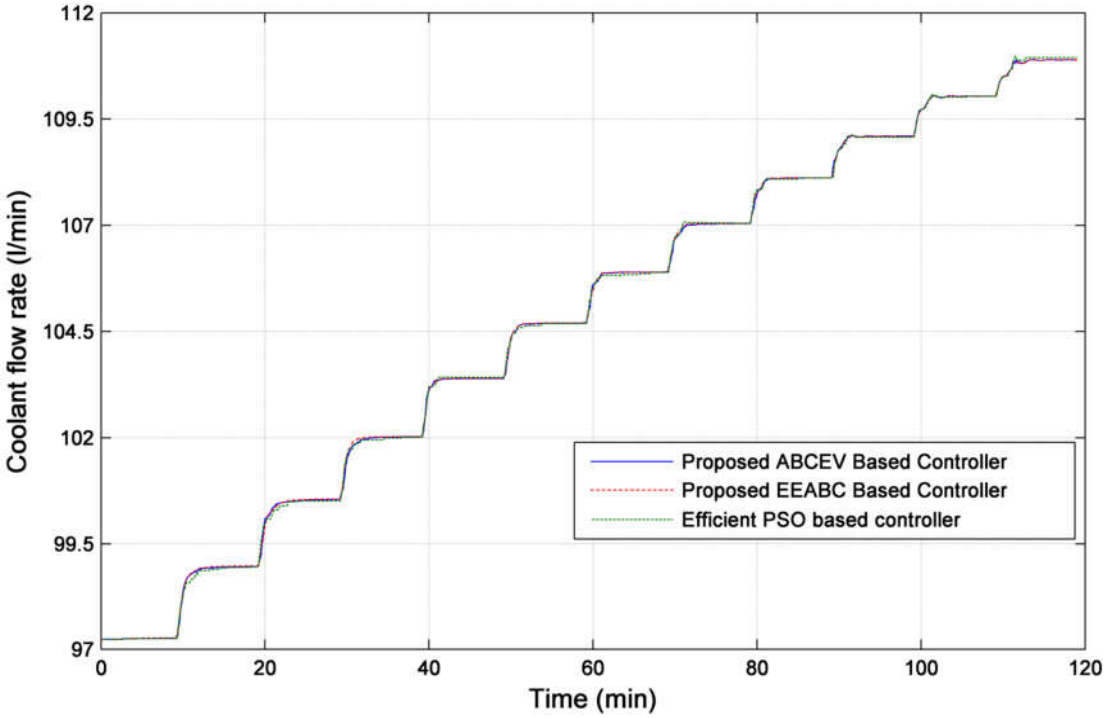

Figure 3.7 : Control signals of the proposed control algorithms for the CSTR.

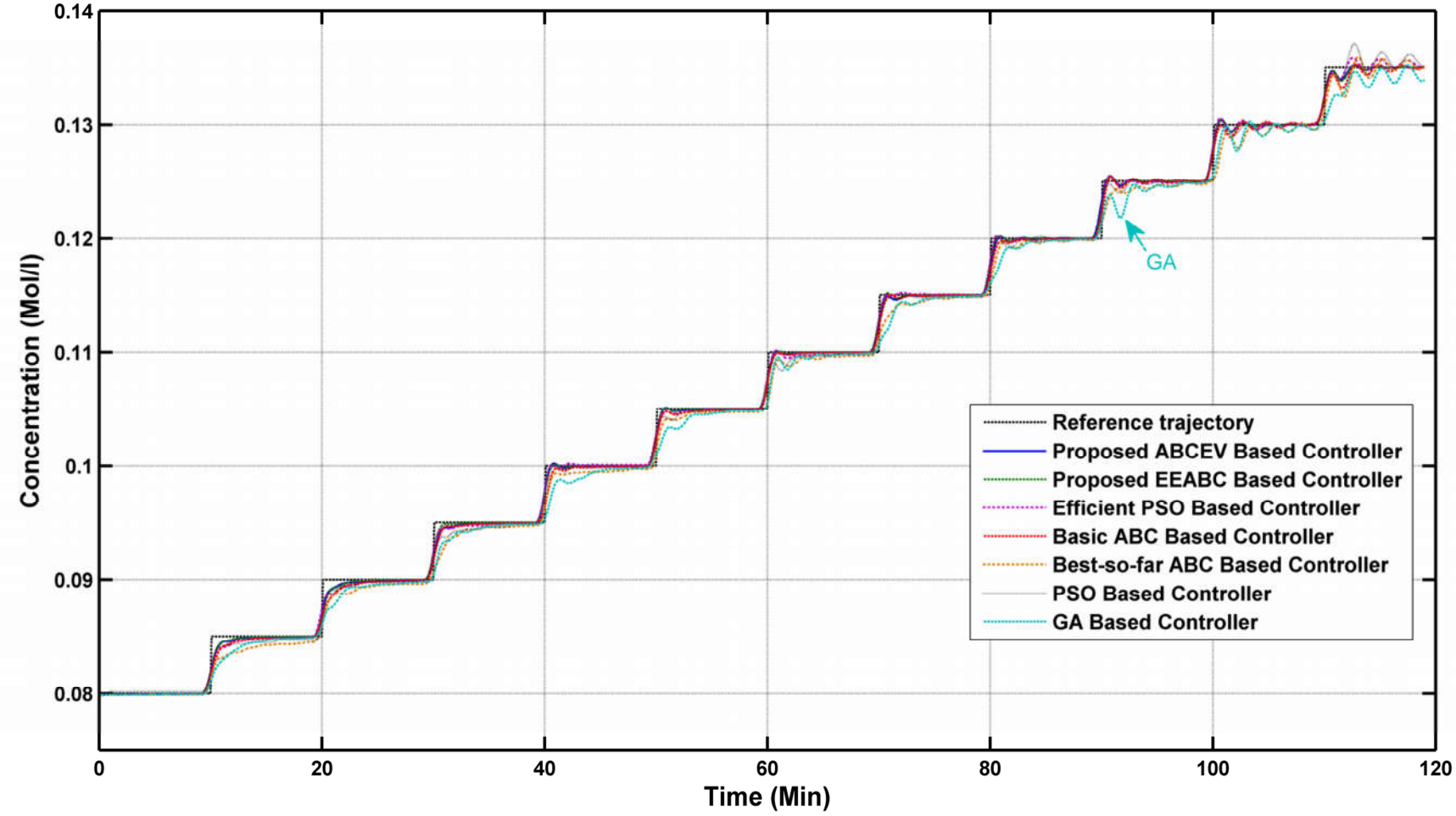

Figure 3.8 : Control performances of the different control algorithms for the CSTR.

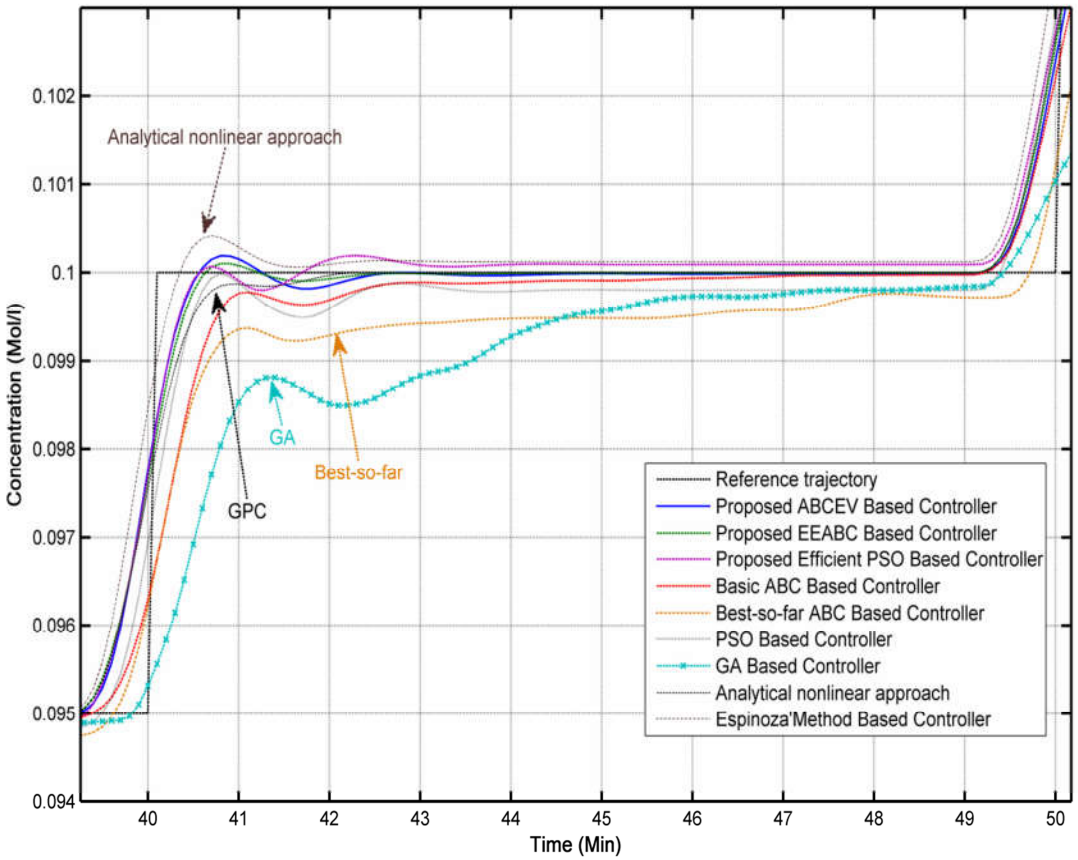

Figure 3.9 : Control performances of the different control algorithms between 40 and 50 min for the CSTR.

The execution time of a control algorithm is a very important parameter to evaluate its computing efficiency and real time implementation applicability. Table 3.1 gives the execution time of the considered NMPC algorithms for several configurations (population size x number of iterations); they were executed on an Intel Core i5 3.1 GHz (TM) based machine.

GA based controller appears to be the most computing efficient control algorithm. However, a larger population size, thus more computing power is required in order to obtain comparable performances with the other control algorithms. On the other hand, all the ABC based controllers are more computing efficient than those based on the PSO algorithm. The EEABC based controller has the lowest computing requirement, with at least 10% less computations compared to the other controllers. The proposed efficient PSO based controller and the PSO based controller have the same computing requirement.

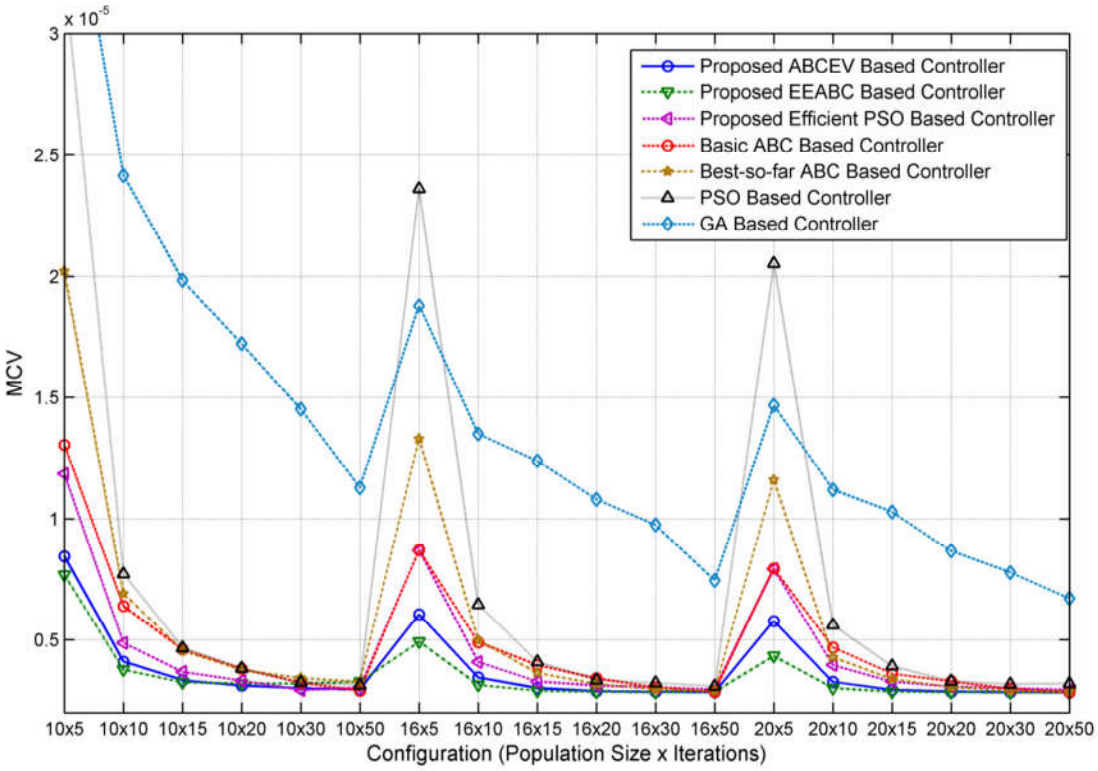

Figure 3.10 : MCV average values for the CSTR.

Table 3.1 : Execution time of the considered control algorithms for the CSTR.

Control algorithms	**Execution time (ms)**								**Computation Vs EEABC**
	Configuration (Population size x Iterations)								
	10x10	10x20	10x30	10x50	30x10	30x20	30x30	30x50	
Basic ABC	15.51	30.38	45.24	75.04	44.92	88.72	132.38	219.93	**+ 11.69 %**
Best-so-far ABC	15.72	30.76	45.69	75.21	45.49	89.25	133.97	221.95	**+ 12.71 %**
Proposed ABCEV	16.55	32.30	48.43	80.17	45.98	90.96	136.16	226.08	**+ 15.89 %**
Proposed EEABC	14.21	27.52	40.99	67.35	40.61	79.33	118.27	195.57	**0 %**
PSO	17.73	33.49	49.42	81.15	52.26	99.59	147.68	242.64	**+ 24.00 %**
Efficient PSO	17.46	33.37	49.34	80.94	51.17	98.76	146.48	241.32	**+ 23.12 %**
Genetic algorithm	12.4	23.84	35.11	57.62	35.53	68.09	101.48	167.44	**- 14.10 %**

4.2. Industrial Boiler

4.2.1. Process description

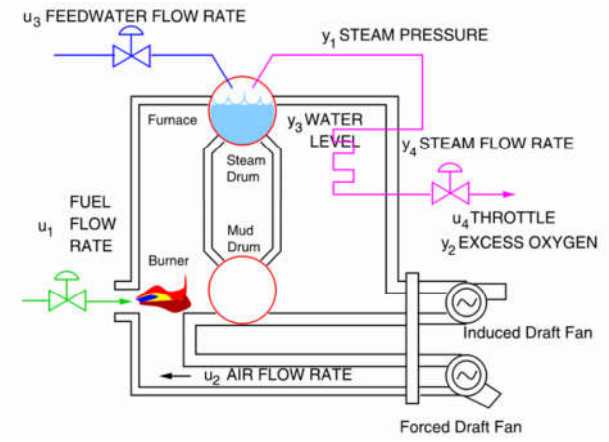

Figure 3.11 : the considered industrial boiler [184].

To further evaluate the performance of the proposed constrained nonlinear predictive controller, the control of the industrial boiler model given in [203] is considered. This model represents a dual fuel (oil and gaz) boiler in the Abott Power Plant in Champaign, Illinois, which could be used for both heating and electric generation. It is a MIMO process (figure 3.11) with four manipulated inputs (fuel flow rate, air flow rate, feed water flow rate and steam demand) and four measured outputs (pressure, oxygen level in the flue gas, drum water level and steam flow). The control objective is to maintain the pressure, the level of the water in the drum and the oxygen level in the flue gas at their desired values regardless of the steam flow rate provided by the boiler and any other disturbances such as the fluctuations in the heating value of the fuel or the variations in the ambient temperature. This boiler is capable of providing an industrial quality steam up to 22.096 kg/s. The used mathematical model includes perturbations and measurement noises, and is described by the following equations:

$$\begin{aligned}
\dot{x}_1(t) &= c_{11}x_4(t)x_1^{\frac{9}{8}}(t) + c_{12}u_1(t-\tau_1) - c_{13}u_3(t-\tau_3) + c_{14} \\
\dot{x}_2(t) &= -c_{21}x_2(t) + \frac{c_{22}u_2(t-\tau_2) - c_{23}u_1(t-\tau_1) - c_{24}u_1(t-\tau_1)x_2(t)}{c_{25}u_2(t-\tau_2) + c_{26}u_1(t-\tau_1)} \\
\dot{x}_3(t) &= c_{31}x_1(t) - c_{32}x_4(t)x_1(t) + c_{33}u_3(t-\tau_3) \\
\dot{x}_4(t) &= -c_{41}x_4(t) + c_{42}u_1(t-\tau_1) + c_{43} + c_{41}u_4(t)
\end{aligned} \tag{3.38}$$

$$
\begin{aligned}
y_1(t) &= c_{51}x_1(t-\tau_4) + n_1(t) \\
y_2(t) &= c_{61}x_2(t-\tau_5) + n_2(t) \\
y_3(t) &= c_{70}x_1(t-\tau_6) + c_{71}x_3(t-\tau_6) + c_{72}x_4(t-\tau_6)x_1(t-\tau_6) + \\
&\quad c_{73}u_3(t-\tau_3-\tau_6) + c_{74}u_1(t-\tau_1-\tau_6) + \\
&\quad \frac{\left[c_{75}x_1(t-\tau_6)+c_{76}\right]\left[1-c_{77}x_3(t-\tau_6)\right]}{x_3(t-\tau_6)\left[x_1(t-\tau_6)+c_{78}\right]} + c_{79} + n_3(t) \\
y_4(t) &= \left[c_{81}x_4(t-\tau_7)+c_{82}\right]x_1(t-\tau_7) + n_4(t)
\end{aligned}
\tag{3.39}
$$

where $\dot{x}_1$ is the drum pressure state $\left(Kgf/cm^2\right)$, y_1 is the measured drum pressure (PSI), y_2 and $\dot{x}_2$ are the measured excess oxygen level and its state (%), respectively, $\dot{x}_3$ is the system fluid density $\left(Kg/m^3\right)$, y_3 is the drum water level (in), y_4 is the steam flow rate $\left(Kg/s\right)$, u_1, u_2 and u_3 are, respectively, the fuel, air, the feed water flow rates which takes values in the interval [0 1] and $\dot{x}_4$ is the exogenous variable related to the steam flowrate demand. The variables n_i are the outputs of the first-order colored noise models driven by zero mean and unit variance white noise. They are considered to be unmeasured output disturbances. The remaining coefficients values could be found in Pellegrinetti and Bentsman [203] or in Espinosa and Vandewalle [204].

$$
n_1 = \frac{0.75s+0.1}{s+0.001}w_1 \;;\; n_2 = \frac{0.019s+0.001}{s+0.024}w_2 \;;\; n_3 = \frac{0.105s+0.038}{s+0.01}w_3 \;;\; n_4 = \frac{0.01s+0.0001}{s+0.001}w_4
$$

where n_i, $i = 1,...,4$ is a colored noise and w_i, $i = 1,...,4$ is the unit variance white noise.

The process is supposed to be initially operating around the following states:

$$
\begin{aligned}
x^0 &= \left[22.5,\ 2.5,\ 621.17,\ 0.6941\right]^T \\
y^0 &= \left[320,\ 2.5,\ 0,\ 9.3053\right]^T \\
u^0 &= \left[0.3227,\ 0.39503,\ 0.37404,\ 0\right]^T
\end{aligned}
\tag{3.40}
$$

The linear representation of the boiler around these operating states is described by:

$$\dot{x} = A_b\, x + B_b\, u$$
$$y = C_b\, x + D_b\, u \tag{3.41}$$

where

$$A_b = \begin{bmatrix} -0.005508 & 0 & 0 & -0.15872 \\ 0 & -0.20625 & 0 & 0 \\ -0.012156 & 0 & 0 & -0.566887 \\ 0 & 0 & 0 & -0.04 \end{bmatrix}, \quad B_b = \begin{bmatrix} 0.28 & 0 & -0.01348 & 0 \\ -9.374893 & 7.658411 & 0 & 0 \\ 0 & 0 & 0.731706 & 0 \\ 0.029989 & 0 & 0 & 0.04 \end{bmatrix}$$

$$C_b = \begin{bmatrix} 14.214 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0.322072 & 0 & 0.149410 & 11.148662 \\ 0.413307 & 0 & 0 & 19.274175 \end{bmatrix}, \quad D_b = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1.272 & 0 & -0.20797 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

There is a direct feedthrough between the output y_3 and the inputs u_1 and u_3 expressed by a non-zero D matrix in the state representation. This property will create some problems when considering the linear MPC controller [12]. In order to have D_b = 0, a delay was introduced to the inputs u_1 and u_3 with regards to the output y_3 (see section 4.2.3 for more details).

This process is inherently unstable. Essentially, a stabilisation scheme of the water level must be introduced to make any identification approaches possible by incorporating a proportional feedforward action (C2) of 0.0403 and a PI controller (C1) with Kp=0.258 and Ti=1.1026e-4, as it is illustrated in figure 3.12.

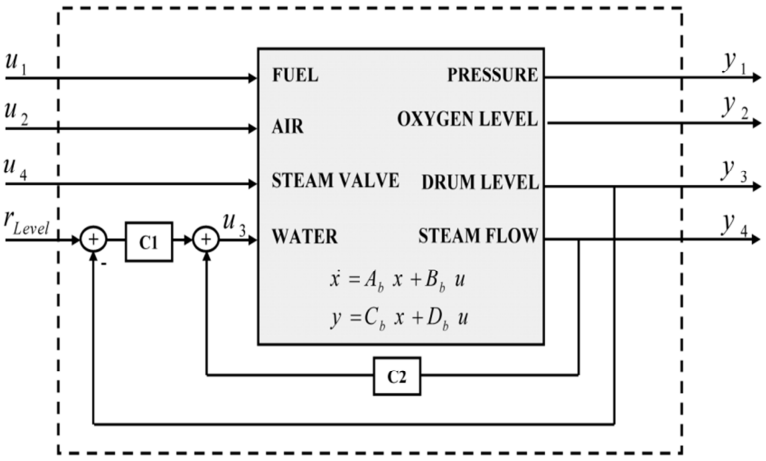

Figure 3.12 : Stabilization scheme for the industrial boiler.

4.2.2. Fuzzy identification

Four fuzzy multi inputs single output models are constructed to predict the future behaviour of the process. Two triangular membership functions per input and the models given by the following equations are used.

$$\hat{y}_1 = f_1\left(\hat{y}_1(k-1), u_1(k-5), \hat{x}_4(k-3)\right) \tag{3.42}$$

$$\hat{y}_3 = f_2\left(\hat{y}_3(k-1), u_1(k-12), r_{Level}(k-13), u_4(k-10), x_4(k-10)\right) \tag{3.43}$$

$$\hat{y}_4 = f_3\left(\hat{y}_4(k-1), u_1(k-4), u_4(k-2), x_4(k-2)\right) \tag{3.44}$$

$$\hat{x}_4 = f_4\left(\hat{x}_4(k-1), u_1(k-2), u_4(k-1)\right) \tag{3.45}$$

where f_i , $i = 1,...,4$ represent the fuzzy relationships between the inputs and the outputs.

The TS fuzzy models were trained using a custom built dataset generated using the mathematical model described by (3.38)-(3.40) and a sampling period of 3s. The validation results of the obtained models are given by figure 3.13.

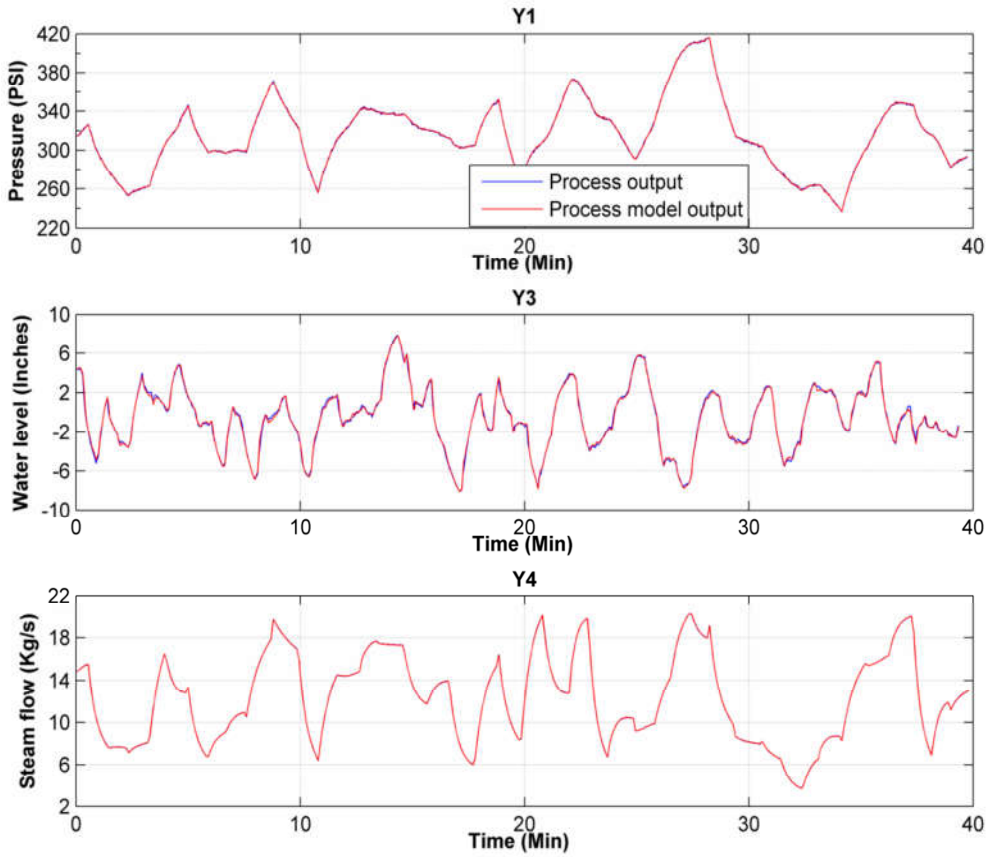

Figure 3.13 : TS fuzzy model validation for the industrial boiler (Blue solid line: Process output and red solid line: Process model output).

4.2.3. Controller implementation

Pellegrinetti [203] has indicated that the relationship between the fuel/air ratio ($FAR = u_2 / u_1$) and the oxygen level output is nearly linear. Consequently, the oxygen level output y_2 can be kept constant at a desired value by maintaining the corresponding FAR value constant. Given that u_1 affects all the systems outputs, the suitable way to maintain the FAR value constant would be to adjust the input u_2 depending on the value of u_1. To keep the desired oxygen level in the flue gas y_2 around 2.5% the FAR must be equal to 1.2241 and u_2 equals 1.2241 u_1.

The control objective is to force the steam flow rate y_4 to track a desired reference trajectory, while fulfilling the following constraints on the manipulated inputs:

$$\begin{aligned} \Delta u_{1\,Max} &= 0.1 \quad \text{and} \quad \Delta u_{2\,Max} = 0.1 \\ -4 < r_{Level} &< 4 \quad \text{and} \quad \Delta u_{r_{Level}\,Max} = 1 \\ -0.2 < u_4 &< 0.35 \quad \text{and} \quad \Delta u_{4\,Max} = 0.1 \\ 0 < u_3 &< 1 \end{aligned} \tag{3.46}$$

and the following constraints on the outputs:

$$\begin{aligned} y_1 &= 320 \\ y_2 &= 2.5 \\ y_3 &= 0 \end{aligned} \tag{3.47}$$

The MPC design parameters have the following values: $N_u = 2$ and R is given by $R = [100\ 0\ 0\ 0;\ 0\ 0\ 0\ 0;\ 0\ 0\ 2\ 0;\ 0\ 0\ 0\ 400]$. The values of the other parameters are gathered in table 3.2.

Table 3.2 : Design parameters.

	N_1	N_2	ε_i	$\Gamma(0)$	$y_{\min}$	$y_{\max}$
$\hat{y}_1$	1	10	1	0.0085	318	322
$\hat{y}_3$	10	20	0.7	0.0065	-1.5	1.5
$\hat{y}_4$	1	10	0	1.006	/	/
$\hat{x}_4$	1	10	0	0	/	/

The following MPC based controllers are considered:

- **Efficient PSO based controller** (control algorithm given by Algorithm 3.1).
- **Proposed ABCEV based controller** (control algorithm given by Algorithm 3.2).
- **Proposed EEABC based controller** (control algorithm given by Algorithm 3.3). The number of solutions to be update in each iteration is computed using:

$$N_{update} = D \times \left(1 - \left(m / Max\_iter\right)\right) \tag{3.48}$$

- **Basic ABC based controller** (the original version of the ABC algorithm proposed in [119] is used).
- **Gbest-guided ABC based controller** (the enhanced version of the basic ABC algorithm proposed in [153] is used).
- **The best-so-far ABC based controller** (the enhanced version of the basic ABC algorithm in [179] is used).
- **PSO algorithm based controller** (the PSO version described in section 4.2 of chapter 2 is used).
- **Linear MPC strategy**: to implement the linear MPC, a linear representation of the process must be derived. First we have to remove the direct feedthrough between the output y_3 and the inputs u_1 and u_3. As such, the linear state space representation of (3.41) is slightly modified by introducing a unit delay in these inputs (figure 3.14). The obtained system is then linearized around the operating state given by (3.40), and discretized using a sampling period of 3s. The following discrete state space representation is obtained:

$$\begin{aligned} \dot{x}(k+1) &= A_b\, x(k) + B_b\, u(k) \\ y(k+1) &= C_b\, x(k) \end{aligned} \tag{3.49}$$

where:

$$A_b = \begin{bmatrix} 0.9861 & 0 & 0.001485 & -0.3706 \\ 0 & 0.5386 & 0 & 0 \\ -0.1736 & 0 & 0.9187 & -5.665 \\ 0 & 0 & 0 & 0.8869 \end{bmatrix}, \quad B_b = \begin{bmatrix} 0.8172 & 0 & -0.009937 & -0.02264 \\ -20.97 & 17.13 & 0 & 0 \\ -0.3384 & 0 & 0.5441 & -0.3525 \\ 0.08478 & 0 & 0 & 0.1131 \end{bmatrix}$$

$$C_b = \begin{bmatrix} 14.21 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0.3221 & 0 & 0.1494 & 11.15 \\ 0.4133 & 0 & 0 & 19.27 \end{bmatrix}$$

Some design parameters have been slightly modified in order to accommodate the linear controller. The new values are: $N_2 = 15$ for all outputs and $Q = [0.4\ 0\ 0\ 0;\ 0\ 0\ 0\ 0;\ 0\ 0\ 0.5\ 0;\ 0\ 0\ 0\ 2.006]$.

For the linear MPC strategy, the constraints are only considered on the inputs variables and their increments.

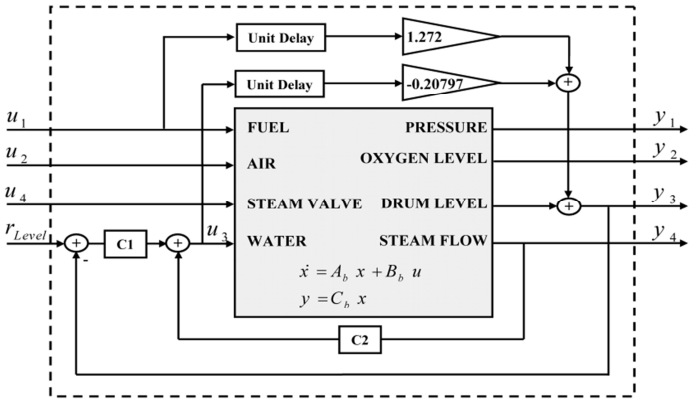

Figure 3.14 : Stabilization scheme for the boiler without direct feedthrough.

A population size of 16 elements (artificial bees or particles) and 10 iterations are used for all these controllers. At the end of each sampling time, the best two food sources are always kept as possible solutions for the next sampling time while in the case of the PSO based algorithm, only the best solution is kept. The EEABC based controller is implemented using $\alpha = 0.05$ and the *limit* expression of (2.13).

The system outputs for the proposed control algorithms and the linear MPC are given by figures 3.15 and 3.16 while the corresponding control signals are depicted by figure 3.17. It can be seen that the proposed controllers presents good performance.

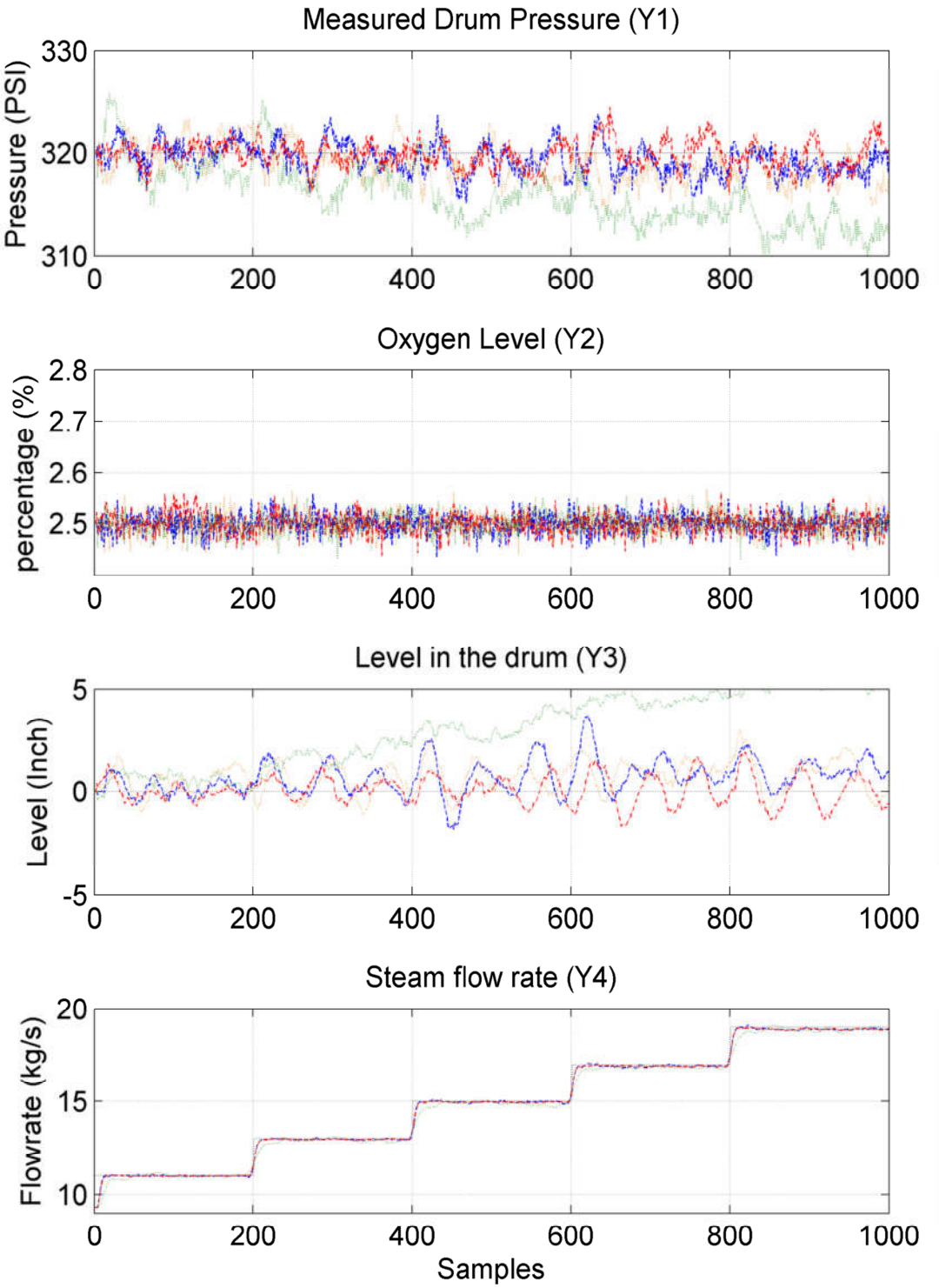

Figure 3.15 : Response of the boiler using the proposed control algorithms and the linear MPC controller (Blue dashed line: ABCEV based controller; the red dashed dotted is the EEABC based controller; brown dotted line is the efficient PSO based controller and the green dotted is the linear MPC based controller).

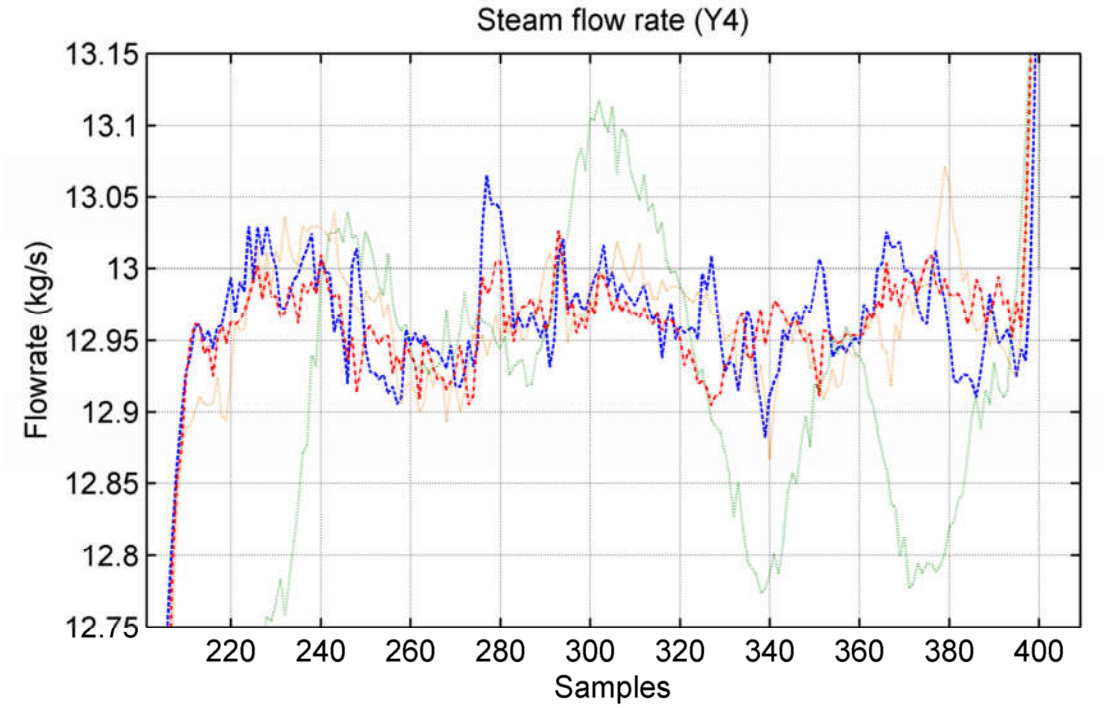

Figure 3.16 : Response of the boiler for the different proposed control algorithms and the linear MPC controller between the 200th and 400th samples for the output y_4 (Blue dashed line: ABCEV based controller; the red dashed dotted is the EEABC based controller; brown dotted line is the efficient PSO based controller and the green dotted is the linear MPC based controller).

To assess the performances of each proposed control algorithms, the Mean Cost Value (MCV) given by (3.37) is used. The MCV is evaluated for several values of the population size (number of bees or particles) and the number of iterations. The design parameters values are the same and each control algorithm is executed 15 times. The average values of the MCV are depicted in figure 3.18.

It can be seen from figure 3.18 that the proposed ABCEV and EEABC control algorithms give the smallest average values of the MCV, although the EEABC is slightly better than the ABCEV. For the same population size, the remaining control algorithms require more iterations to converge toward the same MCV average values as those of the EEABC based controller.

The proposed efficient PSO based controller gives better result than the basic PSO based algorithm; however, the ABC based controllers are better.

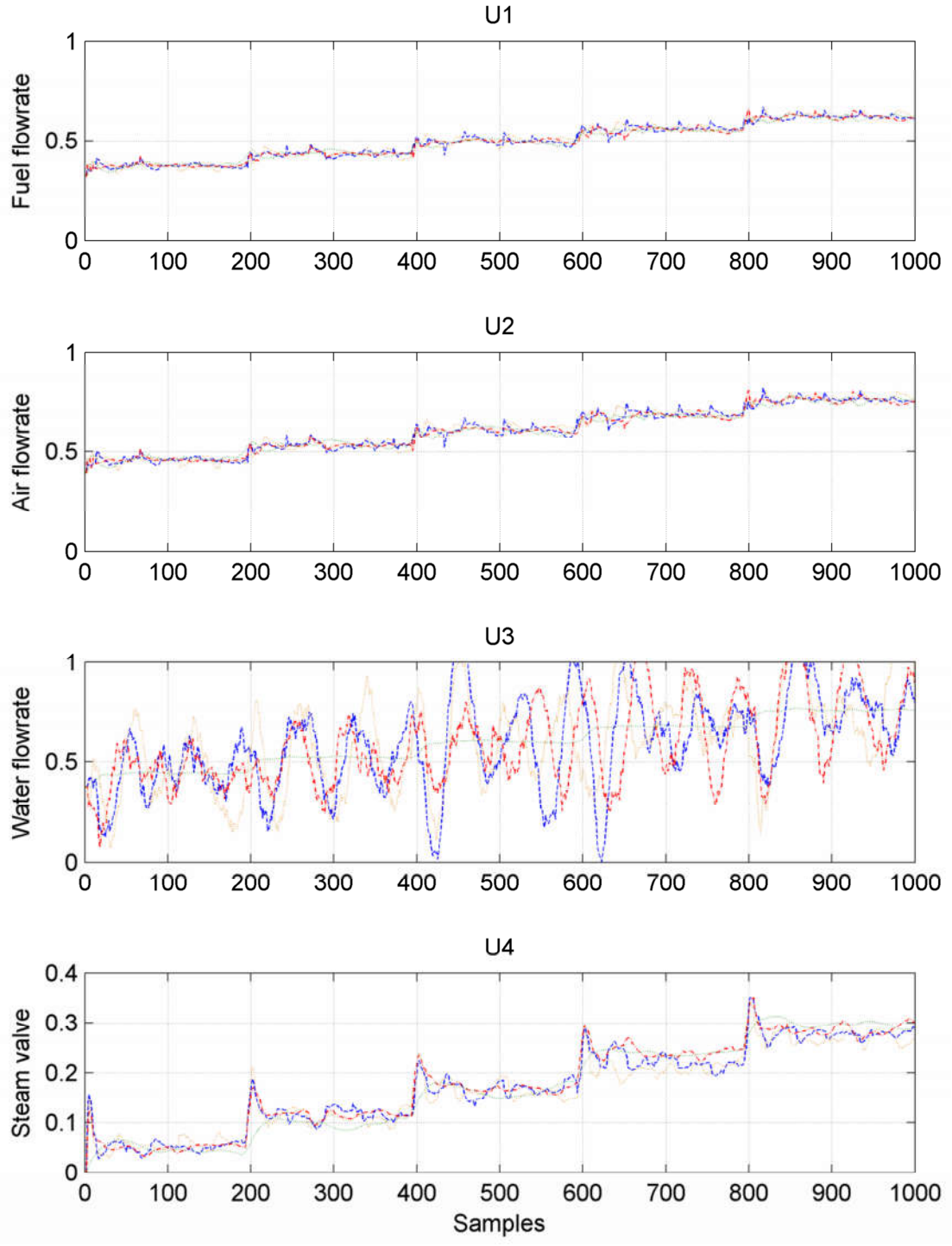

Figure 3.17 : Control signals for the considered controllers and the linear MPC controller (Blue dashed line: ABCEV based controller; the red dashed dotted is the EEABC based controller; brown dotted line is the efficient PSO based controller and the green dotted is the linear MPC based controller).

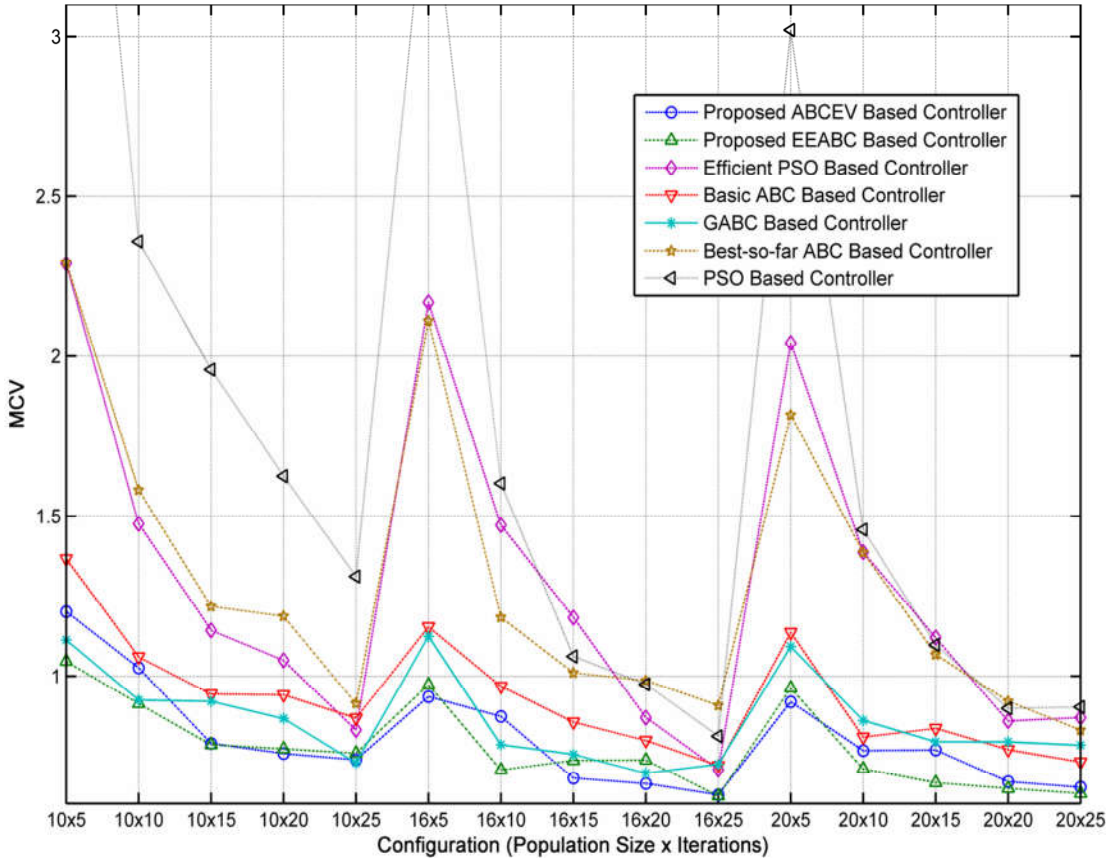

Figure 3.18 : MCV average values for the industrial boiler.

Table 3.3 gives the execution time of the considered NMPC controllers using several population sizes and number of iterations. These algorithms were executed on an Intel Core i5 3.1 GHz (TM) based machine.

The computing time of the EEABC and the ABCEV based controllers is, respectively, around 4% and 10% more important than those of the other ABC based controllers. This could be due to the fact that the proposed ABC based controllers are generating more scout bees because of the reduced adopted *limit* expression, thus both algorithms have more possible solutions to evaluate compared to the other ABC variants.

On the other hand, all ABC based controllers are more computing efficient than those based on the PSO algorithm.

Table 3.3 : Execution time of the considered control algorithms for the industrial boiler.

Control algorithms	Execution time (ms)								Computation Vs EEABC
	10x5	10x10	10x20	10x30	20x5	20x10	20x20	20x30	
Basic ABC	65.04	124.49	242.47	359.58	129.52	247.19	483.07	718.53	**- 3.60 %**
GABC	65.22	125.43	243.15	359.74	130.08	248.31	486.09	720.59	**- 3.25 %**
Best-so-far ABC	64.5	125.84	238.63	355.22	128.57	245.55	477.91	712.2	**- 4.48 %**
Proposed ABCEV	71.63	137.30	268.9	401.46	142.36	275.67	535.8	796.3	**+ 6.95 %**
Proposed EEABC	66.95	128.05	250.9	374.28	133.11	254.73	500.68	749.75	**0 %**
PSO	82.28	151.82	292.61	432.06	164.42	305.2	585.88	863.59	**+ 17.06 %**
Efficient PSO	77.28	141.57	272.4	402.36	152.36	281.58	541.62	800.81	**+ 8.60 %**

5. DSP-based implementation

To experimentally validate the obtained results, a DSP based experimental setup has been used to implement several linear and non-linear control strategies and then compare their performances.

5.1. Description of the eZDSP2812 test bench

Figure 3.19 illustrates the test bench used in this implementation. It contains an eZdsp F2812 starter kit based on the Texas Instrument *TMS320F2812* DSP, a brushed DC motor equipped with an incremental encoder, a power and a protection circuits.

Using Code Composer Studio (CCS), the Internal Development Environment (IDE) provided by Texas Instrument, the different considered control algorithms are coded, compiled and then sent via a parallel cable to the DSP. The control results of the DC motor speed are saved in a file and then plotted using plotting software.

Figure 3.19 : DSP based test bench.

5.2. Fuzzy identification and controllers implementation

To obtain a Takagi-Sugeno FIS model of the free load DC motor, a set of input/output data is generated using a normal random input signal. Triangular membership function and 16 fuzzy rules were used to construct the model given by:

$$\hat{\Omega}(t+1) = f\left(\Omega(t), \Omega(t-1), \Omega(t-2), V(t)\right) \tag{3.50}$$

Where Ω (Tr/min) and V (Volts) represent respectively the motor speed (output) and supply voltage (input).

A sampling period of 0.1s and the following values of the design parameters are used to determine the control law of each controller:

$$N_1 = 1\ ,\ N_2 = 8\ ,\ N_u = 2\ ,\ \lambda = 10^4$$

The supply voltage (control signal) of the DC motor has been limited using the following constraint:

$$6 \leq V(t) \leq 17$$

The control objective is to force the process output (velocity of the DC motor) to follow the reference trajectory given in figure 3.20 as faithfully as possible, while satisfying any imposed constraints.

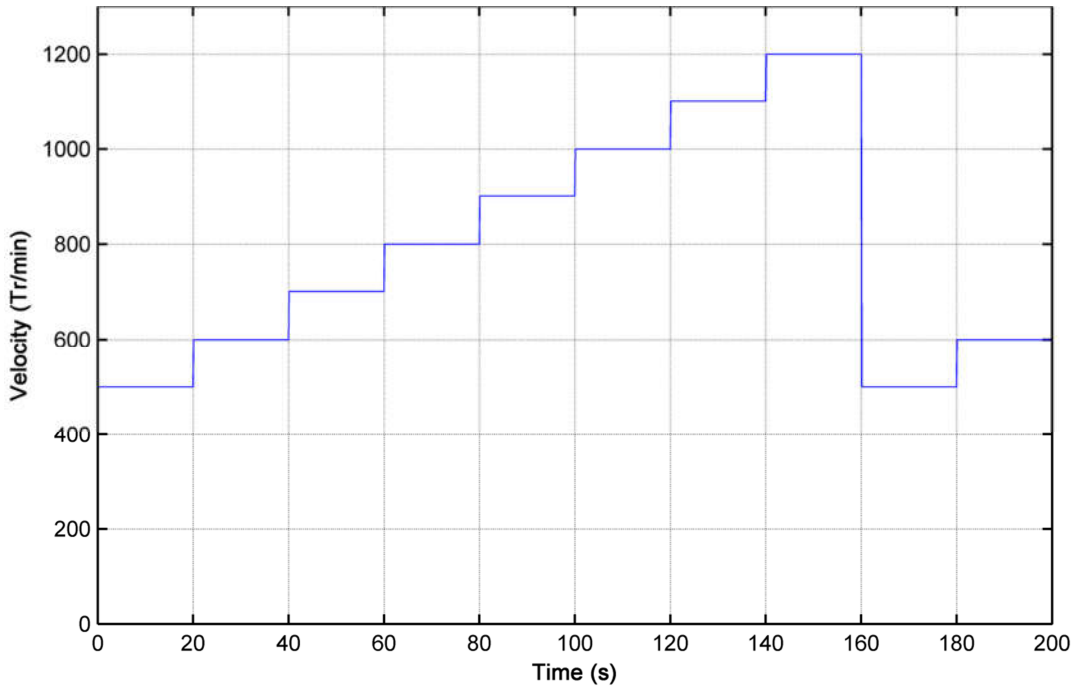

Figure : 3.20 Reference trajectory for the DSP based implementation.

5.3. Comparative study

The following control strategies have been considered:

- **Efficient PSO based controller**: The implemented control algorithm is described by Algorithm 3.1 with $\alpha_1 = 0.001375$ and $\alpha_2 = 0.5$.
- **Proposed ABCEV based controller** (control algorithm given by Algorithm 3.2).
- **Proposed EEABC based controller** (control algorithm given by Algorithm 3.3). The number of solutions to be updated in each iteration is computed using:

$$N_{update} = D \times \left(1 - \left(m / Max\_iter\right)\right) \tag{3.51}$$

- **Basic ABC based controller** (the original version of the ABC algorithm proposed in [119] is used).
- **Best-so-far ABC based controller** (the enhanced version of the basic ABC algorithm by [179] is used).

- **GABC based controller** (the enhanced version GBest-guided ABC [153] is used).
- **MABC based controller** (the enhanced ABC version found in [180] is used).
- **PSO algorithm based controller** (the PSO version described in section 4.2 of chapter 2 is used).
- **GPC controller** the control law is built using the following output error linear model of the considered system:

$$\hat{\Omega}(t+1) = \frac{13.6 \times 26.3 z^{-1}}{1 - 0.86 z^{-1} + 0.1 z^{-2}} V(t) \tag{3.52}$$

- **PID controller** (the implemented controller also uses the linear model of (3.52)).

The population based controllers are implemented with a population of 6 elements (particles or artificial bees) and 5 iterations. In the PSO based controller the solution of the previous sampling time is taken as a possible solution for the current one while in the ABC controllers, the best two food sources from the previous sampling time are always taken as a possible solutions for the current one.

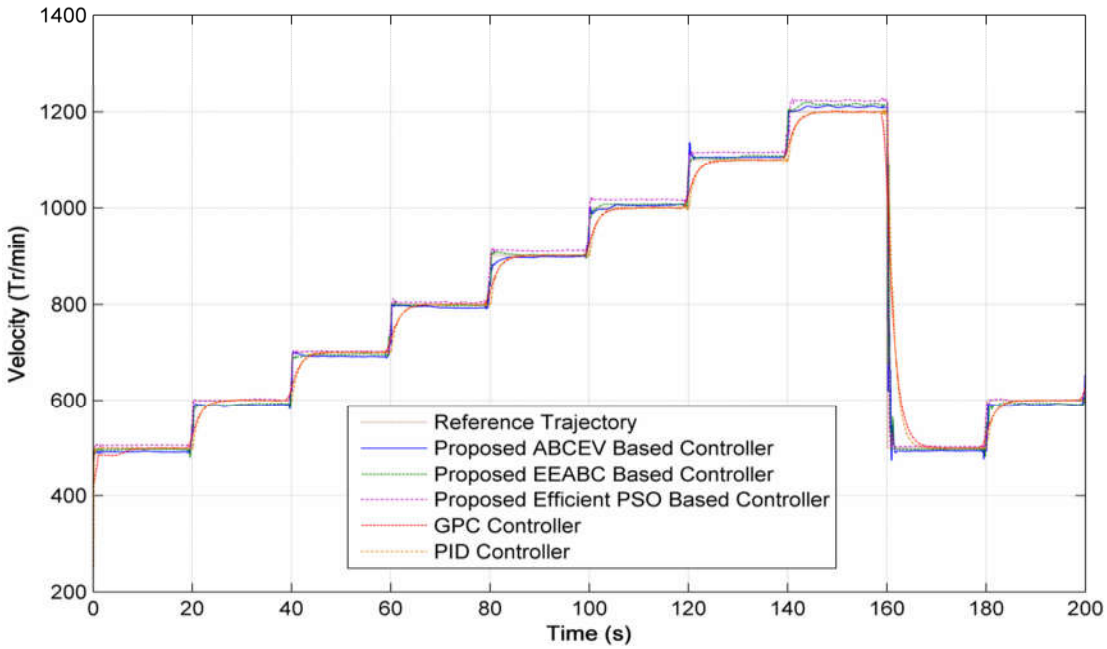

Figure 3.21 : Speed of the free load motor for the implemented controllers.

The control results for some of the implemented controllers are given in figure 3.21. A fast response is obtained in the case of the proposed controllers with a small tracking error.

To assess the robustness of each control algorithm, the experimental setup, where a metallic disc of 1700g attached directly to the motor's shaft, is considered. The obtained results are given by figures 3.22 and 3.23. Compared to the results of figure 3.21, the responses of figure 3.22 are quite slow. This was expected; the introduction of the load reduces the motor acceleration.

A severe degradation of the control performance is observed in the case of the PID controller and the GPC, with the appearance of an oscillatory behaviour with relatively huge overshoots. The proposed controllers' responses were also affected. However, no overshoot is observed. Figure 3.23, presents the associated control signals.

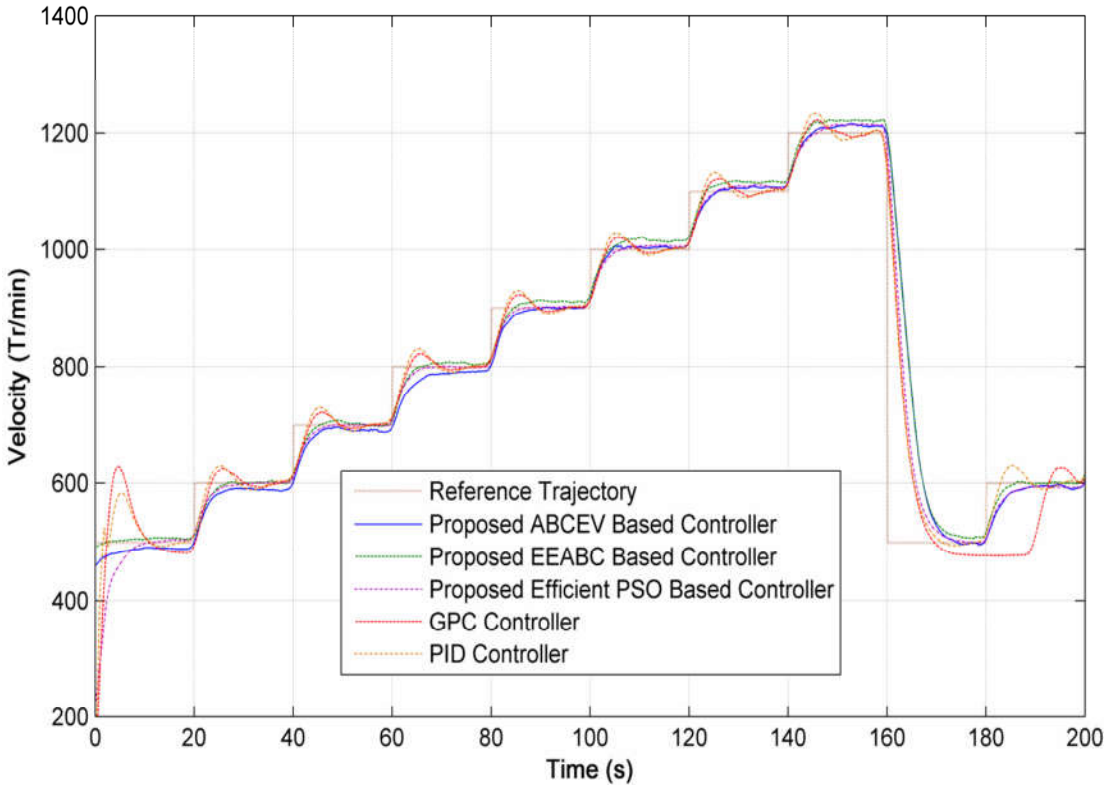

Figure 3.22 : Speed of the motor for the implemented controllers.

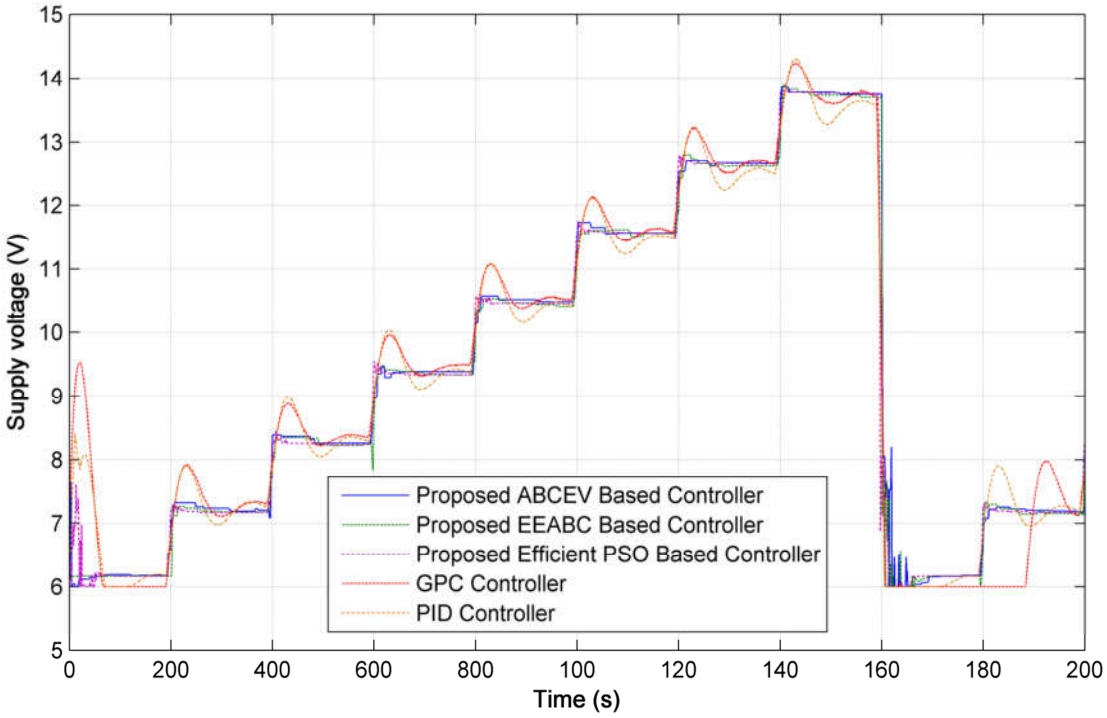

Figure 3.23 : Control signals for the implemented controllers.

The computing complexity of all of the implemented control algorithms has been evaluated by measuring the DSP computing time required to solve the NMPC control problem when considering a single iteration. The results, obtained for multiple population sizes, are gathered in table 3.4.

The proposed EEABC is the most computing efficient control algorithm with around 1% less computations than the other ABC based controllers and around 20% less computations than the PSO algorithms. On the other hand, the proposed ABCEV algorithm along with the remaining ABC variants, have similar computing requirements.

The proposed efficient PSO based controller has the largest computing requirement with around 22% more computations than the EEABC. In fact both PSO algorithms need around 20% more computation than any ABC based algorithm.

These results have further validated the computing efficiency of the ABC algorithm in general and the proposed EEABC variant in particular, especially against PSO based algorithms.

Table 3.4 : Execution time of the considered control algorithms (DSP).

Control algorithms	Execution time (ms)								Computation Vs EEABC
	Population size								
	6	10	16	24	36	48	72	100	
Basic ABC	31.7	51.4	80.8	120.4	179.3	237.7	356.2	493.2	**+ 1.04 %**
GABC	31.7	51.4	80.4	120.4	178.1	237.6	355.4	491.1	**+ 0.74 %**
Best-so-far ABC	31.8	51.6	81.4	121.1	180.5	239.9	357.5	496.9	**+ 1.69 %**
MABC	31.8	51.4	81.4	120.7	179.9	239.2	355.7	493.3	**+ 1.21 %**
Proposed ABCEV	33.7	53.2	82.7	120.2	179.4	238.1	355.4	493.4	**+ 1.39 %**
Proposed EEABC	33.7	51.6	81.2	118.6	177.2	234.3	351.8	486.4	**0 %**
PSO	37.3	60.2	94.9	141.8	213.1	280.6	420.8	581.3	**+ 19.23 %**
Efficient PSO	39.4	62.8	98.9	145.4	216.1	286.9	427.6	594	**+ 21.91 %**

6. conclusion

Within this chapter, the formulation of constrained fuzzy predictive control based on the meta-heuristic algorithms has been presented along with three proposed control algorithms.

The conducted comparative studies on two control problems and a DSP based experimental setup have indicated that the proposed algorithms have presented good performances. The efficient PSO algorithm has outperformed the conventional PSO algorithms in all three considered comparative case studies in regard of the quality of the solutions; however, within regard to its computing complexity, the lead is less apparent. The proposed ABC algorithms on the other hand have outperformed their ABC counterparts, especially the EEABC algorithm, which has generated better solutions with less computing complexity compared to all considered algorithms.

We have also confirmed the computing efficiency of the ABC algorithm against other conventional algorithm such as the PSO. The obtained results have clearly indicated that all ABC based algorithms are more computing efficient than those based on the PSO algorithm.

CONCLUSION

The aim of this thesis was to develop simple and efficient nonlinear predictive control algorithms. The idea was to use the meta-heuristic approach to find, in a reasonable computational time, an accurate suboptimal solution to the nonlinear optimization problem. Mainly, particle swarm optimization and artificial bee colony approaches were envisaged. Without loss of generalization to other models, such as neural networks models, Takagi-Sugeno fuzzy models were used in developing these algorithms.

The MPC is considered as one of the most successful advanced control strategies. Its attraction is due to its ability to handle complex control problems which involve multivariable process interactions, constraints in the system variables, non-minimum phase behaviour, and variable or unknown time delays. Despite the large number of advanced linear predictive control algorithms that were developed, the need for efficient control algorithms that can efficiently handle highly nonlinear systems is still an open research subject. Indeed, using a NMPC scheme should result in a substantial increase in the overall control performances. This aspect is very desirable. However, fully introducing the nonlinear model in the MPC formulation leads to a non-convex optimization problem that cannot be resolved using the well-established techniques of convex optimization.

Solving nonlinear optimization problem using analytical approaches is generally unfeasible even when no constraints are present while the deterministic numerical approaches are less effective and could easily be trapped in local minima. Recently, several works have showed that stochastic meta-heuristic algorithms are powerful tools that can efficiently handle complex nonlinear optimization problems. These algorithms have known a quick dissemination within the academic and industrial society, and several successful implementations to solve multiple engineering problems are continuously developed by an increasingly active research community. This success could be attributed to the fact that the meta-heuristic algorithms are simple, flexible, derivative-free, generally able to handle local optima and generate high quality solutions in a reasonable amount of time. In fact, due to their heuristic and random nature, these

algorithms, that have better searching capability than the classical approaches, are unaffected by initial conditions and can handle almost any optimization problem, albeit discrete or continuous. Despite their success, the meta-heuristic based algorithms, like any other approaches, are not perfect optimization techniques. In fact, several factors could heavily influence the performance of any such algorithm, if care is not taken. One such important factor is the algorithm ability to both explore (global search) the search space looking for regions of interest, and to exploit (local search) these regions in order to locate optimum solutions. Ideally, an optimization algorithm that has both of these characteristics fully integrated should be designed. However, the exploration and the exploitation are somewhat exclusive characteristics.

Although, numerous meta-heuristic variants exist, our interest has been directed to the artificial bee colony (ABC) algorithm. This choice was based on the result of numerous comparative studies between the ABC algorithm and several other meta-heuristic algorithms such as GA, PSO and DE (Differential Evolution). The comparative results have clearly indicated that the ABC performances are better or at least similar to the performances of these algorithms. Moreover, they were found to be more computing efficient, to have better solutions accuracy, and to be simpler than the other algorithms. Nevertheless, some deficiencies were also observed; the ABC algorithm has poor convergence rate in some situations and cannot escape local optimum when solving some of the complex multimodal function optimization problems. To overcome these deficiencies and further improve the exploration/exploitation equilibrium of the algorithm, we have proposed, in the second part of chapter 2, the ABCEV and the EEABC, two enhanced versions of the ABC algorithm. In the first version, instead of using the conventional random distribution, a chaotic based distribution mechanism was used to distribute more efficiently the initial population over the search space. Also, the exploitation capability of the algorithm was increased especially around the current global best food source by using a modified update equation to focus the search in the best food source vicinity. The exploratory behaviour of the algorithm was also increased by tuning the *limit* parameter. This parameter affects the generation of the scout bees, which are responsible for exploring the search space and maintaining diversity within the population.

In the proposed ABCEV algorithm, a new update equation was used. This equation allows increasing the exploitation capabilities of the algorithm especially around the best ever food source increasing the convergence speed. But on the other hand, it makes all the artificial bees within the population converge towards the same region of the search space. Consequently, both the diversity within the population and the exploratory behaviour of the algorithm will be decreased. This will make the algorithm more vulnerable to premature convergence and to get trapped in local optima even when the *limit* parameter value is decreased. To address these issues and further enhance the solution quality and the convergence speed, without compromising the population diversity and the exploratory capability, another version (EEABC) of the ABC algorithm was proposed. In this version a new probability equation, that allows the onlookers to select the food sources for further exploitation, was introduced. Using this probability equation, the onlookers do not just exploit the possible prominent regions of the search space (best food sources) but also exploit the less exploited food sources. Hence, the diversity within the population is increased and the problem of premature convergence can be avoided. To make a good balance between the exploration and exploitation capabilities, depending on which of them is currently more beneficial, and make the algorithm more immune with respect to the premature convergence problem, an adaptive control parameter called '*the exploration rate*' was introduced. Furthermore, to increase the convergence speed of the algorithm without compromising its ability to explore and escape local optima, an adaptive update strategy, where one or several optimization parameters could simultaneously be updated, was developed.

To assess the performance of the proposed algorithms, two sets of benchmark functions encompassing 12 commonly used standard numerical benchmark functions and 15 benchmark functions proposed in a CEC2015 special session, were used to implement a comparative study against four variants of the ABC algorithm. The obtained results have showed the good performance of the proposed algorithms.

It is a known fact that even with an efficient optimization algorithm, the NMPC overall performances is still hugely influenced by the quality of the nonlinear model. Using Takagi-Sugeno fuzzy modelling, three predictive control

algorithms were proposed. The first one is based on an improved version of the PSO algorithm while the other algorithms are based on the proposed versions of the ABC algorithm. To evaluate the computational efficiency and the control performance of the developed algorithms, the control of two benchmark systems of different complexity, namely the model of a continuous stirred tank reactor and the model of an industrial boiler, was considered. The obtained performances were compared to those of several other conventional linear and nonlinear MPC strategies. The comparative study and the DSP based experimental setup have showed that the proposed algorithms present good performances. Indeed, the modified PSO based controller has outperformed the controllers that are based on the other PSO algorithms. Also, good performances were obtained in the case of the controllers that are based on the proposed versions of the ABC algorithm. Especially, the EEABC algorithm which has given accurate solutions with less computing complexity compared to all considered algorithms.

In this dissertation, nonlinear model predictive controllers based on meta-heuristic algorithms were proposed as viable and practical substitute to the conventional NMPC controllers. However, solving the NMPC optimization problem using meta-heuristic is relatively a new and a virgin research subject, for which a lot of enhancements and improvements opportunities exist.

Developing computing efficient NMPC based controllers does not only depend on the adopted optimization algorithm, the used nonlinear model is of capital importance. This aspect has not been assessed in this thesis. A more thorough analysis of the impact of the chosen modelling approach on both the control performance and the computing requirement is a critical task that should be developed. Another important factor that influences both the control performances and the computing requirement of NMPC techniques is how to select the optimal values of the design parameters, especially the prediction and control horizons. As such, adopting a strategy to obtain the optimal values for these parameters should also be envisaged.

Although a DSP based setup was considered, a more in depth experimental implementations are still necessary to further evaluate the proposed control strategies in a real control environment.

The evaluation of the proposed meta-heuristic optimization algorithms at the end of chapter 2 has indicated some deficiencies when handling certain numerical benchmark functions. A thorough review to identify the sources of these deficiencies and then correct the underlying problems should also be investigated.

A. LIST OF ABBREVIATIONS

ABC : Artificial Bee Colony
ABCEV : ABC Enhanced Version
ACO : Ant Colony Optimization
AIS : Artificial Immune Systems
BFO : Bacterial Foraging Optimization
BVP : Boundary Value Problem
CARIMA : Controlled Auto-Regressive Integrated Moving-Average
CCS : Code Composer Studio
CEC : Congress on Evolutionary Computation
CSTR : Continuous Stirred Tank Reactor
DAE : Differential Algebraic Equation
DC : Direct Current
DE : Differential Evolution
DMC : Dynamic Matrix Control
DSP : Digital Signal Processor
EEABC : Equal Exploitation ABC
FAR : Fuel /Air Ratio
FIS : Fuzzy Inference System
FNMPC : Fuzzy based Nonlinear Model Predictive Control
GA : Genetic Algorithm
GABC : Global Best ABC
GPC : Generalized Predictive Control
GSA : Gravitational Search Algorithm
GWO : Grey Wolf Optimizer
HJB : Hamilton-Jacobi-Bellman
IDE : Internal Development Environment
IP : Interior Point
KGMO : Kinetic Gas Molecule Optimization
KKT : Karush-Kuhn-Tucker
LMPC : Linear Model Predictive Control

MABC : Modified ABC
MAC : Model Algorithmic Control
MCV : Mean Cost Value
MIMO : Multi-Input Multi -Output
MPC : Model predictive Control
MPHC : Model Predictive Heuristic Control
NIP : Nonlinear Interior Point
NLP : Nonlinear Programing
NMPC : Nonlinear Model Predictive Control
NMPC-NO : Nonlinear Model Predictive Control with Nonlinear Optimization
NMPC-NPL : Nonlinear Model Predictive Control with Nonlinear Predictions and Linearization
NMPC-SL : Nonlinear Model Predictive Control with Successive Linearization
NOP : Nonlinear Optimization Problem
OCP : Optimal Control Problem
ODE : Ordinary Differential Equation
OP : Optimization Problem
PDE : Partial Differential Equation
PFC : Predictive Functional Control
PID : Proportional Integral Differential
PSO : Particle Swarm Optimization
PSOPC : Passive Congregation PSO
PWA : PieceWise Affine
QDMC : Quadratic Dynamic Matrix Control
QP : Quadratic Programming
SA : Simulated Annealing
SD : Standard Deviation
SISO : Single-Input Single-Output
SQP : Sequential Quadratic Programming
TS : Takagi-Sugeno

B. LIST OF SYMBOLS

a_i	Acceleration of particle i (PSO)
$\mathcal{A}(x)$	Active set at the feasible solution x
x_{best}	Best ever food source (ABC)
X_{Best}	Best solution
$\Delta\hat{u}(t)$	Computed increment of the control action
$\hat{u}(t+1\mid t)$	Control action computed at time t to be applied at time $t+1$
N_u	Control horizon
$J(\ )$	Cost function of the optimization problem
d	Dead time of the system
ε	Degree of softening
$d(t)$	Error between measured and predicted process output
$ExplInd(x_h)$	Exploitation Index of food source x_h
α	exploration rate (ABC)
$F_{itness}(.)$	Fitness function
P_g	Global best position (PSO)
$\mathcal{W}_0$	Initial working set
$y(t)$	Measured value of the process output
Max_iter	Maximal number of iteration
N_2	Maximum of the prediction horizon
$u_{\max}$	Maximal value for the control action
$\Delta u_{\max}$	Maximal value for the increment of the control action
$y_{\max}$	Maximal value of the output
N_1	Minimum of the prediction horizon
r_{dmin}	Minimal radius imposed to r_d
$u_{\min}$	Minimal value for the control action

$\Delta u_{\min}$	Minimal value for the increment of the control action
$y_{\min}$	Minimal value of the output
SN	Number of artificial employed bees (ABC)
N_{update}	Number of dimensions to update in each iteration (ABC)
m	Number of system inputs
n	Number of system outputs
PN	Number of total artificial bees (ABC)
LN	Number of artificial onlooker bees (ABC)
$\mathcal{A}(x^*)$	Optimal active set
$\Gamma_y(y)$	Output-dependent weight function
P_h	Personal best position (PSO)
p_h	Probability attached to the food source h (ABC)
E_j	Polynomial of the Diophantine equation
F_j	Polynomial of the Diophantine equation
n_{pop}	Population size
N_p	Prediction horizon
$\hat{y}(t+j\,\|\,t)$	Predicted values of the process output at time $t+j$ computed at time t
$\hat{y}_{\mathrm{mod}\,el}(t+j\,\|\,t)$	Predicted values of the process output at time $t+j$ computed at time t using the model
r_d	Radius around which the PSO particle will be distributed
$w(t)$	Reference trajectory
V	Supply voltage of the DC motor
$e(t)$	Tracking errors
F_{ree}	Vector which contains the free response of the system
v_i	Velocity of particle i (PSO)
Ω	Velocity of the DC motor
λ	Weighing factor penalizing variations in the control signal within predictive control / Lagrange multiplier vector within KKT

	condition
Q	Weight matrix used to penalize the tracking error
R	Weight matrix used to penalize the control input increments
$\mathcal{W}_k$	Working set at iteration k
$e_G(t)$	zero mean white noise

C. CEC 2015 TECHNICAL REPORT (http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2015)

1. Introduction of the CEC'15 expensive optimization test problems

1.1 Common definitions

All test functions are minimization problems defined as follows:

$$\min f(\mathbf{x}), \boldsymbol{x} = [x_1, x_2, ..., x_D]^{\mathrm{T}}$$

where D is the dimension of the problem, $\boldsymbol{o}_{i1} = [o_{i1}, o_{i2}, ..., o_{iD}]^{\mathrm{T}}$ is the shifted global optimum (defined in "shift_data_x.txt"), which is randomly distributed in $[-80,80]^{\mathrm{D}}$. Each function has a shift data for CEC'15. All test functions are shifted to $\boldsymbol{o}$ and scalable. For convenience, the same search ranges are defined for all test functions as $[-100,100]^{\mathrm{D}}$.

In real-world problems, it is seldom that there rarely exist linkages among all variables. The decision variables are divided into subcomponents randomly. The rotation matrix for each sub-component is generated from standard normally distributed entries by Gram-Schmidt ortho-normalization with condition number *c* that is equal to 1 or 2.

1.2 Summary of CEC'15 expensive optimization test problems

Table I. Summary of the CEC' 15 expensive optimization test problems

Categories	No	Functions	Related basic functions	F_i^*
Unimodal functions	1	Rotated Bent Cigar Function	Bent Cigar Function	100
	2	Rotated Discus Function	Discus Function	200
Simple Multimodal functions	3	Shifted and Rotated Weierstrass Function	Weierstrass Function	300
	4	Shifted and Rotated Schwefel's Function	Schwefel's Function	400
	5	Shifted and Rotated Katsuura Function	Katsuura Function	500
	6	Shifted and Rotated HappyCat Function	HappyCat Function	600
	7	Shifted and Rotated HGBat Function	HGBat Function	700
	8	Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function	Griewank's Function Rosenbrock's Function	800
	9	Shifted and Rotated Expanded Scaffer's F6 Function	Expanded Scaffer's F6 Function	900
Hybrid funtions	10	Hybrid Function 1 (*N*=3)	Schwefel's Function Rastrigin's Function High Conditioned Elliptic Function	1000
	11	Hybrid Function 2 (*N*=4)	Griewank's Function Weierstrass Function Rosenbrock's Function Scaffer's F6 Function	1100
	12	Hybrid Function 3 (*N*=5)	Katsuura Function HappyCat Function Griewank's Function Rosenbrock's Function Schwefel's Function Ackley's Function	1200
Composition functions	13	Composition Function 1 (N=5)	Rosenbrock's Function High Conditioned Elliptic Function Bent Cigar Function Discus Function High Conditioned Elliptic Function	1300
	14	Composition Function 2 (*N*=3)	Schwefel's Function Rastrigin's Function High Conditioned Elliptic Function	1400
	15	Composition Function 3 (*N*=5)	HGBat Function Rastrigin's Function Schwefel's Function Weierstrass Function High Conditioned Elliptic Function	1500

Please note: These problems should be treated as black-box optimization problems and without any prior knowledge. Neither the analytical equations nor the problem landscape characteristics extracted from analytical equations are allowed to be used. However, the dimensionality and the number of available function evaluations can be considered as known values and can be used to tune algorithms.

1.2 Definitions of basic functions

1) Bent Cigar Function

$$f_1(\boldsymbol{x}) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2 \tag{1}$$

2) Discus Function

$$f_2(\boldsymbol{x}) = 10^6 x_1^2 + \sum_{i=2}^{D} x_i^2 \tag{2}$$

3) Weierstrass Function

$$f_3(\boldsymbol{x}) = \sum_{i=1}^{D} \left(\sum_{k=0}^{k\max} [a^k \cos(2\pi b^k (x_i + 0.5))] \right) - D \sum_{k=0}^{k\max} [a^k \cos(2\pi b^k \cdot 0.5)] \tag{3}$$

where a=0.5, b=3, and kmax=20.

4) Modified Schwefel's Function

$$f_4(\boldsymbol{x}) = 418.9829 \times D - \sum_{i=1}^{D} g(z_i), \qquad z_i = x_i + 4.209687462275036\text{e+}002$$

$$g(z_i) = \begin{cases} z_i \sin(|z_i|^{1/2}) & \text{if } |z_i| \le 500 \\ (500 - \mathrm{mod}(z_i, 500)) \sin(\sqrt{|500 - \mathrm{mod}(z_i, 500)|}) - \dfrac{(z_i - 500)^2}{10000D} & \text{if } z_i > 500 \\ (\mathrm{mod}(|z_i|, 500) - 500) \sin(\sqrt{|\mathrm{mod}(|z_i|, 500) - 500|}) - \dfrac{(z_i + 500)^2}{10000D} & \text{if } z_i < -500 \end{cases} \tag{4}$$

5) Katsuura Function

$$f_5(\boldsymbol{x}) = \frac{10}{D^2} \prod_{i=1}^{D} \left(1 + i \sum_{j=1}^{32} \frac{|2^j x_i - round(2^j x_i)|}{2^j}\right)^{\frac{10}{D^{1.2}}} - \frac{10}{D^2} \tag{5}$$

6) HappyCat Function

$$f_6(\boldsymbol{x}) = \left| \sum_{i=1}^{D} x_i^2 - D \right|^{1/4} + (0.5 \sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i) / D + 0.5 \tag{6}$$

7) HGBat Function

$$f_7(\boldsymbol{x}) = \left| (\sum_{i=1}^{D} x_i^2)^2 - (\sum_{i=1}^{D} x_i)^2 \right|^{1/2} + (0.5 \sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i) / D + 0.5 \tag{7}$$

8) Expanded Griewank's plus Rosenbrock's Function

$$f_8(\boldsymbol{x}) = f_{11}(f_{10}(x_1, x_2)) + f_{11}(f_{10}(x_2, x_3)) + \ldots + f_{11}(f_{10}(x_{D-1}, x_D)) + f_{11}(f_{10}(x_D, x_1)) \tag{8}$$

9) Expanded Scaffer's F6 Function

Scaffer's F6 Function:

$$g(x, y) = 0.5 + \frac{(\sin^2(\sqrt{x^2 + y^2}) - 0.5)}{(1 + 0.001(x^2 + y^2))^2}$$

$$f_9(\boldsymbol{x}) = g(x_1, x_2) + g(x_2, x_3) + \ldots + g(x_{D-1}, x_D) + g(x_D, x_1) \tag{9}$$

10) Rosenbrock's Function

$$f_{10}(\boldsymbol{x}) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2) \tag{10}$$

11) Griewank's Function

$$f_{11}(\boldsymbol{x}) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{11}$$

12) Rastrigin's Function

$$f_{12}(\boldsymbol{x}) = \sum_{i=1}^{D} (x_i^{\,2} - 10\cos(2\pi x_i) + 10) \tag{12}$$

13) High Conditioned Elliptic Function

$$f_{13}(\boldsymbol{x}) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} \boldsymbol{x}_i^2 \tag{13}$$

14) Ackley's Function

$$f_{14}(\boldsymbol{x}) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e \tag{14}$$

2. Definitions of the CEC'15 Expensive Test Suite

2.1 Unimodal Functions

1) Rotated Bent Cigar Function

$$F_1(\boldsymbol{x}) = f_1(\mathbf{M}(\boldsymbol{x} - \boldsymbol{o}_1)) + F_1^{\,*} \tag{15}$$

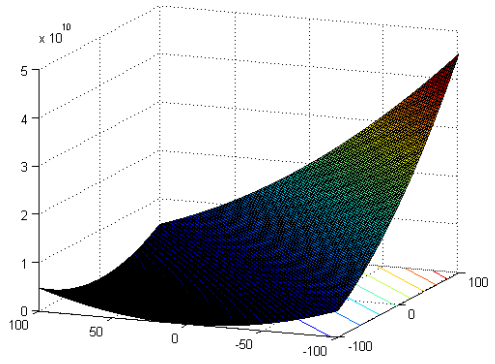

Figure 1. 3-*D* map for 2-*D* function

Properties:

- Unimodal
- Non-separable
- Smooth but narrow ridge

2) Rotated Discus Function

$$F_2(\boldsymbol{x}) = f_2(\mathbf{M}(\boldsymbol{x} - \boldsymbol{o}_2)) + F_2^{\,*} \tag{16}$$

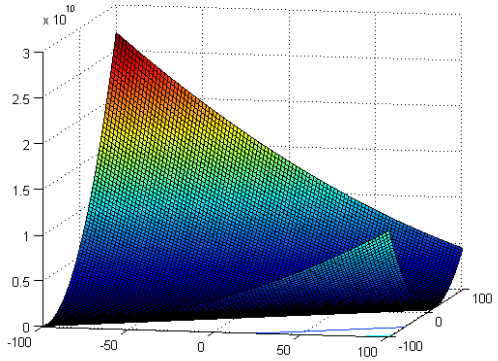

Figure 2. *3-D map for 2-D function*

Properties:

- Unimodal
- Non-separable
- With one sensitive direction

2.2 Simple Multimodal Functions

3) Shifted and Rotated Weierstrass Function

$$F_3(\boldsymbol{x}) = f_3(\mathbf{M}(\frac{0.5(\boldsymbol{x}-\boldsymbol{o}_3)}{100})) + F_3^* \tag{17}$$

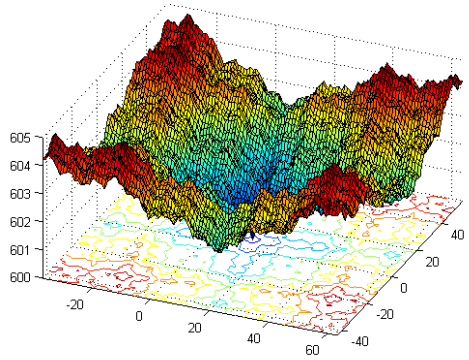

Figure 3. *3-D map for 2-D function*

Properties:

- Multi-modal
- Non-separable
- Continuous but differentiable only on a set of points

4) Shifted and Rotated Schwefel's Function

$$F_4(\boldsymbol{x}) = f_4(\mathbf{M}(\frac{1000(\boldsymbol{x}-\boldsymbol{o}_4)}{100})) + F_4^* \tag{18}$$

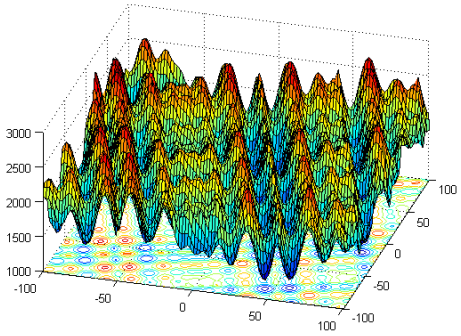

Figure 4(a). *3-D map for 2-D function*

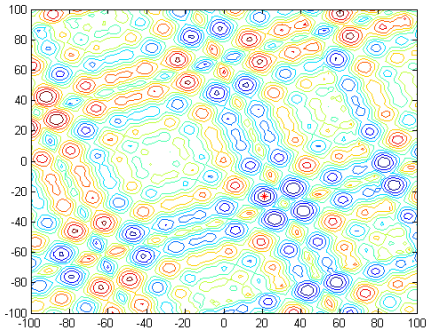

Figure 4(b).*Contour map for 2-D function*

Properties:

- Multi-modal
- Non-separable
- Local optima's number is huge and second better local optimum is far from the global optimum.

5) Shifted and Rotated Katsuura Function

$$F_5(\boldsymbol{x}) = f_5(\mathbf{M}(\frac{5(\boldsymbol{x}-\boldsymbol{o}_5)}{100})) + F_5 * \tag{19}$$

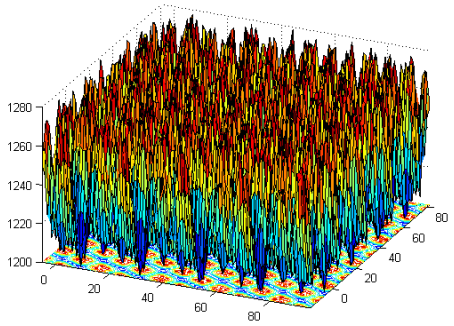

Figure 5(a). *3-D map for 2-D function*

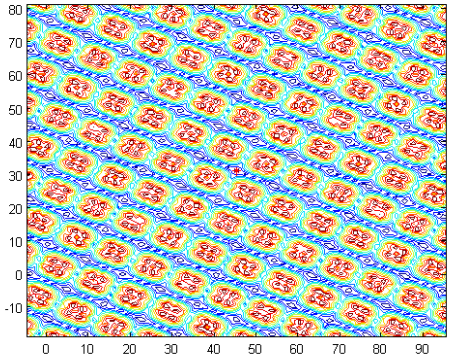

Figure 5(b).*Contour map for 2-D function*

Properties:

- Multi-modal
- Non-separable
- Continuous everywhere yet differentiable nowhere

6) Shifted and Rotated HappyCat Function

$$F_6(\boldsymbol{x}) = f_6(\mathbf{M}(\frac{5(\boldsymbol{x}-\boldsymbol{o}_6)}{100})) + F_6^* \qquad \textbf{(20)}$$

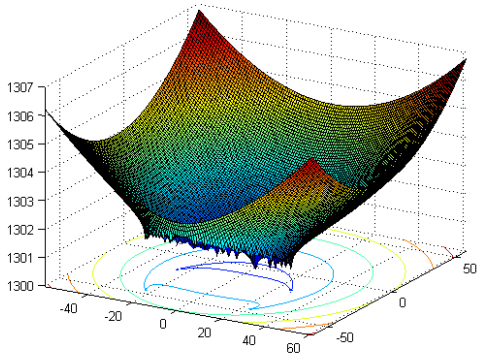

Figure 6(a). *3-D map for 2-D function*

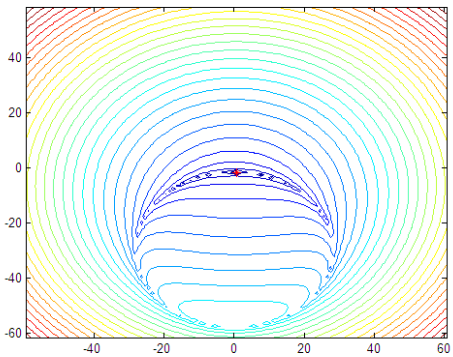

Figure 6(b).*Contour map for 2-D function*

Properties:

- Multi-modal
- Non-separable

7) Shifted and Rotated HGBat Function

$$F_7(\boldsymbol{x}) = f_7(\mathbf{M}(\frac{5(\boldsymbol{x}-\boldsymbol{o}_7)}{100})) + F_7^* \tag{21}$$

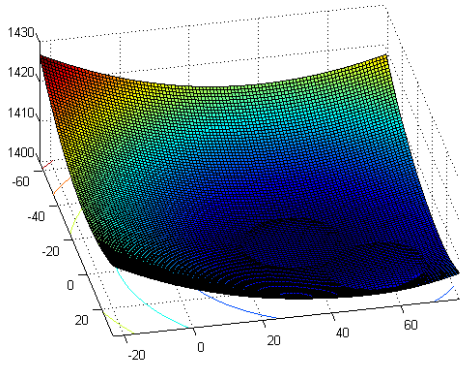

Figure 7(a). *3-D map for 2-D function*

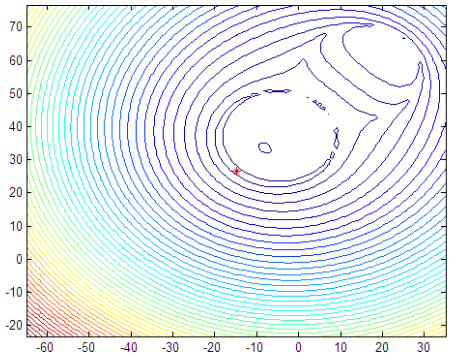

Figure 7(b).*Contour map for 2-D function*

Properties:

- Multi-modal
- Non-separable

8) Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function

$$F_8(\boldsymbol{x}) = f_8(\mathbf{M}(\frac{5(\boldsymbol{x}-\boldsymbol{o}_8)}{100}) + 1) + F_8^* \tag{22}$$

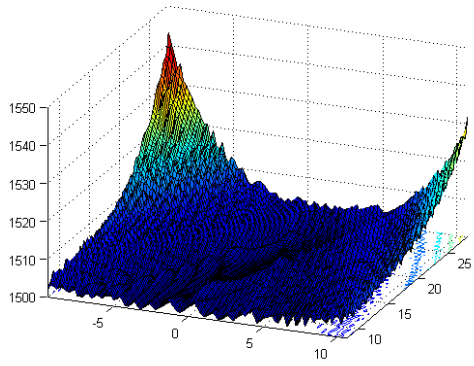

Figure 8(a). *3-D map for 2-D function*

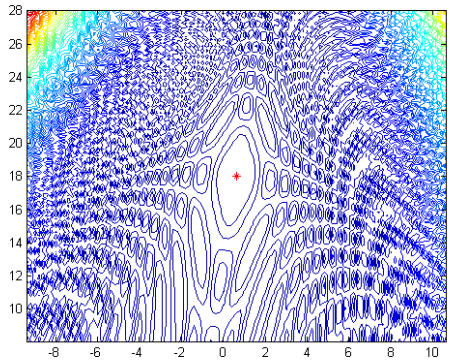

Figure 8(b).*Contour map for 2-D function*

Properties:

- Multi-modal
- Non-separable

9) Shifted and Rotated Expanded Scaffer's F6 Function

$$F_9(\boldsymbol{x}) = f_9(\mathbf{M}(\boldsymbol{x}-\boldsymbol{o}_9)+1)+F_9^* \qquad \textbf{(23)}$$

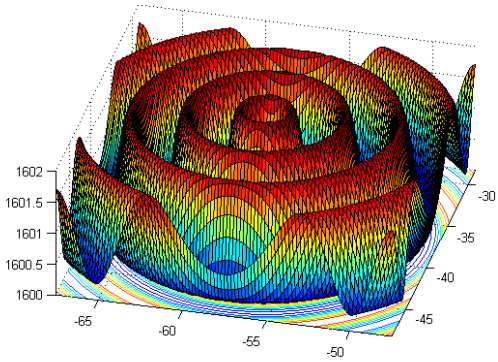

Figure 9. *3-D map for 2-D function*

Properties:

- Multi-modal
- Non-separable

2.3 Hybrid Functions

In real-world optimization problems, different subsets of the variables may have different properties. In this set of hybrid functions, the variables are randomly divided into some subsets and then different basic functions are used for different subsets.

$$F(\boldsymbol{x}) = g_1(\mathbf{M}_1\boldsymbol{z}_1) + g_2(\mathbf{M}_2\boldsymbol{z}_2) + \ldots + g_N(\mathbf{M}_N\boldsymbol{z}_N) + F^*(\boldsymbol{x}) \quad \textbf{(24)}$$

$F(\boldsymbol{x})$: hybrid function
$g_i(\boldsymbol{x})$: i^{th} basic function used to construct the hybrid function
N: number of basic functions

$$\boldsymbol{z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_N]$$

$$\boldsymbol{z}_1 = [\boldsymbol{y}_{S_1}, \boldsymbol{y}_{S_2}, \ldots, \boldsymbol{y}_{S_{n_1}}], \boldsymbol{z}_2 = [\boldsymbol{y}_{S_{n_1+1}}, \boldsymbol{y}_{S_{n_1+2}}, \ldots, \boldsymbol{y}_{S_{n_1+n_2}}], \ldots, \boldsymbol{z}_N = [\boldsymbol{y}_{S_{\sum_{i=1}^{N-1} n_i+1}}, \boldsymbol{y}_{S_{\sum_{k=1}^{N-1} n_i+2}}, \ldots, \boldsymbol{y}_{S_D}] \quad \textbf{(25)}$$

where, $\boldsymbol{y} = \boldsymbol{x} - \boldsymbol{o}_i$ and $S = randperm(1:D)$

p_i: used to control the percentage of $g_i(\boldsymbol{x})$

n_i: dimension for each basic function $\sum_{i=1}^{N} n_i = D$

$$n_1 = \lceil p_1 D \rceil, n_2 = \lceil p_2 D \rceil, \ldots, n_{N-1} = \lceil p_{N-1} D \rceil, n_N = D - \sum_{i=1}^{N-1} n_i \quad \textbf{(26)}$$

10) Hybrid Function 1 (N=3)

p = [0.3, 0.3, 0.4]
g_1: Modified Schwefel's Function f_4
g_2: Rastrigin's Function f_{12}
g_3: High Conditioned Elliptic Function f_{13}

11) Hybrid Function 2 (N=4)

p = [0.2, 0.2, 0.3, 0.3]
g_1: Griewank's Function f_{11}
g_2: Weierstrass Function f_3
g_3: Rosenbrock's Function f_{10}
g_4: Scaffer's F6 Function f_9

12) Hybrid Function 3 (N=5)

p = [0.1, 0.2, 0.2, 0.2, 0.3]
g_1: Katsuura Function f_5
g_2: HappyCat Function f_6
g_3: Expanded Griewank's plus Rosenbrock's Function f_8
g_4: Modified Schwefel's Function f_4

g_5: Ackley's Function f_{14}

2.4 Composite Functions

$$F(\boldsymbol{x}) = \sum_{i=1}^{N} \{\omega_i * [\lambda_i g_i(\boldsymbol{x}) + bias_i]\} + f^* \quad \textbf{(27)}$$

$F(\boldsymbol{x})$: composition function
$g_i(\boldsymbol{x})$: i^{th} basic function used to construct the composition function
N: number of basic functions
o_i: new shifted optimum position for each $g_i(\boldsymbol{x})$, define the global and local optima's position
$bias_i$: defines which optimum is global optimum
σ_i: used to control each $g_i(\boldsymbol{x})$'s coverage range, a small σ_i give a narrow range for that $g_i(x)$
λ_i: used to control each $g_i(\boldsymbol{x})$'s height
w_i: weight value for each $g_i(\boldsymbol{x})$, calculated as below:

$$w_i = \frac{1}{\sqrt{\sum_{j=1}^{D} (x_j - o_{ij})^2}} \exp\left(-\frac{\sum_{j=1}^{D} (x_j - o_{ij})^2}{2D\sigma_i^2}\right) \quad \textbf{(28)}$$

Then normalize the weight $\omega_i = w_i / \sum_{i=1}^{n} w_i$

So when $\boldsymbol{x} = \boldsymbol{o}_i$, $\omega_j = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases}$ for $j = 1, 2, ..., N$, $f(x) = bias_i + f^*$

The optimum which has the smallest bias value is the global optimum. The composition function merges the properties of the sub-functions better and maintains continuity around the global/local optima.

13) Composition Function 1 (N=5)

N= 5， σ = [10, 20, 30, 40, 50]
λ = [1, 1e-6, 1e-26, 1e-6, 1e-6]
$bias$ = [0, 100, 200, 300, 400]
g_1: Rotated Rosenbrock's Function f_{10}
g_2: High Conditioned Elliptic Function f_{13}
g_3: Rotated Bent Cigar Function f_1
g_4: Rotated Discus Function f_2
g_5: High Conditioned Elliptic Function f_{13}

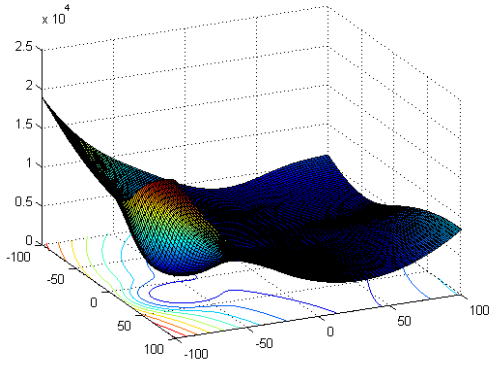

Figure 10(a). *3-D map for 2-D function*

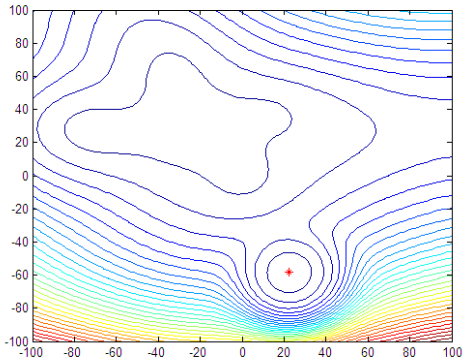

Figure 10 (b).*Contour map for 2-D function*

Properties:

- Multi-modal
- Non-separable
- Asymmetrical
- Different properties around different local optima

14) Composition Function 2 (N=3)

N = 3
σ = [10, 30, 50]
λ = [0.25, 1, 1e-7]
bias = [0, 100, 200]
g_1: Rotated Schwefel's Function f_4
g_2: Rotated Rastrigin's Function f_{12}
g_3: Rotated High Conditioned Elliptic Function f_{13}

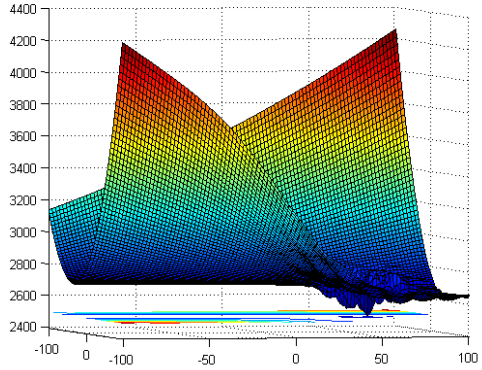

Figure 11(a). *3-D map for 2-D function*

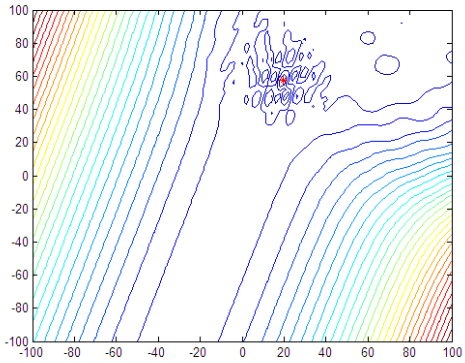

Figure 11(b).*Contour map for 2-D function*

Properties:

- Multi-modal
- Non-separable
- Asymmetrical
- Different properties around different local optima

15) Composition Function 3 (N=5)

N = 5
σ = [10, 10, 10, 20, 20]
λ = [10, 10, 2.5, 25, 1e-6]
bias = [0, 100, 200, 300, 400]
g_1: Rotated HGBat Function f_7
g_2: Rotated Rastrigin's Function f_{12}
g_3: Rotated Schwefel's Function f_4
g_4: Rotated Weierstrass Function f_3
g_5: Rotated High Conditioned Elliptic Function f_{13}

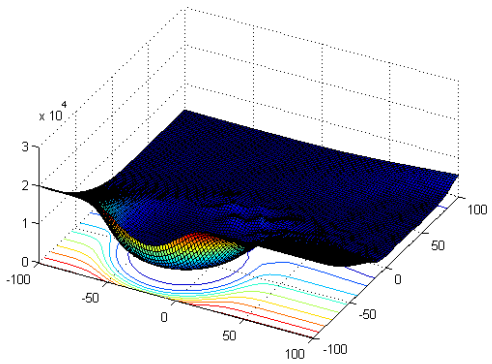

Figure 12(a). *3-D map for 2-D function*

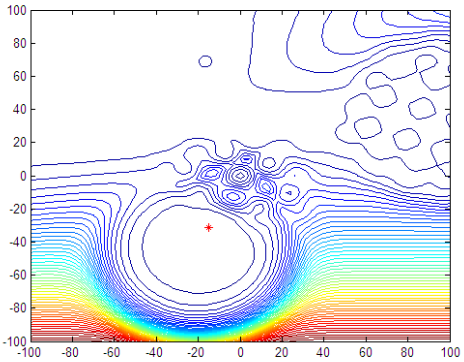

Figure 12(b).*Contour map for 2-D function*

Properties:

- Multi-modal
- Non-separable
- Asymmetrical
- Different properties around different local optima

D. LIST OF PUBLICATIONS

- Ait sahed Oussama, Kamel Kara, and Mohamed Laid Hadjili. "Pso-Based Fuzzy Predictive Control." Paper presented at the IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society, 25-28 Oct. 2012 2012.
- Mohamed Laid Hadjili, Kara Kamel, Ait sahed Oussama, and Bouyanzar Jamal. "Fuzzy Predictive Control Using Particle Swarm Optimization: Application to Scara Robot." *Applied Mechanics and Materials* 527 (2014): 230-36.
- Ait sahed Oussama, Kamel Kara, and Abousoufyane Benyoucef. "Artificial Bee Colony-Based Predictive Control for Non-Linear Systems." *Transactions of the Institute of Measurement and Control* 37 (2015): 780-792.
- Benyoucef Abou soufyane, Aissa Chouder, Kamel Kara, Santiago Silvestre, and Ait sahed Oussama. "Artificial Bee Colony Based Algorithm for Maximum Power Point Tracking (Mppt) for Pv Systems Operating under Partial Shaded Conditions." *Applied Soft Computing* 32, no. 0 (2015): 38-48.
- Ait sahed Oussama, Kamel Kara, and Mohamed Laid Hadjili. "Constrained Fuzzy Predictive Control Using Particle Swarm Optimization." *Applied Computational Intelligence and Soft Computing* (2015): 1-15.
- Ait sahed Oussama, Kamel Kara, Abousoufyane Benyoucef, and Mohamed Laid Hadjili. "A new artificial bee colony algorithm for numerical optimization." *presented at the* CEIT'2015 3rd International Conference on Control, Engineering & Information Technology, 25-27 May, 2015.

REFERENCES

1. Richalet, J., A. Rault, J.L. Testud, and J. Pagon. *Algorithmic control of industrial processes*. in *proceedings of the 4th IFAC Symposium on Identification and System Parameter Estimation*. 1976.
2. Richalet, J., A. Rault, J.L. Testud, and J. Pagon, *Model predictive heuristic control: application to industrial processes.* Automatica, 1978. 14: p. 413-428.
3. Cutler, C.R. and B.C. Ramaker. *Dynamic Matrix Control- A computer control algorithm*. in *proceedings of the automatic control conference*. 1980.
4. Garcia, C.E. and M. Morari, *Internal model control. A unifying review and some new results.* Ind. Eng. Chem. Proc. Des. Dev., 1982. 21: p. 308-323.
5. Morshedi, M., C.R. Cutler, and T.A. Skrovanek. *Optimal solution of dynamic matrix control with linear programming techniques (LDMC)*. in *proceedings of the American Control Conference*. 1985.
6. Garcia, C.E. and A.M. Morshedi, *Quadratic programming solution of dynamic matrix control (QDMC).* Chemical Engineering Communications, 1986. 46: p. 73-87.
7. Clarke, D.W., C. Mohtadi, and P.S. Tuffts, *Generalized predictive control part I, The basic algorithm.* Automatica, 1987. 23: p. 137-148.
8. Clarke, D.W., C. Mohtadi, and P.S. Tuffts, *Generalized predictive control part II, Extensions and interpretations.* Automatica, 1987. 23: p. 149-160.
9. Kouvaritakis, B. and M. Cannon, *Nonlinear Predictive Control, theory and practice*. 2001, United Kingdom: The Institution of Engineering and Technology.
10. Camacho, E.F. and C. Bordons, *Model Predictive Control*. 1999, London: Springer London.
11. Richalet, J. and D. O'Donovan, *Functional Control, Principles and Industrial Applications*. 2009, London: Springer-Verlag London Limited.
12. Maciejowski, J.M., *Predictive Control with Constraints*. 2002, England: Prentice Hall.
13. Sarimveis, H. and G. Bafas, *Fuzzy model predictive control of non-linear processes using genetic algorithms.* Fuzzy Sets and Systems, 2003. 139: p. 59-80.
14. Salahshoor, K., E. Safari, and I. Ahangari, *A novel adaptive fuzzy predictive control for hybrid systems with mixed inputs.* Engineering Applications of Artificial Intelligence, 2013. 26(5–6): p. 1512-1531.
15. Yanhui, X., R. Ghaemi, J. Sun, and J.S. Freudenberg, *Model Predictive Control for a Full Bridge DC/DC Converter.* Control Systems Technology, IEEE Transactions on, 2012. 20(1): p. 164-172.
16. Grancharova, A. and T. Johansen, *Explicit NMPC via Approximate mp-NLP*, in *Explicit Nonlinear Model Predictive Control*. 2012, Springer Berlin Heidelberg. p. 87-110.
17. Marusak, P.M., *Advantages of an easy to design fuzzy predictive algorithm in control systems of nonlinear chemical reactors.* Applied Soft Computing, 2009. 9: p. 1111-1125.

18. Hong-Gui, H., W. Xiao-Long, and Q. Jun-Fei, *Real-Time Model Predictive Control Using a Self-Organizing Neural Network.* Neural Networks and Learning Systems, IEEE Transactions on, 2013. 24(9): p. 1425-1436.

19. Mayne, D.Q., J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert, *Constrained model predictive control: Stability and optimality.* Automatica, 2000. 36(6): p. 789-814.

20. Rossiter, J.A., *Model-Based Predictive Control. A Practical Approach*. 2004, Florida: CRC Press.

21. Roubos, J., R. Babuska, P. Bruijn, and H. Verbruggen. *Predictive control by local linearization of a Takagi-Sugeno fuzzy model*. in *proceedings of the IEEE International Conference on Fuzzy Systems*. 1998.

22. Kosko, B. *Fuzzy systems as universal approximators*. in *proceedings of First IEEE International Conference on Fuzzy System*. 1992.

23. Baocang, D. and P. Xubin, *Output Feedback Predictive Control With One Free Control Move for Nonlinear Systems Represented by a Takagi-Sugeno Model.* Fuzzy Systems, IEEE Transactions on, 2014. 22(2): p. 249-263.

24. Takagi, T. and M. Sugeno. *Identification of systems and its applications to modeling and control*. in *proceedings of the International Conference on Systems, Man and Cybernetics*. 1985.

25. Jiang, H., C.K. Kwong, Z. Cen, and Y.C. Ysim, *Chaos particle swarm optimization and T-S fuzzy modeling approaches to constrained predictive control.* Expert Systems with Applications, 2012. 39: p. 194-201.

26. Tatjewski, P., *Advanced Control of Industrial Process Structure and Algorithms*. Advances in industrial control. 2007, London: Springer-Verlag London Limited.

27. Grüne, L. and J. Pannek, *Nonlinear Model Predictive Control: Theory and Algorithms*. 2011, London: Springer London.

28. Xiangjie, L., G. Ping, and C.W. Chan, *Nonlinear Multivariable Power Plant Coordinate Control by Constrained Predictive Scheme.* Control Systems Technology, IEEE Transactions on, 2010. 18(5): p. 1116-1125.

29. Coen, T., J. Anthonis, and J. De Baerdemaeker, *Cruise control using model predictive control with constraints.* Computers and Electronics in Agriculture, 2008. 63(2): p. 227-236.

30. Borrelli, F., M. Baotić, J. Pekar, and G. Stewart, *On the computation of linear model predictive control laws.* Automatica, 2010. 46(6): p. 1035-1041.

31. Rao, C.V., S.J. Wright, and J.B. Rawlings, *Application of Interior-Point Methods to Model Predictive Control.* Journal of Optimization Theory and Applications, 1998. 99(3): p. 723-757.

32. Harinath, E., L.T. Biegler, and G.A. Dumont, *Control and optimization strategies for thermo-mechanical pulping processes: Nonlinear model predictive control.* Journal of Process Control, 2011. 21(4): p. 519-528.

33. Rocha, M., R. Mendes, O. Rocha, I. Rocha, and E.C. Ferreira, *Optimization of fed-batch fermentation processes with bio-inspired algorithms.* Expert Systems with Applications, 2014. 41(5): p. 2186-2195.

34. Li, Y., J. Shen, K.Y. Lee, and X. Liu, *Offset-free fuzzy model predictive control of a boiler–turbine system based on genetic algorithm.* Simulation Modelling Practice and Theory, 2012. 26: p. 77-95.

35. Modares, H. and M.-B. Naghibi Sistani, *Solving nonlinear optimal control problems using a hybrid IPSO–SQP algorithm.* Engineering Applications of Artificial Intelligence, 2011. 24(3): p. 476-484.

36. Johansen, T.A., *Introduction to nonlinear model predictive control and moving horizon estimation.* Selected Topics on Constrained and Nonlinear Control, 2011: p. 187.

37. Betts, J.T., *Survey of Numerical Methods for Trajectory Optimization.* Journal of Guidance, Control, and Dynamics, 1998. 21(2): p. 193-207.

38. Parsopoulos, K.E. and M.N. Vrahatis, *Particle Swarm Optimization and Intelligence: Advances and Applications*. 2010, United States of America: Information Science Reference (an imprint of IGI Global).

39. Talbi, E.-G., *Metaheuristics: From Design to Implementation*. 2009: Wiley Publishing. 593.

40. Clerc, M., *Particle Swarm Optimization*. 2006, United Kingdom: ISTE Ltd.

41. Xin-She, Y. and S. Deb. *Cuckoo Search via Lévy flights*. in *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*. 2009.

42. Mirjalili, S., S.M. Mirjalili, and A. Lewis, *Grey Wolf Optimizer.* Advances in Engineering Software, 2014. 69(0): p. 46-61.

43. dos Santos Coelho, L. and P. Alotto, *Gaussian Artificial Bee Colony Algorithm Approach Applied to Loney's Solenoid Benchmark Problem.* Magnetics, IEEE Transactions on, 2011. 47(5): p. 1326-1329.

44. Kronfeld, M. and A. Zell, *Gaussian Process Assisted Particle Swarm Optimization*, in *Learning and Intelligent Optimization*, C. Blum and R. Battiti, Editors. 2010, Springer Berlin Heidelberg. p. 139-153.

45. Nasir, M., S. Das, D. Maity, S. Sengupta, U. Halder, and P.N. Suganthan, *A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization.* Information Sciences, 2012. 209(0): p. 16-36.

46. Germin Nisha, M. and G.N. Pillai, *Nonlinear model predictive control with relevance vector regression and particle swarm optimization.* Journal of Control Theory and Applications, 2013. 11(4): p. 563-569.

47. Bououden, S., M. Chadli, and H.R. Karimi, *An ant colony optimization-based fuzzy predictive control approach for nonlinear processes.* Information Sciences, 2015. 299(0): p. 143-158.

48. Chakrabarty, A., S. Banerjee, S. Maity, and A. Chatterjee, *Fuzzy model predictive control of non-linear processes using convolution models and foraging algorithms.* Measurement, 2013. 46(4): p. 1616-1629.

49. Karaboga, D. and B. Basturk, *On the performance of artificial bee colony (ABC) algorithm.* applied Soft Computing, 2008. 8: p. 687-697.

50. Karaboga, D. and B. Akay, *A comparative study of artificial bee colony algorithm.* applied Mathematics and Computation, 2009. 214: p. 108-132.

51. Karaboga, D., B. Gorkemli, C. Ozturk, and N. Karaboga, *A comprehensive survey: artificial bee colony (ABC) algorithm and applications.* Artificial Intelligence Review, 2014. 42(1): p. 21-57.

52. Henson, M.A., *Nonlinear model predictive control: current status and future directions.* Computers & Chemical Engineering, 1998. 23(2): p. 187-202.

53. Duchêne, P. and P. Rouchon, *Kinetic scheme reduction via geometric singular perturbation techniques.* Chemical Engineering Science, 1996. 51(20): p. 4661-4672.

54. Lévine, J. and P. Rouchon, *Quality control of binary distillation columns via nonlinear aggregated models.* Automatica, 1991. 27(3): p. 463-480.

55. Kara, K., M.L. Hadjili, K.e. Hemsas, and T. Missoum. *Predictive Control Using Neural Networks*. 2009. Portugal.

56. Ferreira, P.M., A.E. Ruano, S. Silva, and E.Z.E. Conceição, *Neural networks based predictive control for thermal comfort and energy savings in public buildings.* Energy and Buildings, 2012. 55(0): p. 238-251.

57. Ait Sahed, O., K. Kara, and A. Benyoucef, *Artificial bee colony-based predictive control for non-linear systems.* Transactions of the Institute of Measurement and Control, 2015. 37(6): p. 780-792.

58. Ait Sahed, O., K. Kara, and M.L. Hadjili, *Constrained Fuzzy Predictive Control Using Particle Swarm Optimization.* Applied Computational Intelligence and Soft Computing, 2015. 2015: p. 1-15.

59. Fruzzetti, K.P., A. Palazoğlu, and K.A. McDonald, *Nolinear model predictive control using Hammerstein models.* Journal of Process Control, 1997. 7(1): p. 31-41.

60. Jurado, F., *Predictive control of solid oxide fuel cells using fuzzy Hammerstein models.* Journal of Power Sources, 2006. 158(1): p. 245-253.

61. Mahmoodi, S., J. Poshtan, M.R. Jahed-Motlagh, and A. Montazeri, *Nonlinear model predictive control of a pH neutralization process based on Wiener–Laguerre model.* Chemical Engineering Journal, 2009. 146(3): p. 328-337.

62. Shafiee, G., M.M. Arefi, M.R. Jahed-Motlagh, and A.A. Jalali, *Nonlinear predictive control of a polymerization reactor based on piecewise linear Wiener model.* Chemical Engineering Journal, 2008. 143(1–3): p. 282-292.

63. Qin, S.J. and T.A. Badgwell, *A survey of industrial model predictive control technology.* Control Engineering Practice, 2003. 11(7): p. 733-764.

64. Camacho, E. and C. Bordons, *Nonlinear Model Predictive Control: An Introductory Review*, in *Assessment and Future Directions of Nonlinear Model Predictive Control*, R. Findeisen, F. Allgöwer, and L. Biegler, Editors. 2007, Springer Berlin Heidelberg. p. 1-16.

65. Richalet, J., *Pratique de la commande predictive*. 1992: Hermes.

66. Baocang, D., *Modern predictive control*. 2010, United States of America: CRC Press.

67. Nocedal, J. and S.J. Wright, *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. 1999, New York: Springer New York.

68. Bartholomew–Biggs, M., *Nonlinear Optimization with Engineering Applications*. Vol. 19. 2008, New York, USA: Springer US.

69. Wright, S.J., *Primal-Dual Interior-Point Methods*. 1997, Philadelphia. USA: SIAM Publications.

70. Fletcher, R., *Practical Methods of Optimization: 2nd edition*. 1987: Wiley.

71. Wright, S.J., *Applying New Optimization Algorithms To Model Predictive Control*, in *Fifth International Conference on Chemical Process Control – CPC V*, J.C. Kantor, C.E. García, and B. Carnahan, Editors. 1996, CACHE Corporation, AIChE and CACHE Corporation. p. 147-155.

72. Cannon, M., B. Kouvaritakis, and J.A. Rossiter, *Efficient active set optimization in triple mode MPC.* Automatic Control, IEEE Transactions on, 2001. 46(8): p. 1307-1312.

73. Ferreau, H.J., P. Ortner, P. Langthaler, L.d. Re, and M. Diehl, *Predictive control of a real-world Diesel engine using an extended online active set strategy.* Annual Reviews in Control, 2007. 31(2): p. 293-301.

74. Naik, V.V., D.N. Sonawane, D.D. Ingole, and D. Ginoya. *Model Predictive Control of DC Servomotor using Active Set Method*. in *Control Applications (CCA), 2013 IEEE International Conference on*. 2013.

75. Karmarkar, N., *A new polynomial-time algorithm for linear programming.* Combinatorica, 1984. 4(4): p. 373-395.

76. Patrinos, P., P. Sopasakis, and H. Sarimveis, *A global piecewise smooth Newton method for fast large-scale model predictive control.* Automatica, 2011. 47(9): p. 2016-2022.

77. Ye, Y., *Interior point algorithms: theory and analysis*. 1997: John Wiley & Sons, Inc. 418.

78. Gondzio, J. and A. Grothey, *A New Unblocking Technique to Warmstart Interior Point Methods Based on Sensitivity Analysis.* SIAM Journal on Optimization, 2008. 19(3): p. 1184-1210.

79. Cai, X., M.J. Tippett, L. Xie, and J. Bao, *Fast distributed MPC based on active set method.* Computers & Chemical Engineering, 2014. 71(0): p. 158-170.

80. Shahzad, A., E.C. Kerrigan, and G.A. Constantinides. *A warm-start interior-point method for predictive control*. in *Control 2010, UKACC International Conference on*. 2010.

81. Zang, H., H. Li, J. Huang, and J. Wang, *A Composite Model Predictive Control Strategy for Furnaces.* Chinese Journal of Chemical Engineering, 2014. 22(7): p. 788-794.

82. Jerez, J.L., E.C. Kerrigan, and G.A. Constantinides, *A sparse and condensed QP formulation for predictive control of LTI systems.* Automatica, 2012. 48(5): p. 999-1002.

83. Amrit, R., J.B. Rawlings, and L.T. Biegler, *Optimizing process economics online using model predictive control.* Computers & Chemical Engineering, 2013. 58(0): p. 334-343.

84. Lu, C.-H. and C.-C. Tsai, *Adaptive Predictive Control With Recurrent Neural Network for Industrial Processes: An Application to Temperature Control of a Variable-Frequency Oil-Cooling Machine.* IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, 2008. 55(3): p. 1366-1375.

85. Allgower, F., F. Rolf, and K.N. Zoltan, *Nonlinear Model Predictive Control: From Theory to Application.* Journal of the Chinese Institute of Chemical Engineers, 2004. 35: p. 299-315.

86. Roubos, J.A., R. Babuska, P.M. Bruijn, and H. Verbruggen. *Predictive control by local linearization of a Takagi-Sugeno fuzzy model*. in *Fuzzy Systems Proceedings, 1998. IEEE*

World Congress on Computational Intelligence., The 1998 IEEE International Conference on. 1998.

87. Abonyi, J., L. Nagy, and F. Szeifert, *Fuzzy model-based predictive control by instantaneous linearization.* Fuzzy Sets and Systems, 2001. 120(1): p. 109-122.

88. Townsend, S. and G.W. Irwin, *Nonlinear model based predictive control using multiple local models*. Non-linear Predictive Control: theory and practice. 2001: Institution of Engineering and Technology. 223-244.

89. Cannon, M., D. Ng, and B. Kouvaritakis, *Successive Linearization NMPC for a Class of Stochastic Nonlinear Systems*, in *Nonlinear Model Predictive Control*, L. Magni, D. Raimondo, and F. Allgöwer, Editors. 2009, Springer Berlin Heidelberg. p. 249-262.

90. Garcia, C.E. *Quadratic/dynamic matrix control of nonlinear processes: an application to a batch reaction process*. in *AIChE annual meeting*. 1984. San Francesco.

91. Marusak, P. and P. Tatjewski. *Fuzzy Dynamic Matrix Control algorithm for nonlinear plants*. in *6th Int. Conf. methods and models in Automation and obotics MMAR'00*. 2000. Miedzyzdroje, Poland.

92. Camacho, E.F., M. Berenguel, and F.R. Rubio, *Advanced Control of Solar Plants*. Advances in Industrial Control. 1997: Springer London.

93. Diehl, M., H. Ferreau, and N. Haverbeke, *Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation*, in *Nonlinear Model Predictive Control*, L. Magni, D. Raimondo, and F. Allgöwer, Editors. 2009, Springer Berlin Heidelberg. p. 391-417.

94. Binder, T., L. Blank, H.G. Bock, R. Bulirsch, W. Dahmen, M. Diehl, T. Kronseder, W. Marquardt, J. Schlöder, and O. von Stryk, *Introduction to Model Based Optimization of Chemical Processes on Moving Horizons*, in *Online Optimization of Large Scale Systems*, M. Grötschel, S. Krumke, and J. Rambau, Editors. 2001, Springer Berlin Heidelberg. p. 295-339.

95. Saerens, B., M. Diehl, and E. Van den Bulck, *Optimal Control Using Pontryagin's Maximum Principle and Dynamic Programming*, in *Automotive Model Predictive Control*, L. del Re, F. Allgöwer, L. Glielmo, C. Guardiola, and I. Kolmanovsky, Editors. 2010, Springer London. p. 119-138.

96. Mayne, D.Q., *Optimization in Model Predictive Control*, in *Methods of Model Based Process Control*, R. Berber, Editor. 1995, Springer Netherlands. p. 367-396.

97. Findeisen, R. and F. Allgöwer, *An introduction to nonlinear model predictive control*, in *21st Benelux Meeting on Systems and Control*. 2002.

98. Herceg, M., M. Kvasnica, and M. Fikar, *Parametric Approach to Nonlinear Model Predictive Control*, in *Nonlinear Model Predictive Control Lecture Notes in Control and Information Sciences*, L. Magni, D.M. Raimondo, and F. Allgöwer, Editors. 2009, Springer Berlin Heidelberg: Berlin. p. 381-389.

99. Nevistic, V. and J.A. Primbs. *Model predictive control: breaking through constraints*. in *Decision and Control, 1996., Proceedings of the 35th IEEE Conference on*. 1996.

100. Diehl, M., *Optimization Algorithms for Model Predictive Control*, in *Encyclopedia of Systems and Control*, J. Baillieul and T. Samad, Editors. 2014, Springer London. p. 1-11.

101. Betts, J.T., *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. 2010: SIAM.

102. Lebedev, L.P. and M.J. Cloud, *The Calculus of Variations and Functional Analysis: With Optimal Control and Applications in Mechanics*. 2003, Singapore: World Scientific Publishing Co. Pte. Ltd.

103. Powell, M.J.D., *A fast algorithm for nonlinearly constrained optimization calculations*, in *Numerical Analysis*, G.A. Watson, Editor. 1978, Springer Berlin Heidelberg. p. 144-157.

104. Byrd, R. and J. Nocedal, *An analysis of reduced Hessian methods for constrained optimization.* Mathematical Programming, 1990. 49(1-3): p. 285-323.

105. Boggs, P.T. and J.W. Tolle, *Sequential Quadratic Programming.* Acta Numerica, 1995. 4: p. 1-51.

106. Schäfer, A., P. Kühl, M. Diehl, J. Schlöder, and H.G. Bock, *Fast reduced multiple shooting methods for nonlinear model predictive control.* Chemical Engineering and Processing: Process Intensification, 2007. 46(11): p. 1200-1214.

107. Kirches, C., L. Wirsching, H.G. Bock, and J.P. Schlöder, *Efficient direct multiple shooting for nonlinear model predictive control on long horizons.* Journal of Process Control, 2012. 22(3): p. 540-550.

108. Wang, S.W., D.L. Yu, J.B. Gomm, G.F. Page, and S.S. Douglas, *Adaptive neural network model based predictive control for air–fuel ratio of SI engines.* Engineering Applications of Artificial Intelligence, 2006. 19(2): p. 189-200.

109. Sargent, R.W.H. and G.R. Sullivan, *The development of an efficient optimal control package*, in *Optimization Techniques*, J. Stoer, Editor. 1978, Springer Berlin Heidelberg. p. 158-168.

110. Biegler, L.T., *Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation.* Computers & Chemical Engineering, 1984. 8(3–4): p. 243-247.

111. Bayón, L., J.M. Grau, M.M. Ruiz, and P.M. Suárez, *Initial guess of the solution of dynamic optimization of chemical processes.* Journal of Mathematical Chemistry, 2010. 48(1): p. 28-37.

112. Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*. 1989: Addison-Wesley Longman Publishing Co., Inc. 372.

113. Haupt, R.L. and S.E. Haupt, *Practical Genetic Algorithms*. 2004: John Wiley & Sons, Inc.

114. Storn, R. and K. Price, *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces*. 1995, Technical Report TR-95-012, International Computer Science Institute, Berkeley, USA.

115. Feoktistov, V., *Differential Evolution In Search of Solutions*. Differential Evolution. Vol. 5. 2006: Springer US.

116. Kennedy, J. and R. Eberhart. *Particle swarm optimization*. in *Neural Networks, 1995. Proceedings., IEEE International Conference on*. 1995.

117. Dorigo, M., V. Maniezzo, and A. Colorni, *Ant system: optimization by a colony of cooperating agents.* IEEE Trans Syst Man Cybern B Cybern, 1996. 26(1): p. 29-41.

118. Bonabeau, E., M. Dorigo, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems*. 1999: Oxford university press.

119. Karaboga, D., *An idea based on honey bee swarm for numerical optimization*. 2005, Erciyes University: Kayseri/Turkey.

120. Karaboga, D. and B. Basturk, *A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm.* journal of global optimization, 2007. 39(3): p. 459-471.

121. Rashedi, E., H. Nezamabadi-pour, and S. Saryazdi, *GSA: A Gravitational Search Algorithm.* Information Sciences, 2009. 179(13): p. 2232-2248.

122. Rashedi, E., H. Nezamabadi-pour, and S. Saryazdi, *BGSA: binary gravitational search algorithm.* Natural Computing, 2010. 9(3): p. 727-745.

123. Moein, S. and R. Logeswaran, *KGMO: A swarm optimization algorithm based on the kinetic energy of gas molecules.* Information Sciences, 2014. 275(0): p. 127-144.

124. Dréo, J., P. Siarry, A. Pétrowski, and E. Taillard, *Metaheuristics for Hard Optimization*. 2006: Springer Berlin Heidelberg.

125. Maniezzo, V., T. Stützle, and S. Voß, *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*. Annals of Information Systems. Vol. 10. 2010: Springer US.

126. Bemporad, A., M. Morari, V. Dua, and E.N. Pistikopoulos. *The explicit solution of model predictive control via multiparametric quadratic programming*. in *American Control Conference, 2000. Proceedings of the 2000*. 2000.

127. Alessio, A. and A. Bemporad, *A Survey on Explicit Model Predictive Control*, in *Nonlinear Model Predictive Control*, L. Magni, D. Raimondo, and F. Allgöwer, Editors. 2009, Springer Berlin Heidelberg. p. 345-369.

128. Johansen, T.A. *On multi-parametric nonlinear programming and explicit nonlinear model predictive control*. in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*. 2002.

129. Johansen, T.A., *Approximate explicit receding horizon control of constrained nonlinear systems.* Automatica, 2004. 40(2): p. 293-300.

130. Grancharova, A., T. Johansen, and P. Tøndel, *Computational Aspects of Approximate Explicit Nonlinear Model Predictive Control*, in *Assessment and Future Directions of Nonlinear Model Predictive Control*, R. Findeisen, F. Allgöwer, and L. Biegler, Editors. 2007, Springer Berlin Heidelberg. p. 181-192.

131. Zheng, A. and W.-h. Zhang, *Computationally efficient non linear predictive control algorithm for control of constrained nonlinear systems*. Non-linear Predictive Control: theory and practice, ed. B. Kouvaritakis and M. Cannon. 2001: Institution of Engineering and Technology. 173-188.

132. Kerrigan, E.C., *Robust Constraint Satisfaction Invariant Sets and Predictive Control*, in *Department of Engineering*. 2000, University of Cambridge.

133. Bithmead, R.P., V. Wertz, and M. Gerers, *Adaptive Optimal Control: The Thinking Man's G.P.C*. 1991: Prentice Hall Professional Technical Reference. 244.

134. Allgöwer, F., T.A. Badgwell, J.S. Qin, J.B. Rawlings, and S.J. Wright, *Nonlinear Predictive Control and Moving Horizon Estimation — An Introductory Overview*, in *Advances in Control*, P. Frank, Editor. 1999, Springer London. p. 391-449.

135. Chen, H. and F. AllgöWer, *A Quasi-Infinite Horizon Nonlinear Model Predictive Control Scheme with Guaranteed Stability.* Automatica, 1998. 34(10): p. 1205-1217.

136. de Oliveira, S. and M. Morari, *Contractive Model Predictive Control with Local Linearization for Nonlinear Systems*, in *Nonlinear Model Based Process Control*, R. Berber and C. Kravaris, Editors. 1998, Springer Netherlands. p. 403-431.

137. Kalman, R.E., *Contributions to the theory of optimal contro.* Boletin Sociedad Matematica Mexicana, 1960. 5: p. 102-119.

138. Scokaert, P.O.M., D.Q. Mayne, and J.B. Rawlings, *Suboptimal model predictive control (feasibility implies stability).* Automatic Control, IEEE Transactions on, 1999. 44(3): p. 648-654.

139. Michalska, H. and D.Q. Mayne, *Robust receding horizon control of constrained nonlinear systems.* Automatic Control, IEEE Transactions on, 1993. 38(11): p. 1623-1633.

140. Yang, T.H. and E. Polak, *Moving horizon control of nonlinear systems with input saturation, disturbances and plant uncertainty.* International Journal of Control, 1993. 58(4): p. 875-903.

141. Polya, G., *How to Solve It: A New Aspect of Mathematical Method*. 1945, Princeton, New Jersey: Princeton University Press.

142. Caserta, M. and S. Voß, *Metaheuristics: Intelligent Problem Solving*, in *Matheuristics*, V. Maniezzo, T. Stützle, and S. Voß, Editors. 2010, Springer US. p. 1-38.

143. Glover, F., *Future paths for integer programming and links to artificial intelligence.* Computers & Operations Research, 1986. 13(5): p. 533-549.

144. Osman, I.H. and J.P. Kelly, *Meta-Heuristics: An Overview*, in *Meta-Heuristics*, I.H. Osman and J.P. Kelly, Editors. 1996, Springer US. p. 1-21.

145. Maaranen, H., K. Miettinen, and A. Penttinen, *On initial populations of a genetic algorithm for continuous optimization problems.* journal of global optimization, 2007. 37(3): p. 405-436.

146. Grosan, C., A. A, and M. Nicoara, *Search Optimization Using Hybrid Particle Sub-Swarms and Evolutionary Algorithms.* International Journal of Simulation Systems, Science & Technology, 2005. 6(10&11): p. 60-79.

147. Storn, R. and K. Price, *Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces.* journal of global optimization, 1997. 11(4): p. 341-359.

148. Iba, H. and C. Aranha, *Practical Applications of Evolutionary Computation to Financial Engineering*. Vol. 11. 2012: Springer Berlin Heidelberg.

149. Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*. 1996: Springer Berlin Heidelberg.

150. Hui, W., S. Rahnamayan, and W. Zhijian. *Adaptive Differential Evolution with variable population size for solving high-dimensional problems*. in *Evolutionary Computation (CEC), 2011 IEEE Congress on*. 2011.

151. Chen, S.-M., A. Sarosh, and Y.-F. Dong, *Simulated annealing based artificial bee colony algorithm for global numerical optimization.* Applied Mathematics and Computation, 2012. 219(8): p. 3575-3589.

152. Xiang, W.-l. and M.-q. An, *An efficient and robust artificial bee colony algorithm for numerical optimization.* Computers & Operations Research, 2013. 40(5): p. 1256-1265.

153. Zhu, G. and S. Kwong, *Gbest-guided artificial bee colony algorithm for numerical function optimization.* Applied Mathematics and Computation, 2010. 217(7): p. 3166-3173.

154. Wolpert, D.H. and W.G. Macready, *No free lunch theorems for search*. July 1995, Santa Fe Institute: Santa Fe, NM.

155. Wolpert, D.H. and W.G. Macready, *No free lunch theorems for optimization.* Evolutionary Computation, IEEE Transactions on, 1997. 1(1): p. 67-82.

156. Holland, J.H., *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. 1975: U Michigan Press.

157. Yang, X.-S., *Engineering Optimization. An Introduction with Metaheuristic Applications*. 2010, New Jersey. USA: John Wiley & Sons, Inc.

158. Reeves, C.R., *Genetic Algorithms*, in *Handbook of Metaheuristics*, M. Gendreau and J.-Y. Potvin, Editors. 2010, Springer US. p. 109-139.

159. Thakur, M., *A new genetic algorithm for global optimization of multimodal continuous functions.* Journal of Computational Science, 2014. 5(2): p. 298-311.

160. Valle, Y.d., G.K. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, and R.G. Harley, *Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems.* IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, 2008. 12(2): p. 171-195.

161. Radcliffe, N.J., *Equivalence class analysis of genetic algorithms.* Complex systems, 1991. 5(2): p. 183-205.

162. Eshelman, L.J. and J.D. Schaffer, *Real-Coded Genetic Algorithms and Interval-Schemata*, in *Foundations of Genetic Algorithms*, W. L. Darrell, Editor. 1993, Elsevier. p. 187-202.

163. Kaelo, P. and M.M. Ali, *Integrated crossover rules in real coded genetic algorithms.* European Journal of Operational Research, 2007. 176(1): p. 60-76.

164. Deep, K. and M. Thakur, *A new crossover operator for real coded genetic algorithms.* Applied Mathematics and Computation, 2007. 188(1): p. 895-911.

165. Blum, C. and D. Merkle, *Swarm Intelligence, Introduction and Applications*. 2008, Berlin: Springer.

166. Clerc, M. and J. Kennedy, *The particle swarm - explosion, stability, and convergence in a multidimensional complex space.* Evolutionary Computation, IEEE Transactions on, 2002. 6(1): p. 58-73.

167. Xinchao, Z., *A perturbed particle swarm algorithm for numerical optimization.* Applied Soft Computing, 2010. 10(1): p. 119-124.

168. Ostadmohammadi Arani, B., P. Mirzabeygi, and M. Shariat Panahi, *An improved PSO algorithm with a territorial diversity-preserving scheme and enhanced exploration–exploitation balance.* Swarm and Evolutionary Computation, 2013. 11(0): p. 1-15.

169. Silva, A., A. Neves, and E. Costa, *SAPPO: A Simple, Adaptable, Predator Prey Optimiser*, in *Progress in Artificial Intelligence*, F. Pires and S. Abreu, Editors. 2003, Springer Berlin Heidelberg. p. 59-73.

170. He, S., Q.H. Wu, J.Y. Wen, J.R. Saunders, and R.C. Paton, *A particle swarm optimizer with passive congregation.* Biosystems, 2004. 78(1–3): p. 135-147.

171. Sun, J., W. Fang, V. Palade, X. Wu, and W. Xu, *Quantum-behaved particle swarm optimization with Gaussian distributed local attractor point.* Applied Mathematics and Computation, 2011. 218(7): p. 3763-3775.

172. Ji, W., J. Wang, and J. Zhang, *Improved PSO based on update strategy of double extreme value.* International Journal of Control and Automation, 2014. 7(2): p. 231-240.

173. Kaveh, A. and A. Zolghadr, *Democratic PSO for truss layout and size optimization with frequency constraints.* Computers & Structures, 2014. 130(0): p. 10-21.

174. He, G. and N.-j. Huang, *A new particle swarm optimization algorithm with an application.* Applied Mathematics and Computation, 2014. 232(0): p. 521-528.

175. Wang, H., H. Sun, C. Li, S. Rahnamayan, and J.-s. Pan, *Diversity enhanced particle swarm optimization with neighborhood search.* Information Sciences, 2013. 223(0): p. 119-135.

176. Gao, W. and S. Liu, *A modified artificial bee colony algorithm.* Computers & Operations Research, 2012. 39: p. 687-697.

177. Gao, W., S. Liu, and L. Huang, *A global best artificial bee colony algorithm for global optimization.* Journal of Computational and Applied Mathematics, 2012. 326: p. 2741-2753.

178. Bäck, T., *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. 1996, New York, USA: Oxford University Press Oxford.

179. Banharnsakun, A., T. Achalakul, and B. Sirinaovakul, *The best-so-far selection in Artificial Bee Colony algorithm.* Applied Soft Computing, 2011. 11: p. 2888-2901.

180. Akay, B. and D. Karaboga, *A modified Artificial Bee Colony algorithm for real-parameter optimization.* Information Sciences, 2012. 192: p. 120-142.

181. Li, G., P. Niu, and X. Xiao, *Development and investigation of efficient artificial bee colony algorithm for numerical function optimization.* Applied Soft Computing, 2012. 12: p. 320-332.

182. Karaboga, D. and B. Akay, *A comparative study of Artificial Bee Colony algorithm.* Applied Mathematics and Computation, 2009. 214(1): p. 108-132.

183. Chen, Q., B. Liu, Q. Zhang, P.N.S. J. J. Liang, and B.Y. Qu, *Problem Definition and Evaluation Criteria for CEC 2015 Special Session and Competition on Bound Constrained Single-Objective Computationally Expensive Numerical Optimization*. Nov 2014, Technical Report, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and technical report, Nanyang Technological University, Singapore.

184. Espinosa, J., J. Vandewalle, and V. Wertz, *Fuzzy logic, identification and predictive control*. 2005, United Kingdom: Springer-Verlag London Limited.

185. Takagi, T. and M. Sugeno, *identification of systems and its applications to modeling and control.* IEEE transactions on systems, man and cybernetics, 1985. SMC-15: p. 116-132.

186. Jiang, H., C.K. Kwong, Z. Chen, and Y.C. Ysim, *Chaos particle swarm optimization and T–S fuzzy modeling approaches to constrained predictive control.* Expert Systems with Applications, 2012. 39(1): p. 194-201.

187. Hadjili, M.L., *Fuzzy logic in nonlinear modeling and control*. Université Catholique de Louvain: Belgium.

188. Shridhar, R. and D.J. Cooper, *A Tuning Strategy for Unconstrained SISO Model Predictive Control.* Industrial & Engineering Chemistry Research, 1997. 36: p. 729-746.

189. Venkateswarlu, C. and A.D. Reddy, *Nonlinear Model Predictive Control of Reactive Distillation Based on Stochastic Optimization.* Industrial & Engineering Chemistry Research, 2008. 47(18): p. 6949-6960.

190. Grosman, B. and D.R. Lewin, *Automated nonlinear model predictive control using genetic programming.* Computers & Chemical Engineering, 2002. 26(4–5): p. 631-640.

191. Onnen, C., R. Babuška, U. Kaymak, J.M. Sousa, H.B. Verbruggen, and R. Isermann, *Genetic algorithms for optimization in predictive control.* Control Engineering Practice, 1997. 5(10): p. 1363-1372.

192. Coelho, J.P., P.B. de Moura Oliveira, and J.B. Cunha, *Greenhouse air temperature predictive control using the particle swarm optimisation algorithm.* Computers and Electronics in Agriculture, 2005. 49(3): p. 330-344.

193. Ait sahed, O., K. Kara, and M.L. Hadjili. *PSO-based fuzzy predictive control*. in *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*. 2012.

194. Hadjili, M.L., K. Kara, O. Ait sahed, and J. Bouyanzar, *Fuzzy Predictive Control Using Particle Swarm Optimization: Application to SCARA Robot.* Applied Mechanics and Materials, 2014. 527: p. 230-236.

195. Ying, S., C. Zengqiang, and Y. Zhuzhi, *New Chaotic PSO-Based Neural Network Predictive Control for Nonlinear Process.* Neural Networks, IEEE Transactions on, 2007. 18(2): p. 595-601.

196. Chang, W.-D., *Nonlinear CSTR control system design using an artificial bee colony algorithm.* Simulation Modeling Practice and Theory, 2013. 31: p. 1-9.

197. Bose, D., S. Biswas, A.V. Vasilakos, and S. Laha, *Optimal filter design using an improved artificial bee colony algorithm.* Information Sciences, 2014. 281(0): p. 443-461.

198. Zhou, J., X. Zhang, G. Zhang, and D. Chen, *Optimization and Parameters Estimation in Ultrasonic Echo Problems Using Modified Artificial Bee Colony Algorithm.* Journal of Bionic Engineering, 2015. 12(1): p. 160-169.

199. Govardhan, M. and R. Roy, *Generation scheduling in smart grid environment using global best artificial bee colony algorithm.* International Journal of Electrical Power & Energy Systems, 2015. 64(0): p. 260-274.

200. Contreras-Cruz, M.A., V. Ayala-Ramirez, and U.H. Hernandez-Belmonte, *Mobile robot path planning using artificial bee colony and evolutionary programming.* Applied Soft Computing, 2015. 30(0): p. 319-328.

201. Benyoucef, A.s., A. Chouder, K. Kara, S. Silvestre, and O. Ait sahed, *Artificial bee colony based algorithm for maximum power point tracking (MPPT) for PV systems operating under partial shaded conditions.* Applied Soft Computing, 2015. 32(0): p. 38-48.

202. De Moor, B.L.R., *DaISy: Database for the Identification of Systems*, E.S. Department of Electrical Engineering, K.U.Leuven, Editor.: Belgium.

203. Pellegrinetti, G. and J. Bentsman, *Nonlinear Control Oriented Boiler Modeling-A Benchmark Problem for Controller Design.* IEEE Transactions on Control Systems Technology, 1996. 4: p. 57-64.

204. Espinosa, J. and J. Vandewalle. *Predictive Control Using Fuzzy Models Applied to a Steam Generating Unit*. in *proceedings of the 3rd International Workshop on Fuzzy Logic and Intelligent Technologies for Nuclear Science and Industry*. 1998.