Multi Objective Optimization of Control Parameters for Auto-Steering of Off-Road Vehicles

Hamed Mahdizadeh

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the Degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
March 2014
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Elvan Yılmaz
Director

I certify that this thesis satisfies the requirements as a thesis for the degree Master of Science in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Asst. Prof. Dr. Mehmet Bodur
Supervisor

Examining Committee

1. Asst. Prof. Dr. Adnan Acan
2. Asst. Prof. Dr. Mehmet Bodur
3. Asst. Prof. Dr. Ahmet Ünveren

ABSTRACT

In this thesis, the lateral controller parameters of an agricultural tractor vehicle were optimized to reduce both the root mean square error (E_{RMS}), and the peak error (E_{peak}), by using two evolutionary multi-objective optimization algorithms. The lateral controller of a tractor provides tracking of a desired path with minimum lateral error, which enhances the efficiency of agricultural plantation since many processes in agriculture require tracking a desired path.

The evolutionary multi-objective optimization algorithms: NSGA-II and MODE are commonly used search algorithms to find the Pareto-front of the optimal solutions for multiple fitness functions. In parameter optimization of the lateral controller, two fitness functions, E_{RMS} and E_{peak}, were evaluated along a predefined reference path of tracking through the simulation of the tractor motion in an agricultural field.

Results of the optimization by both methods supported each other closely, and the optimization reduced the error figures down to 0.0016 m E_{peak}, and 0.0004 m E_{RMS}. The obtained Pareto-front can be used to compromise between the E_{peak} and E_{RMS} in setting the controller parameters best way for the conditions of the application.

Keywords: NSGA-II - MODE - Lateral Error – Auto-Steering Control

ÖZ

Bu tezde, otomatik sürüş denetimli bir tarım aracının sürüş denetleç parametreleri, evrimsel çok-amaçlı optimizasyon algoritması kullanarak, hem hatanın karesinin ortalamasının kökünu (E_{RMS}) hem de hatanın tepe değerini azaltmak üzere optimize edildi. Traktörün sürüş denetleci takip edilecek yolun en az yanal hata ile izlenmesini sağlar. Tarımda takip edilecek bir yolun izlenmesini gerektiren bir çok süreç bulunduğundan yanal hata azalınca tarımda ekim verimi de artar.

Çok amaçlı evrimsel optimizasyon algoritmaları olan NSGA-II ve MODE genellikle birden fazla uygunluk fonksiyonlu problemler için optimum çözümlerdeki Pareto-önü elde etmek için kullanılan arama algoritmalarıdır. Yanal denetim parametrelerinin optimizasyonunda, uygunluk fonksiyonları olarak kullanılan rmse ve tepe hata, traktörün önceden seçilmiş bir izlenecek yol boyunca simülasyonu yoluyla elde edilmiştir.

Her iki yöntemle optimizasyon sonuçları birbirlerini yakından desteklemektedir. Optimizasyon sonucu bulunan parametrelerden elde edilen yanal hatanın rmse değeri 0.0004 m'ye, tepe değeri 0.0016 m'ye kadar düşmüştür. Pareto-önden elde edilen parametrelerden uygulama koşullarına bağlı olarak tepe ve rmse hatalar arasında en iyi uzlaşma sağlayacak olanını kullanmak mümkündür.

Anahtar Kelimeler: NSGA-II - MODE - Yanal hata – Otomatik sürüş denetimi

ACKNOWLEDGMENT

I am very grateful for the completion of this thesis. At first, I thank God for giving me the strength and potential to continue, guidance and opportunity to pursue the present study up to a concluding level. Secondly, my immense gratitude goes to my family for growing the science passion in sacrifices as well as their encouragement and me even when I almost gave up.

I want to thank some important people without whom I would not have been able to complete this research work. Firstly, my supervisors: Asst. Prof. Dr. Mehmet Bodur, for his professional guidance and time. Secondly, my profound appreciation goes to those who helped me in one way or the other to contribute great ideas and advices especially my classmates and close friends for without them, this study would not be possible.

Finally, my all appreciation also goes to my darling friend, Paniz for her advice, time and support. I must say you are a good friend indeed.

To My Family

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

NSGA-II: Non-Domination sorting Genetic Algorithm

MODE: Multi Objective Differential Evolution

RMSE: Root Mean Square Error

Chapter 1

INTRODUCTION

Automation and introduction of robots in industrial and agricultural production are typical demands of the contemporary industrialization. Consequently, the dynamic control of robots has been an important research field in systems control and artificial intelligence areas [1]. The tuning of controller parameters has been studied using various approaches including adaptive [2], and evolutionary search methods [3].

Modernization and automation of agricultural processes are necessary to satisfy the qualitative and quantitative market demands of the increased population. Automation of farming tasks on a field requires incorporation of many technologies. Energy savings by agricultural automation may enhance production efficiency, which can result in remarkable financial benefits. The efficiency of the agricultural production depends mainly on the terrain properties, surrounding environment conditions, and the precision of the farming machines, including the path tracking accuracy of the agricultural machines. The high lateral error at the curvature transitions is a common problem in path tracking control systems of existing autonomous agricultural machine technology. The lateral tracking error of auto-steering tractors, which is an important factor on efficiency of the automated agriculture, has two main components, the peak error, and the root-mean-square error, along the path. Among the auto-steering lateral error control systems, the double look-ahead reference point

(DLARP) controller provides the best results provided their controller settings are optimal [4].

The main topic of this thesis is to obtain the optimal controller parameters of a DLARP controller by means of evolutionary optimization methods. The tracking process has mainly two kinds of independent lateral errors: peak and rmse. A multi-objective optimization method is necessary to minimize these two independent errors simultaneously [4].

The search of a set of solutions for a multi-input fitness function by testing and evolving generations of population forms an evolutionary search algorithm. The evolutionary algorithms are considered an alternative to solve difficult optimization problems, relied on mainly genetic algorithms and evolutionary strategies. Among many evolutionary algorithms, this study has focused on differential evolution (DE) and genetic algorithms.

Darwin's evolution theory together with the advanced knowledge of genetics explains the steady change of species to evolve in the direction of fitness to the natural conditions of the life by three mechanisms of the chromosomes in a population: the mutation of a gene, the crossover of the genes, and the selection of genes for reproduction. Genetic algorithms are inspired from this theory to search the optimum solution of a mathematical function to minimize a cost function which is also called an objective function, or a fitness function. An evolutionary algorithm starts with a population of chromosomes, which corresponds the parameters of candidate-solutions. It selects a number of chromosomes according to their fitness scores as parents. It uses a kind of mutation and crossover on the parent

chromosomes to get the child chromosomes that forms the population of the next-generation. The algorithm needs several generations to reduce the fitness function to a satisfactory level using crossover, mutation, and selection operators [15].

The NSGA-II algorithm is non-dominated sorting genetic algorithm. It is a genetic algorithm modified for deployment of the solutions using multiple objective optimizations function. The algorithm uses an evolutionary process to develop next generations by evolutionary operators including selection by multiple fitness functions, genetic crossover, and genetic mutation [5].

The differential evolution (DE) algorithm uses a simple mutation operator based on differences between pairs of solutions with the aim of finding a search direction based on the repartition of solutions in the current population. DE also utilizes a steady-state-like replacement mechanism, where the newly generated offspring competes only against its corresponding parent and replaces it if the offspring has a higher fitness value. DE uses similar computational steps of typical standard evolutionary algorithm. The DE-variants perturb the chromosomes in the current-generation population with the scaled differences of randomly selected and distinct population members, without needing the probability distribution of the population while generating the offspring population [7].

The next chapter describes the dynamics of an auto-steered tractor and its lateral DLARP control unit, which is modelled to simulate the motion of the tractor along a typical desired path for the evaluation of the peak and the rmse value of the lateral error along that desired path as the two fitness functions of the evolutionary optimization processes. The third and fifth chapters introduce NSGA-II and MODE

algorithms on two simple examples. The fourth and sixth chapters display the results of NSGA-II and MODE on the optimization of the DLARP controller parameters. The last chapter concludes the overall results of the thesis.

The third chapter explains NSGA-II and its application to obtain the pareto-front of the steering controller settings. The forth chapter explains MODE to determine the Pareto-front of the steering control parameters to reduce booth peak and RMS errors. The fifth chapter contains a discussion about the effect of Pareto-optimal solution on the lateral error of the tractor. The sixth chapter contains a conclusion of the thesis.

Chapter 2

DLARP AS AUTO STEER CONTROL OF A TRACTOR

2.1 Lateral Control of a Tractor

An agricultural tractor mostly steered along a predefined reference trajectories. Driving the tractor along these trajectories is a tedious task, and manual driving mostly results in considerable deviation from these desired reference trajectories. The driving task is accomplished automatically using a lateral controller that decides on the steering angle of the front wheels of a typical four wheel tractor. At a typical 8 m/s speed, a manually driven tractor may typically have lateral error around 0.2 m in average while tracking a line, and up to 1.2 m peak error is easily observable especially at the transients of the curvatures. For automatic steering of the tractors, the best performing lateral control method in the literature is obtained by DLARP, double look-ahead reference point control law [4].

A four wheel vehicle with front wheel steering moves tangential to the rear and front tyres if the friction forces on the tyres stops sidewise slip movements. But the soil is not a solid ground therefore slip and skid is considerably large for agricultural applications [4].

2.1.1 Non-Holonomic Tangential Motion and Side Slip Motion of a Tractor

A typical four wheel tractor on a solid ground is expected to move tangential to the rear and front tyre directions. They are equipped with Ackerman type steering mechanism that directs the front wheels to a desired steering angle δ within the upper

and lower bounds of the mechanism. The kinematic motion of the tractor is called non-holonomic because of this tangential motion constraint. However, a tractor on the typical agricultural soil makes considerable amount of sidewise slip motion together with the translational skid motion. Thus, a tractor floats on the soil surface because of lateral forces on the tyres while it moves forward tangential to the tyres [4]. The motion of the tractor is mostly modelled by shifting left and right tyres on the central axis to simplify the tractor dynamics to a bicycle as seen in Figure 1.

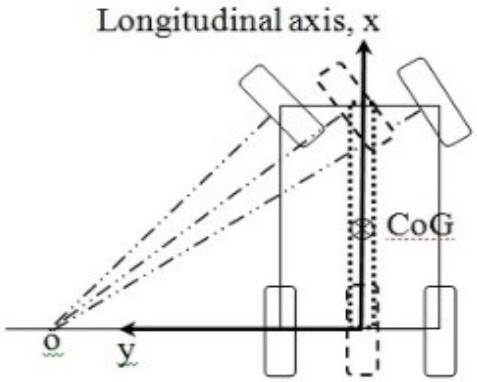

Figure 1: Non-holonomic Motion of 4-wheel Tractor
with Ackermann Steering Mechanism while turning about the center ‘o’. Dashed figure is bicycle representation of the tractor.

2.1.2 Fundamental Laws and Constraints of Motion for Dynamic Simulation

The motion of a bicycle-tractor on the loose soil surface may be modelled considering the following laws and constraints: a) Newton’s law of acceleration that explains the relation between the force on a body and the linear acceleration of the body on the direction of the force, b) Euler’s law of angular acceleration that determines the direction of the motion as a result of torque on the tractor body, c) tangential constraints of holonomic motion that relates the steering direction to the

side slip forces on the tyres and results in change of direction of the tractors motion, d) the random effect of the soil clods on the side slip forces that gives a random disturbance to the direction of motion. All of these forces are formulated in [4] for the successful simulation of motion of an agricultural machine by the MatLab codes in Appendix 1 [4].

In addition to the coding of the equation of motion, the simulation of the motion of an auto-steering tractor requires two main components: a desired trajectory which is described by a sequence of points in a plane, and a simulation of a control law that governs the steering angle as a function of the states of the tractor and the observed deviations from the desired path. There are commonly used desired test paths to test the performance of the system that contains typical common patterns of tractor paths in an agricultural field, such as a circular section between two lines. The control law DLARP is known as one of the most successful control laws to reduce the lateral deviations from the desired path [4].

2.2 Lateral Control Law of Auto-Steerring Tractor

The aim of an auto-steering control law is to keep the tractor on the desired reference trajectory with minimum deviation from the path. The deviation from the path at a given time is measured by the distance d_N from the centre of gravity of the tractor to the nearest point on the reference path. This distance is called the lateral displacement error, the lateral error, or shortly error. The quality of an auto-steering control law may be determined by measuring this error along a typical desired reference trajectory that contains commonly used components of desired path sections for a typical agricultural activity. These typical desired trajectories are called *test trajectory* for performance measurement of the control law [4].

Along with the lateral displacement error, the directional displacement error is also an important parameter considering the control of the tractor motion. However, directional error has negligible effect compared to the effect of the lateral error on the agricultural product efficiency [1],[4].

2.2.1 Peak and RMS Lateral Displacement Errors

The peak error, E_{peak}, of the tractor is the maximum absolute lateral displacement along the test path. It is an important performance criterion for the performance of the tracking control since a large peak error means a large deviation from the test path. The peak error occurs especially when the tractor is taking the corners or making U-turns. The advanced adaptive control laws make their peak error especially at the points of curvature transition, until the adaptive law reduce the error to the minimum level right after the curvature is changed [1],[4].

If the motion of the tractor is stable, along the linear or circular parts of the test path, any deviation from the path converges asymptotically to zero. A good measure of the performance is the root of the mean of the squared error, shortly abbreviated by RMSE and shown by the symbol E_{RMS}.

$$E_{\text{peak}} = \max_{t=0}^{T_e}(|d_N(t)|) \tag{1}$$

$$E_{\text{RMS}} = \sqrt{\frac{1}{T_e}\int_{t=0}^{T_e}(d_N(t))^2 dt}\,. \tag{2}$$

In a computer simulation, peak and RMSE can be easily calculated using N samples of d_{N} which are observed periodically with time steps $T=T_e/N$.

$$E_{\text{peak}} = \max_{n=1}^{N}(|d_N(n)|) \tag{3}$$

$$E_{\text{RMS}} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(d_N(n))^2} \tag{4}$$

Both E_{peak} and E_{RMS} are non-negative real numbers determined by the calculation of the lateral deviations d_{N} of the tractor along a desired reference test path [4].

2.2.2 Desired Test Path of the Simulations

The *desired reference test path*, shortly the *desired test path* or *desired path* is represented by a sequence of coordinates which shall be tracked by the auto-steered tractor. In Appendix 1, the program "*pathmaker.m*" generates a typical desired path for the simulation. In the coding, the first line determines the arguments of the pathmaker function. The arguments *x0* and *y0* are the coordinates of the start point of the path, *dt* is the duration of timesteps of the points on the path, *speed* is the forward speed of the tractor, *firstlength* is the length of the first linear section that goes along x coordinate, radius is the radius of the *u-turn*, and *lastlength* is the *lenght* of the linear section after the *u-turn*. The second line gives a typical example for the arguments, which is employed to generate all desired paths of the simulation runs in this thesis. Lines 4-7 of the code opens a file named *pathdef.m* and puts the headings of the columns in the file. Lines 8-11 generate the points along the first linear section of the path. Lines 12-16 generate the u-turn of the path as a 180 degrees semi-circular section. Lines 17-20 generate the linear part after the u-turn. Lines 21-23 terminate the *path* matrix in *pathdef.m*, and finally close the file *pathdef.m*, which returns the generated matrix of path to the simulation code with the name *path*. The matrix path contains around 6700 rows of vectors. Each row is made of {s, x, y, a, i}, where s is the curvilinear distance of the vector from the start point, x and y are the

positional coordinates x and y, a is the approach angle of the path in radians, and i is the identification number of the vector starting from 1 for the first row [1],[4].

2.2.3 Look-ahead Reference Point Control

The look-ahead reference point control law proposes a control rule which calculates the steering angle δ_{des} by four terms based on the nearest (normal) point P_{N} on the path, and two look-ahead reference points P_{L1} and P_{L2}.

$$\delta_{des} = K_d d_N + K_N \theta_N + K_1 \theta_1 + K_2 \theta_2 \tag{5}$$

where, d_N is the lateral deviation of the tractor to the path which is measured by the distance from P_N to the centre-of-gravity (CoG) of the tractor; θ_N, θ_1 and θ_2 are angular deviations between the heading angles of the tractor and the path points P_{N}, P_{L1} and P_{L2}; $\{K_d, K_{\text{N}}, K_1, K_2\}$ are controller coefficients, which are the control parameters to be searched to reduce d_N to a reasonably low level along the desired test path. Moreover, the points P_{L1} and P_{L2} are at the distances L_1 and L_2 from the point P_{N} as seen in Figure 2. The four controller coefficients together with the two look-ahead distances form the parameter set of the lateral control [1].

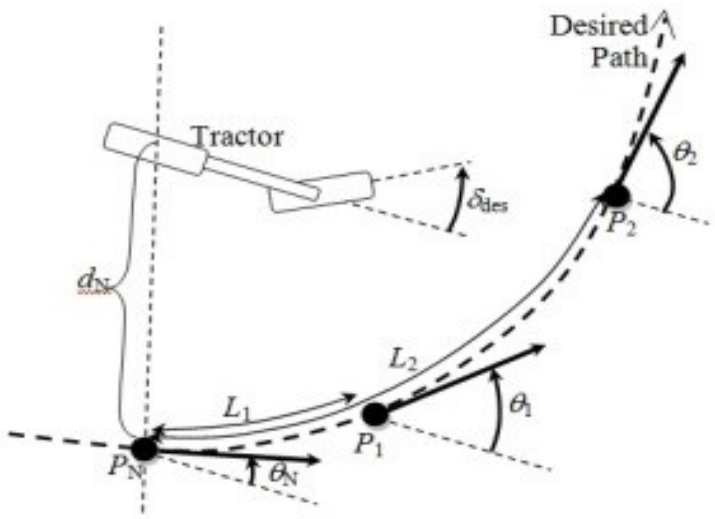

Figure 2: Illustration of related variables of auto-steering control

Although there are total six controller parameters in the control law, there are two constraints to be satisfied by the controller for a successful tracking along a line and along a circle. The first constraint is related to the stability along a linear path, and it specifies that $K_N+K_1+K_2=K_{NLine}$. This constraint is easy to verify analytically since $\theta_N=\theta_1=\theta_2$ while the desired path is a line. Its value is obtained along a linear test path as K_{NLine}= 5.6. The second constraint is related to the compensation of the centrifugal forces along the circular movements. It specifies that $K_{LCirc} = K_1 L_1 + K_2 L_2$ shall be constant to compensate the centrifugal effect of circumflex on a circular path. The value K_{LCirc} =2.28 is easily obtained by testing the tractor motion along the circular section of the test path. These constraints are employed to determine i) K_1 from K_2, L_1, and L_2 and then ii) K_N from K_1 and K_2 using

$$K_1= (K_{LCirc}-K_2 L_2)/L_1; \text{ and } K_N=K_{NLine}-K_1-K_2. \quad (6)$$

Consecutively, introducing the constants $\{K_{LCirc}, K_{NLine}\}$ reduces the number of independent parameters to four, namely $\{K_D, K_2, L_1, L_2\}$ [1],[2].

2.3 Evaluation of the Fitness Functions

The MatLab code of the fitness function is listed in the Appendix 1 with the file name *fitness2.m*. In this code, the function is called by six parameters, corresponding to $\{K_N, K_D, K_1, K_2, L_1,$ and $L_2\}$, and it returns two values, E_{peak}, and E_{RMS}. Lines 3 to 6 contain the settings for path, and permission of three kinds of plots. Lines 7 to 19 initialize the coefficients for the simulation of the motion of the tractor, which is described by Bevly and Derrick as described in [4]. Lines 20 and 21 are related to the linear and circular tests, and obsolete for the search of the best parameters. Lines 22 to 26 contain the typical values of control parameters, and the linear and circular path constraints. Lines 27 to 37 initialize the simulation variables of the test to set the tractor to the initial point. Line 38 is the start of the simulation loop. Lines 39 to 45 update the time, and several Cartesian and angular coordinates related to the motion of tractor. Lines 46 to 57 search the normal point d_N on the desired path and finds steer angle. Lines 58 to 70 calculate the position of look ahead points and then find angular deviation for each point. Line 73 applies control law to the tractor once at every 50ms period. Line 78 provides shortcut to quit if the controller settings are unsuccessful. Lines 81 to 84 simulate the hydraulic servo-actuator of the tractor. Lines 86 to 102 contain the equation of motion for local longitudinal and lateral acceleration and velocities of the tractor. Lines 103 to 105 integrate the velocities to the position of the tractor in absolute coordinate frame. E_{peak} and E_{RMS} are calculated at lines 116 and 117. Lines 112 to 128 contain the codes of plots [2],[4].

Chapter 3

NON-DOMINATED SORTING GENETIC ALGORITHM-II

3.1 Evolutionary Optimization

Evolutionary optimization algorithms have been inspired from Darwin's Evolution Theory and the Genetics Science that explains the evolution of the population to have higher survival rate with every new generation. The first idea started with Genetic Algorithms (GA) to search optimum solutions for *general single-objective optimization problems*.

3.1.1 Single-Objective Optimization

A *single-objective optimization problem* is defined as a search of solution x that minimize a scalar $f(x)$ satisfying the constraints $g_i(x) \geq 0$ and $h_i(x)=0$ in the universe of solutions $x \in \Omega$. In many applications, the constraints g and h may not exist.

In general case of single-objective optimization, the vector of decision variables $x=(x_1, x_2, x_3, \ldots, x_n)^T$ is n-dimensional, and the function $f(x)$ is a scalar from R^n to R. The function $f(x)$ may have a single minimum, or multiple minimum points. The problem of searching global minimum of *f(x)* forms a *single-objective global minimum optimization problem*.

3.1.2 Evolutionary Optimization Algorithms

In evolutionary optimization algorithms, an *individual* or a *chromosome* corresponds to a vector $x=(x_1, x_2, x_3, \ldots, x_n)^T$ in the universe of solutions Ω. Each of the components of the solution vector is called a *decision variable*, or a *gene*.

Darwin's concept of *"Survival of the fittest"* matches to score an individual by the scalar value of $f(x)$. Consequently, $f(x)$ as a score is called an objective function, or a *fitness* function.

The algorithm works iteratively on a *population of candidate solutions,* in other terms *population of individuals*, which is simply a collection of chromosomes. Each iteration is called a *generation*. The offspring population is generated from *parent population* by a set of evolutionary operations inspired from nature, such as, *recombination*, and *mutation*. A *selection* operator determines the population of the next generation. A *termination condition* determines the end of the algorithm. The termination condition of most algorithms is based on count of iterations, count of fitness evaluation, or CPU time.

3.1.3 Crossover, Mutation, and Selection Operators

Mutation corresponds to simply random change of one of the decision variables of an individual, or equivalently it corresponds to a random change of a gene in a chromosome. The mutation operator generates an offspring individual from an individual $x=(x_1, x_2, x_3, \ldots, x_n)^T$ by randomly changing at least one of the decision variables.

Crossover generates offspring individuals by combining suitable parts of at least two parent solutions. One of many existing crossover methods may be used depending on

the structure of the problem and its representation. In simplest form, single point crossover requires randomly determination of the crossover point, where first part of the parent is combined to the second part of another parent to form two offspring. Crossover operation may generate a better offspring individual in fitness compared to both parents by combining the contributive parts of the parents to an offspring. More elaborated two-point crossover operator requires two randomly selected crossover points to replace the inner part of the chromosomes of the two parents.

An *evolutionary algorithm* applies crossover and mutation operations on the current population of the solutions to generate a higher number of offspring population compared to the current population. The selection operator determines the individuals that will take part in the population of the next generation. From many selection methods, elitist selection takes the individuals with the best fitness values into the next population.

Elitist selection has disadvantages such as easily locking to local optimums instead of searching for global optimum. The tournament selection method selects the best of the randomly selected set of individuals, allowing some of the less fit individuals into the next generation. Similarly, to overcome the same problem, the *fitness proportionate selection method*, also called *roulette wheel method*, uses the probability $p_i=f_i/(\Sigma_{j=1\ldots N} f_j)$ while selecting among N individuals.

3.1.4 Genetic Algorithm

A genetic algorithm is defined on a fitness function $f(x)$, with solution universe $x \in R^n$, a parent population size $p \in Z^+$, a larger offspring population size $q \in Z^+$, the evolutionary operators c: crossover operator, m: mutation operator, and s: selection operator, and the termination condition χ by the following structure:

start with t=0; and initial population $P(0)=\{ x_1, \ldots, x_p\}$,

while(termination condition χ not satisfied) *do*

apply crossover to get offspring population: $P'(t)=c(P(t))$,

apply mutation on the offspring population: $P''(t)=m(P'(t))$,

select next population: $P(t+1)=s(P(0),P''(0))$,

update iteration count: $t=t+1$,

enddo.

The termination condition is mostly specified on number of generations.

3.1.5 Multi Objective Optimization

A *multi-objective optimization problem* is similar to single objective problem, but, it is defined by a search of solutions x that minimize a multi-valued objective function $F(x)$ in the universe of solutions $x \in \Omega$. The problem of searching global minimum of $F(x)$ forms a *multi-objective global minimum optimization problem.*

Since $F(x)$ is not scalar, the minimum of the objective function is not a scalar value. There are many methods to deal with the multi-objective optimization problems, such as converting multi-valued objectives to a scalar, for example, by introducing some weighting factors. However, Vilfredo Pareto's idea of Pareto-Optimality forms a very common base to determine the Pareto-front of the multi objective optimization problems.

3.1.6 Pareto Optimal Solutions

A point $x^* \in \Omega$ is a *weakly Pareto optimal* if there is no $x \in \Omega$ that satisfies $x^* \neq x$ and $f_i(x^*) < f_i(x)$, for $i=1,\ldots, k$. That means, there is no other solution x which is better than x^* for any objectives of the multi-objective optimization problem. Furthermore, A point $x^* \in \Omega$ is a *strict Pareto optimal* if there is no $x \in \Omega$ that satisfies $x^* \neq x$ and $f_i(x^*) \leq f_i(x)$, for $i=1,\ldots, k$. Accordingly, a solution is pareto-optimal if it is not-dominated by any other solution in decision variable space. A pareto-optimal solution is the best optimal solution with respect to all objectives, and, it cannot be improved in any objective without worsening in another objective. Therefore, terms non-dominated solution and Pareto-optimal refers exactly to the same concept.

3.1.7 Pareto Front of a Multi-Objective Problem

For a given multi-objective problem with the objective function $F(x)$, and Pareto optimal set P^*, the Pareto front PF^* is collection of all non-dominated vectors in the objective function space, $PF^* = \{u = F(x) \mid x \in P^*\}$.

In other words, the values of objective functions related to each solution of a Pareto-optimal set in objective space are called Pareto-front.

The Pareto-front of a problem has a significant importance in determination of the best values for the decision variables since it directly displays the values of the objective functions corresponding to the Pareto optimal. This thesis is focused on obtaining the Pareto-front for the for the automatic steering controller parameters of a tractor by evolutionary methods to minimize both its lateral peak and lateral RMS error.

3.1.8 Non-dominated Sorting Genetic Algorithm (NSGA)

One of the commonly used popular multi-objective genetic algorithms is NSGA. It is known as a very effective optimization algorithm, but, it has been generally criticized for its computational complexity, lack of elitism and difficulty to set sharing parameter values [9].

3.1.9 Dominance Rank of Individuals

In multi-objective evolutionary algorithms, the selection is based on the measures of dominance, such as dominance rank, dominance count and dominance depth. The *dominance rank r* of an individual *x* is the number of individuals that dominates *x* plus 1. The dominance rank is used in selection operator of NSGA to converge the population to a Pareto optimal rich set.

Although NSGA converges to a Pareto optimal rich population, there is no mechanism to distribute the Pareto optimal solutions on the Pareto front homogenously. This disadvantage of the algorithm is defeated in further developed multi-objective evolutionary optimization algorithm, NSGA-II.

3.2 NSGA-II Algorithm

NSGA-II is improved from NSGA mainly in sorting method. Faster conversion rates were obtained by elitism. In addition, the initialization of a sharing parameter is eliminated from the algorithm. The diversity of Pareto optimal solutions in NSGA-II is obtained using "*crowding-distance*" density estimation method [5].

Elitism is name of method, which copies a number of best chromosomes (or a few best chromosomes) to new population, and uses other standard methods for selection

of the remaining. Elitism can very rapidly increase performance of GA, because it prevents losing the best-found solution [5].

As shown in Figure 3, the population in initialized the population is sorted based on non-domination into each front. Individuals are assigned fitness and rank values. Non-dominated individuals are assigned rank one, and individuals dominated only by one individual are assigned rank two. For each individual, *crowding distance* value is calculated as a measure of closeness of individual to its neighbors. Binary tournament method is applied to select parents from the population according to the rank and crowding distance. Individuals of a smaller rank and those with higher crowding distance are preferred with higher probability. Crossover and mutation operators are applied on the parent population to generate offspring population. The population with the current population and current offspring is sorted again based on rank and distance, and only the best N individuals are selected for the next population [5].

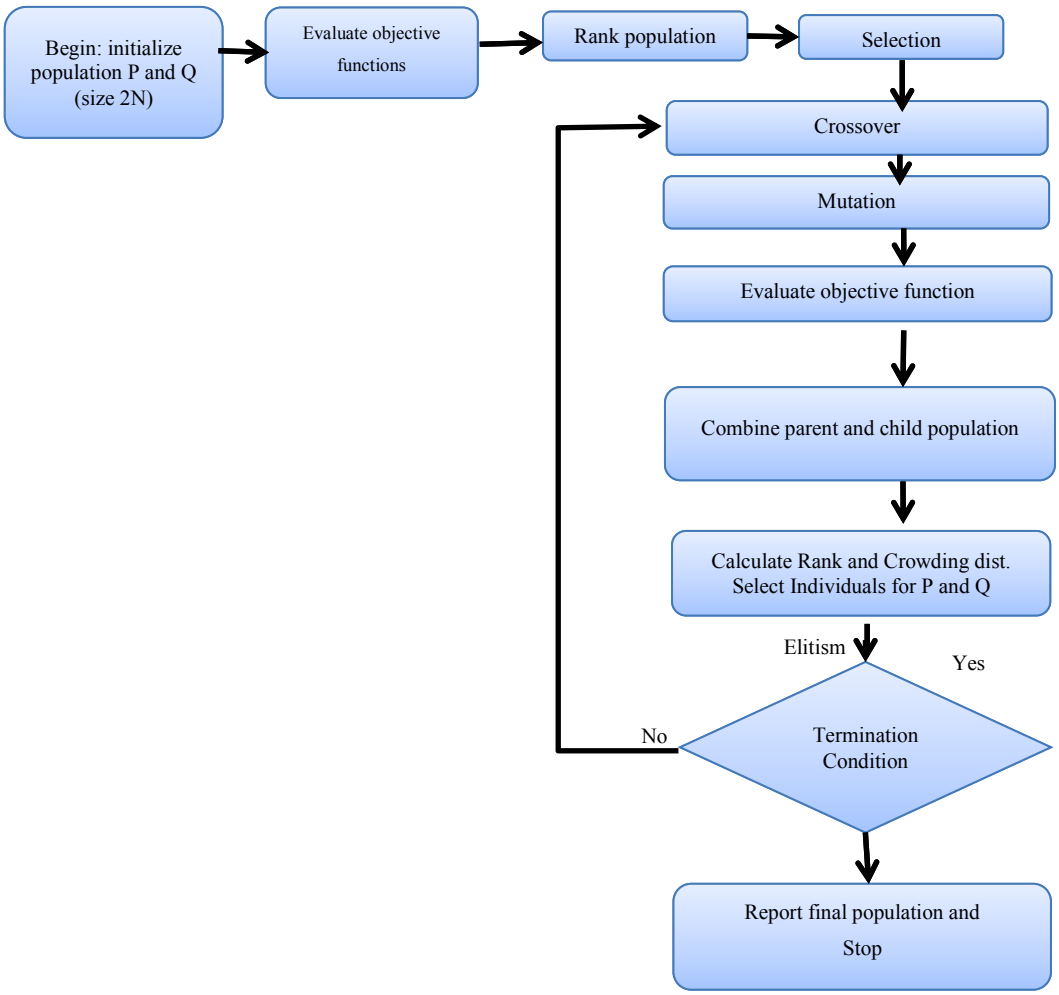

Figure 3: Flowchart for NSGA-II [5]

3.2.1 Initialization of Population of Chromosomes

Population is initialized by random values that are within the specified range. Each chromosome consists of the decision variables. Moreover, the value of the objective functions, rank and crowding distance information is added to the chromosome vector but only the elements of the vector, which has the decision variables, are operated upon to perform the genetic operations like crossover and mutation. The fitness of population is sorted by using non-domination-sort. This returns two columns for each individual, which are the rank and the crowding distance corresponding to their position in the front they belong. At this stage the rank and the

crowding distance for each chromosome is added to the chromosome vector for easy of computation [9].

3.2.2 Selection of Parents to Generate Child Population

Selection operator sorts the individuals based on non-domination and with crowding distance. The individuals are selected by using a binary tournament selection with crowed-comparison-operator [10].

NSGA-II uses binary tournament selection. In binary tournament process, the objective function of two randomly selected individuals is compared, and better one is selected as a parent. It is repeatedly carried out for the pool size, which is the number of parents to be selected. The tournament selection function has three major arguments: chromosomes, pool and tour. The function uses only the information from last two elements, the rank of domination and the crowding distance. Selection is based on rank, and crowding distance. A lower rank and higher crowding distance is the selection criteria [9].

3.2.3 Simulated Binary Crossover (SBX)

The simulated binary crossover is widely used in real valued genetic algorithms to generate and select the children $c_{i,k}$ from the parents $p_{i,k}$ with a spread factor $\beta_k \geq 0$ [8].

$$c_{1,k} = \frac{1}{2}\left[(1-\beta_k)p_{1,k} + (1+\beta_k)p_{2,k}\right] \tag{7}$$

$$c_{2,k} = \frac{1}{2}\left[(1+\beta_k)p_{1,k} + (1-\beta_k)p_{2,k}\right] \tag{8}$$

Children selected by this method have almost similar distribution, and similar search power compared to the one-point crossover with a binary coded search. The probability distribution of child solution is:

$$P(\beta) = \frac{1}{2}(\eta_c + 1)\beta_k \quad \text{if} \quad 0 \le \beta \le 1 \tag{9}$$

$$P(\beta) = \frac{1}{2}(\eta_c + 1)\frac{1}{\beta_k + 2} \quad \text{Otherwise} \quad , \tag{10}$$

where η_c is the distribution for crossover and the distribution can be obtained by sampling randomly in (0, 1) [8].

SBX algorithm starts with a uniform random number u in interval (0,1). If u<0.5 then β is calculated by

$$\beta(u) = (2u)^{\frac{1}{(\eta_c+1)}} \quad , \tag{11}$$

else, it is calculated by

$$\beta(u) = \frac{1}{[2(1-u)]^{\frac{1}{\eta_c+1}}} \ . \tag{12}$$

Higher η_c increases the probability of children closer to parents. Using β, the children are computed by (7) and (8).

3.2.4 Polynomial Mutation

The mutant gene of the child c_k is obtained from the parent gene p_k according to the upper and lower bounds of the gene, and a random variation δ_k.

$$c_k = p_k + \left(p_k^u - p_k^l\right)\delta_k \tag{13}$$

Where p_k^u and p_k^l are the upper and lower bound on the parent gene, and δ_k is calculated from a polynomial distribution by using a random number in (0,1) interval [8].

$$\delta_k = \left(2_{r_k}\right)^{\frac{1}{\eta_m+1}} - 1 \quad , \quad \text{if} \quad r_k < 0.5 \tag{14}$$

$$\delta_k = 1 - \left[2\left(1 - r_k\right)\right]^{\frac{1}{\eta_m+1}} \quad , \quad \text{if} \quad r_k \geq 0.5 \tag{15}$$

where η_m is mutation distribution index.

3.2.5 Next Generation

The current and new generations are combined as a solution pool. The solution pool is sorted using non-dominated sorting algorithm and crowding distance methods. Next generation is formed by choosing the members among the solution pool.

3.3 Application of NSGA-II for Controller Parameters

In this section, the searches of optimum control parameters of an automated farming vehicle are conducted to track a predetermined path on a loose soil surface. In optimizing the formulated problem, we require the minimization of both E_{RMS} and E_{peak}, which are measures of the lateral error of tracking on the linear and circular

paths. The problem has a multi-objective character and suitable for searching Pareto-optimal surface by NSGA-II algorithm. As explained in Chapter 2, the lateral error E_{peak} and E_{RMS} depends on the controller settings K_D, K_2, L_1, and L_2. The chromosomes of NSGA-II algorithm are composed of four genes, and four attachments: the two objective functions $f_1=E_{peak}$ and $f_2=E_{\text{RMS}}$, R=rank and CD=crowding distance.

3.3.1 Initialization of the Population

At the initialization phase of the NSGA-II algorithm the initial population size is set to 100 chromosomes, and the crossover and mutation ratio of the algorithm were set to 0.8 and 0.2. The genes of the chromosomes were generated randomly within the boundaries (-5, 5). The fitness values of each chromosome are attached to the chromosomes as seen in Figure 5.

K_{D}	K_2	L_1	L_2	E_{peak}	E_{RMS}

Figure 4: Chromosome in Population

The fitness values, the rank of domination, and the crowding distance of each chromosome are attached to the chromosomes, and the selection process was carried using these attached decision variables. The selected individuals were processed by crossover and mutation procedures. Table 1 contains the first ten individuals from the initialized population and their attached fitness values.

Table 1: Initialized Population and Evaluated Objective

K_D	K_2	L_1	L_2	$f_1=E_{peak}$	$f_2=E_{RMS}$
-3.1565	0.8315	1.5844	-0.919	0.0322	0.00921
1.3115	-1.9286	0.6983	3.868	0.0333	0.03570
0.9143	-2.0292	0.6954	-0.7162	0.0354	0.0088
-1.3897	-0.3658	0.9004	2.9311	0.0358	0.00576
0.3575	1.5155	-1.4868	-0.2155	0.0376	0.00126
2.3110	-0.6576	2.4121	-4.9363	0.0565	0.00187
-3.915	0.9298	-0.6848	0.9412	0.0630	0.00156
0.6294	-2.8116	-2.7464	0.8268	0.0741	0.00197
1.2647	-0.8049	-0.4431	4.6336	0.0808	0.03105
-0.4462	-2.9077	1.8057	2.6469	0.0300	0.00518

3.3.2 Binary Crossover

Binary crossover operator processed two selected parents and generated two offspring. As an example, let the parents be crossed at a random gene marked by superscripts (1) and (2):

p_1= (**-3.1565, 0.8315$^{(1)}$, 1.5844, -0.919**) and p_2= (**1.3115, -1.9286$^{(2)}$, 0.6983, 3.868**).

The offspring are composed of the parents by replacing the selected gene, i.e,

o_1= (**-3.1565, -1.9286$^{(2)}$, 1.5844, -0.919**), and o_2= (**1.3115, 0.8315$^{(1)}$, 0.6983, 3.868**).

This operation is carried for all genes of the selected parents with the probability that is specified by the *crossover ratio*.

3.3.3 Polynomial Mutation

The mutation operator is applied on a randomly selected gene of the selected offspring with a probability specified by *mutation ratio*. For example let the underlined gene of offspring o_1 be the randomly selected gene.

o_1= (**-3.1565, -1.9286$^{(2)}$, <u>1.5844,</u> -0.919**).

The underlined gene is replaced by a random number in the bounds of that gene:

c_1= (**-3.1565, -1.9286$^{(2)}$, <u>2.1473,</u> -0.919**).

3.3.4 Selection

Non-dominated sorting is carried out by sorting the chromosomes in their rank of domination, and in each rank in their crowding distance. For this purpose, the rank and crowding distance of each chromosome is calculated, and attached to the chromosomes. By tournament selection, the chromosomes are selected starting from the best-ranked (rank1) solutions. Selected chromosomes are placed in the archive. Rank-2 and further rank solutions were selected with decreasing probabilities.

3.3.5 Iteration of Generations

The procedures described by 3.3.2 to 3.3.4 are carried iteratively to select the best solutions into the archive. Figure 5 shows the Pareto-front after three hundred generation.

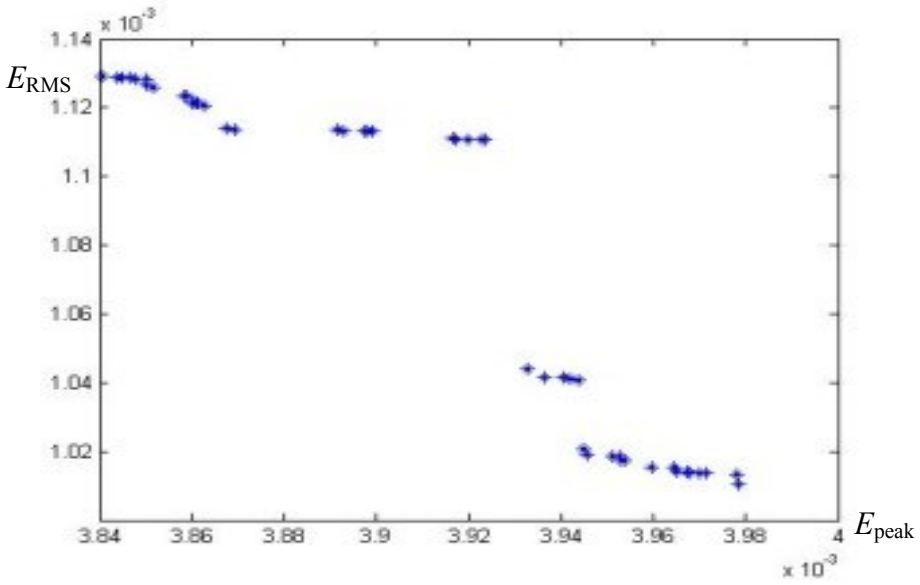

Figure 5: Pareto Front for NSGA-II after 300 Generations

Similarly, Figure 6 illustrates the best non-dominated solutions after one thousand generations.

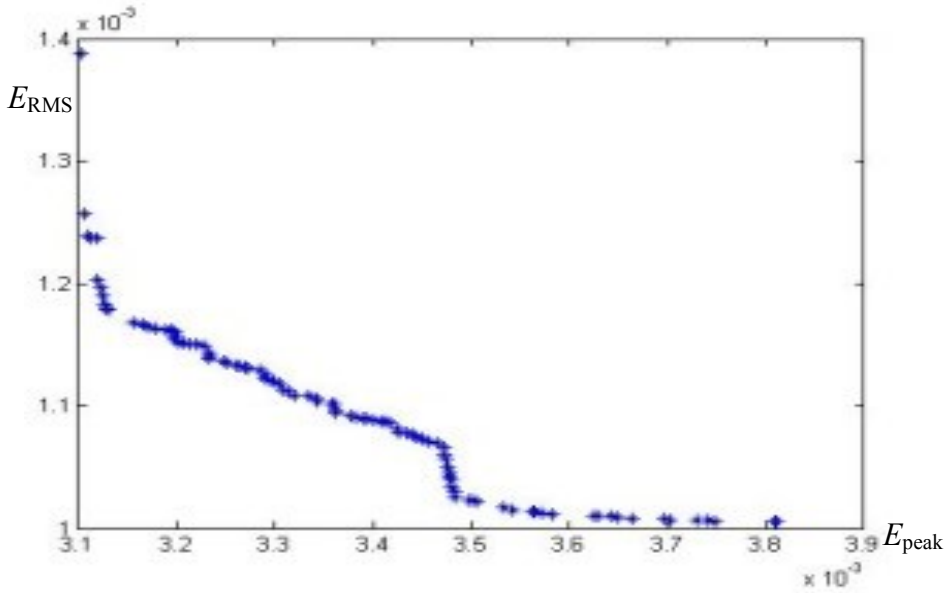

Figure 6: Pareto Front for NSGA-II after 1000 Generations

3.4 Summary

In this chapter, we described NSGA-II algorithm, and applied the algorithm to get the Pareto-front of the problem of lateral control of a tractor.

Chapter 4

MULTI-OBJECTIVE DIFFERENTIAL EVOLUTION

4.1 Differential Evolution

Differential evolution (DE) is an evolutionary optimization method that searches better solutions by random walk. Differential Evolution was introduced by Ken Price and Rainer Storm, as a simple evolutionary algorithm that generates new chromosome solutions by vector sum of three selected individuals of the population. As shown in Figure 7 the algorithm has a greedy-like selection of child only if its fitness is better compared to parents. DE often outperforms traditional evolutionary algorithms in searching best solutions if the genes consist of only scalars. A variety of DE operators was proposed for producing the next generation [14].

4.2 Multi-Objective Optimization Differential Evolution

The single objective DE algorithm cannot be applied directly for multi objective problems because the selection criteria by better fitness cannot work in multi objective case. In single objective, the offspring replace the parents when it is better. In multi-objective case, the solutions that dominate the others are the best solutions of the population, and therefore the offspring replace the parents if they dominate parents. This procedure is repeated until the number of created offspring reach to the population size [14].

4.2.1 Population

The initial population is started by random chromosomes. The fitness values of each chromosome of the population is computed and attached to the chromosomes. Both the chromosomes of the current population and new population are used to generate next population by the mutation and binomial crossover operators.

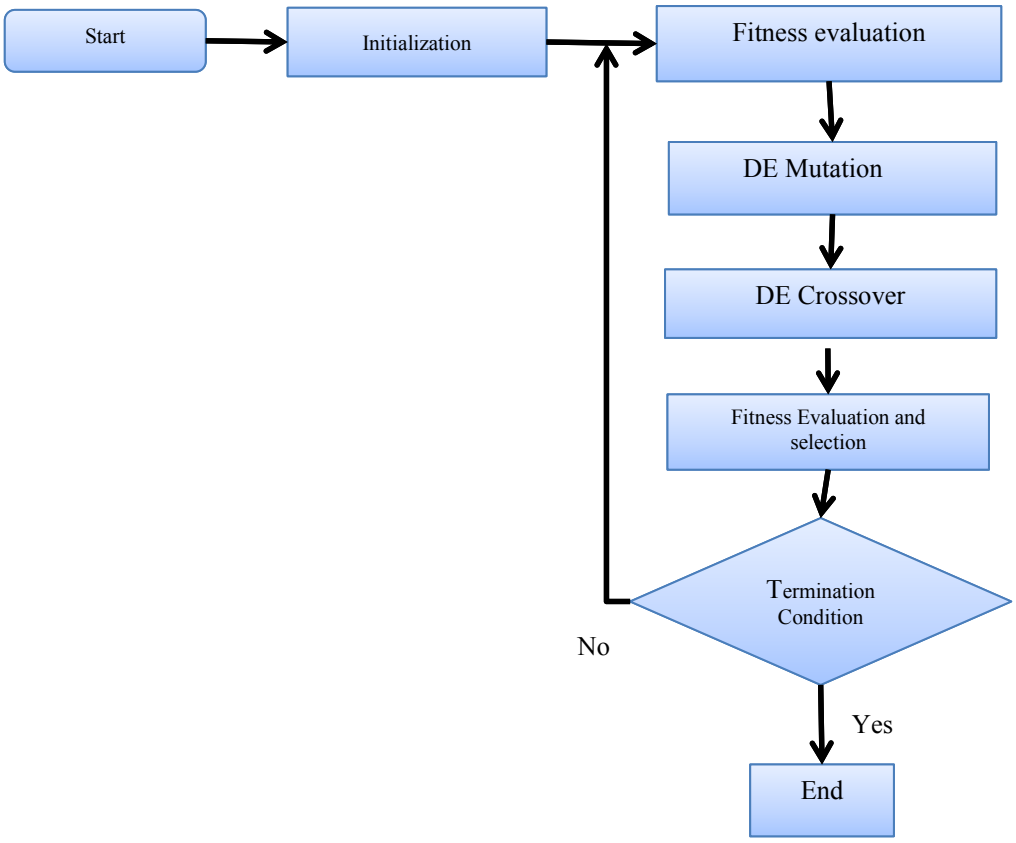

Figure 7: MODE Flowchart [7]

4.2.2 Rank and Crowding Distance Calculation

The individuals are compared for domination in objective functions, and their rank is assigned accordingly. Next, the Crowding-distance of the individuals are calculated and attached to the chromosomes [7].

After ranking and Crowding-distance assignment, the selected N_p individuals are processed by DE operation, which is a sequence of DE mutation, crossover and selection [7].

4.2.3 DE Mutation

The mutation operator selects randomly three individuals p_1, p_2, p_3 of the population as parent, and generates the new offspring o using the scale factor F_s [14].

$$o = p_{i1} + F_s\,(p_{i2} - p_{i3}) \tag{16}$$

4.2.4 DE Crossover

DE crossover operation is applied on all genes of the mutant chromosome in a random sequence with a crossover probability CR. For this purpose, for each gene, a random value η is chosen in (0, 1). If both CR$<\eta$ and the mutant chromosome has better rank than the parent, then mutant is taken as a child [14].

4.2.5 Selection

The elitist selection operator selects the chromosome for next generation by comparing the rank and Crowding-distance of the children and parents. The individuals with a lower rank and higher Crowding-distance are selected as the new parent. The lowest rank individuals form the set of Pareto-optimal solutions [7].

4.3 Application of MODE to search Pareto Front of Controller

In this chapter, the search of optimum control parameters of an automated farming vehicle planned to track a predetermined path on a loose soil surface. The motion of the auto steered tractor is modeled and the test procedure of the system to measure the lateral peak and RMS errors ıs explained in Chapter 2. MODE algorithm is suitable in searching the Pareto optimal solutions, and graphically represents the

corresponding Pareto-front to pick the best solution depending on the preference of the farming operator.

4.3.1 Structure of the Chromosomes

For MODE algorithm, each solution shall be expressed as a chromosome that consists of an array of genes. The four independent controller coefficients, K_D, K_2, L_1, and L_2 are the four scalar genes of a chromosome. Depending on the needs of the operators, some values such as the values of objective (fitness) functions $f_1=E_{\text{peak}}$ and $f_2=E_{\text{RMS}}$ are attached to each chromosome.

4.3.2 Initialization of the Population

The population is initialized by homogeneously distributed random numbers in interval (-5, 5). The selected crossover ratio and scale factor is CR=0.5 and F=0.5 as given in typical examples of similar problems.

Lateral peak and RMS errors for each chromosome of the population is evaluated to obtain the values of the objective functions. After evaluating their objective values, the population is sorted by non-dominated sorting procedure as described in Section 3.3.3 for NSGA-II. Table 2 shows the first ten chromosomes of the initial population and the values of two objective functions.

Table 2: Initialized Population and Evaluated Objective

K_D	K_2	L_1	L_2	E_{peak}	E_{RMS}
-1.4834	3.3083	0.8526	0.4972	0.0253	0.00334
4.1719	-2.1416	2.5720	2.5373	0.0252	0.00702
-1.1955	0.6782	-4.2415	-4.4615	0.0210	0.00515
0.3080	2.7917	4.3417	0.7722	0.0474	0.00947
1.2338	-0.6938	-0.1452	0.9178	0.0819	0.00482
-0.7409	-0.7096	0.9967	-0.6944	0.0235	0.00459
1.9266	0.5422	-2.2066	-0.1770	0.0373	0.00738
-0.9930	1.9960	-4.7404	0.8693	0.0305	0.00345
0.8929	-0.6535	-0.0351	-0.3587	0.0344	0.00804
-2.7102	4.3734	-3.4762	3.2582	0.0224	0.00330

4.4 Mutation

Three parents, p_{i1}, p_{i2} and pi_3, of population are selected for mutation operator and the mutant m is calculated by using the mutation scale factor F=0.5.

$$m = p_{i1} + F(p_{i2} - p_{i3})$$

$$m = \begin{pmatrix} -1.4834 \\ 3.3083 \\ 0.8526 \\ 0.4972 \end{pmatrix} + F\left[\begin{pmatrix} 4.1719 \\ -2.1416 \\ 2.5720 \\ 2.5373 \end{pmatrix} - \begin{pmatrix} -1.1955 \\ 0.6782 \\ -4.2415 \\ -4.4615 \end{pmatrix}\right]$$

4.5 Crossover

Crossover operator acts with probability CR=0.5, by generating a random number η in interval (0, 1), and comparing it to CR. If CR<η then the parent is selected as child. Otherwise, mutant is select as a child.

$$c = \begin{cases} p & \text{if } CR < \eta \\ m & \text{otherwise} \end{cases}$$

4.6 Selection

For the selection, the objective functions of both parent and child are evaluated to get their objective values (fitness). The elitist selection operation prefers the smaller rank of domination and larger Crowding-distance for the next generation and pareto-front.

Figure 8 illustrates the best non-dominated solution set in the pareto-front for the MODE problem after three hundred generation.

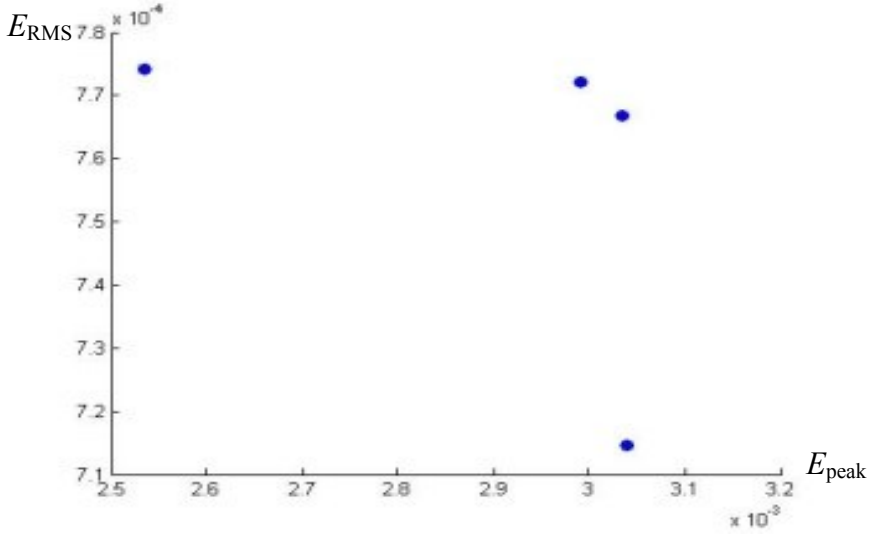

Figure 8: Pareto Front by MODE after 300 Generations

The Figure 9 illustrates the best non-domination solution set in the pareto-front for the MODE problem for one thousand generation.

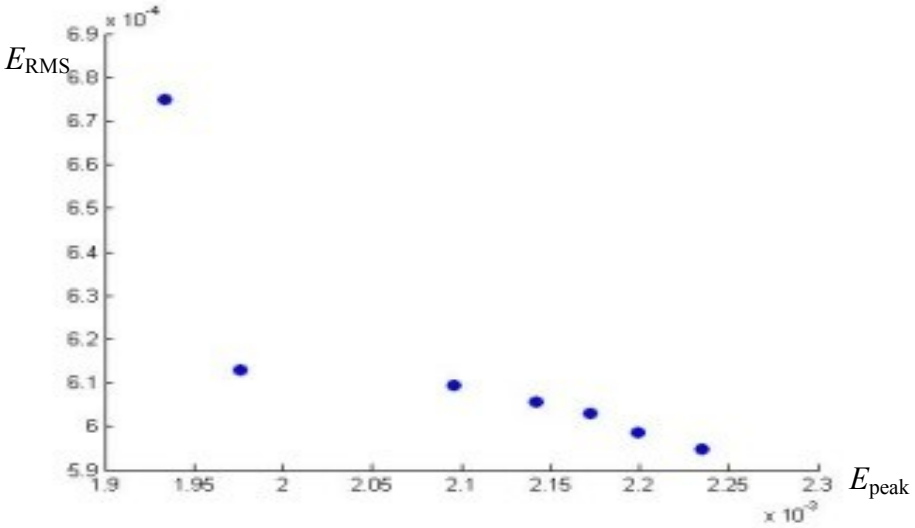

Figure 9: Pareto Front in MODE after 1000 iteration of Generations

4.7 Summary

In this chapter, we have given a simple example to explain MODE and show on this problem to create a small population to apply systematic mutation, crossover and selection and to create next generation. After more and more iteration, it converges to a set of non-dominated solution and starts to build up the Pareto front of the solution space.

Chapter 5

RESULTS AND DISCUSSIONS

5.1 Multi Objective Nature of Problem

The efficiency of the agricultural processes strongly depends on the terrain properties, surrounding environment conditions, and the path tracking accuracy of the automatic agricultural machines. The curvature transitions create common problems in path tracking control systems resulting in lateral tracking deviation. The lateral tracking error has two main components, the E_{peak}, and the root-mean-square E_{RMS}, along the path. The importance of the lateral error at the curvature transitions was addressed by Lenain et al. (2006) and the controller parameters of a double look-ahead reference point controller has been optimized to minimize both the E_{peak} and the E_{RMS} of an agricultural tractor manually by a multi stage steepest descent optimization algorithm.

NSGAII and MODE are multi-objective evolutionary optimization algorithms, which can find the pareto-optimal border of the solution space for a multi-objective problem. In this study, the controller parameters of an agricultural tractor are optimized using two multi objective optimization algorithms, NSGA-II and MODE [2].

Both NSGA-II and MODE requires mainly two structural parameters, the population size, and the number of generations to terminate the algorithm. The evolutionary

optimization algorithms used the fitness values, E_{peak} and E_{RMS}, which were obtained by the simulated runs of the tractor along a typical test-path. Tests were carried for a set of structural parameters and the followings are observed from the results of these test runs.

5.2 NSGA-II and MODE Settings

Those genes of the chromosome, K_D, K_2, L_1, and L_2 were bounded in interval (-5, 5) and the population size is tested at 300 and 1000. The crossover and mutation probability ratios in the NSGA-II for all cases are 0.8 for crossover and 0.2 for mutation. In the MODE algorithm, the crossover is 0.5 and scale factor value is 0.5. After a number of tests, we found that the best value for the DE scale factor is 0.5 and DE crossover ratio is 0.5, as it is given in examples.

In this group of runs, the populations of both algorithms were set to one hundred, and, the number of generations was set to three hundred. Both algorithms were started with random population interval K_D = (-5, 5), K_2 = (-5, 5), L_1 = (-5, 5) and L_2= (-5, 5). For NSGAII the execution time takes 35 min. For MODE, it takes 40 min to complete 300 generation.

5.3 Results with Population Size 100 after 300 and 1000 Generations

In Figure 10, the y-axis represents the E_{RMS} and the x-axis represents the E_{peak}. Pareto front of both NSGA-II and MODE are plotted in the same graph. The minimum E_{peak} and the minimum E_{RMS} obtained by NSGA-II are 0.0038m, and 0.0011m, respectively. For the MODE, the minimum E_{peak} is 0.0025m, and the minimum E_{RMS} comes out 0.00077m. The large difference in the minimum errors of these algorithms indicates that the population size and the number of generations shall be larger to get better results. The pareto-optimal solutions of MODE are listed in Table 3.

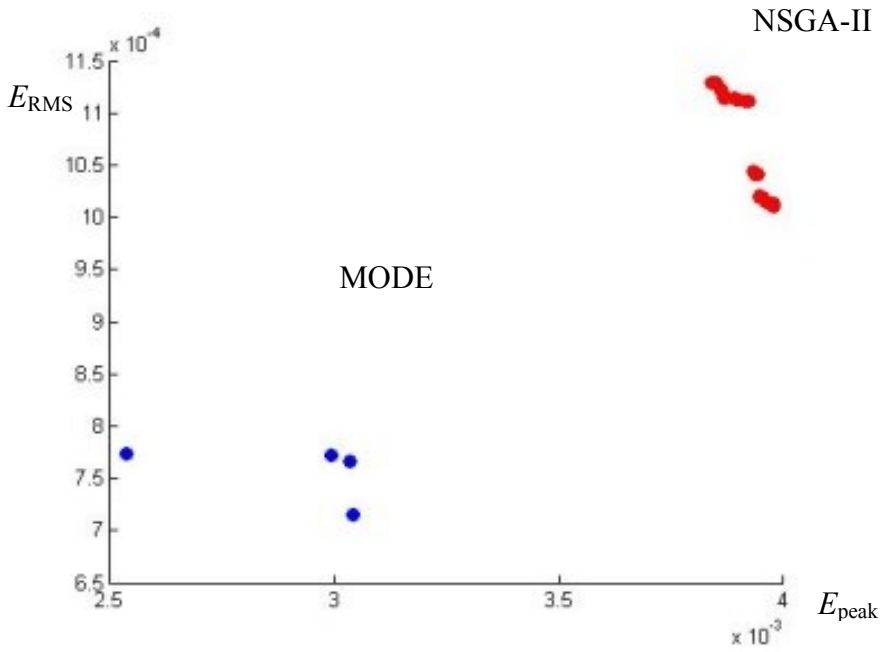

Figure 10: Pareto Front in NSGA-II and MODE after 300 Generations

Table 3: Controller Parameters for Pareto Optimal Points

Pareto Optimal Solution				Pareto Front	
K_D	K_2	L_1	L_2	E_{peak}	E_{RMS}
1.0981	2.16178	0.681764	-0.5935122	0.0029915	0.000772037
0.4169	2.31855	0.671671	-0.6032823	0.0025353	0.000774146
0.5624	2.29602	0.696577	-0.5679995	0.0030398	0.000714612
0.3833	1.93876	0.710725	-0.6242900	0.0030345	0.000766813

Figure 11 is obtained as a result of setting the populations of the algorithms both to one hundred and the number of generations to one thousand. The algorithms started with random population interval K_D =(-5, 5), K_2 =(-5, 5), L_1 =(-5, 5) and L_2 =(-5, 5).

In NSGAII, the crossover was 0.8 and mutation operation was 0.2. The implementation time was 3:30 min. For MODE, the crossover factor fixed 0.5, and the scaling factor fixed 0.5. The implementation time was 4 hour.

In Figure 11, the pareto-front of NSGA-II and MODE are shown on the same plot. The y-axis denotes the E_{RMS} and the x-axis denotes the E_{peak}. The minimum E_{peak} and, the minimum E_{RMS} achieve by NSGA-II are 0.0031m, and 0.0013m, respectively. From the MODE, the minimum E_{peak} is 0.0019m, and the minimum E_{RMS} comes out 0.00067. The large difference in the minimum error of these algorithms, indicate that the population size and the number of generation shall be larger to get better results.

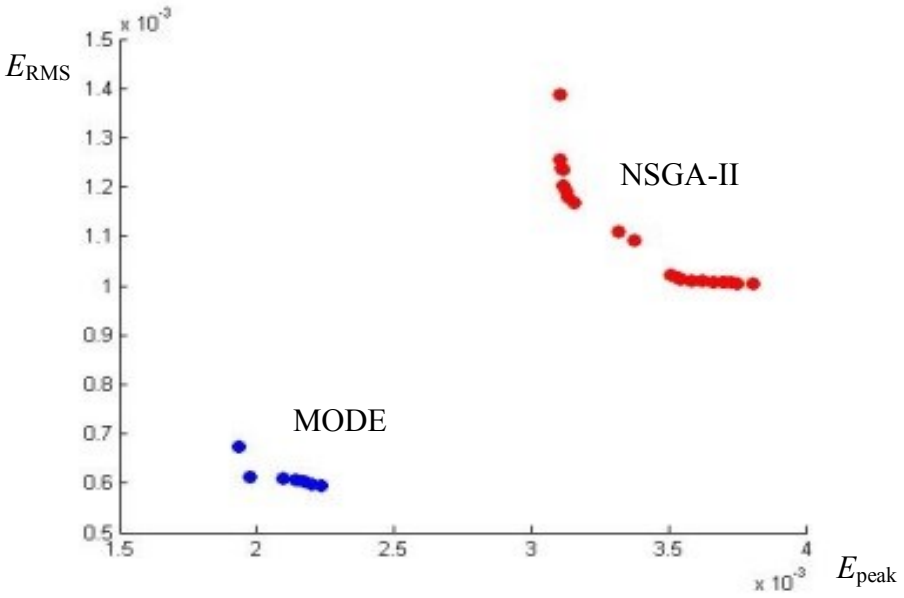

Figure 11: Pareto Front in NSGA-II and MODE after 1000 Generation

Table 4 shows the controller parameters for pareto-optimal points after 1000 generations.

Table 4: Controller Parameters for Pareto Optimal Points

Pareto Optimal Solution					Pareto Front	
#	K_D	K_2	L_1	L_2	E_{peak}	E_{RMS}
1	0.5544	3.9159	0.613971	-0.4458441	0.0019759	0.000613085
2	0.4561	3.8005	0.627220	-0.4444833	0.0020955	0.000609584
3	0.6897	3.7699	0.624200	-0.4395144	0.0022349	0.000594679
4	0.1473	3.4240	0.639277	-0.4690074	0.0019338	0.000674838
5	0.4965	3.9447	0.627356	-0.4307716	0.0021417	0.000605518

Figure 12 shows the resulting lateral deviation for a simulation along the desired reference path using the Pareto-optimal solution #3 in the left plot. The right side is the zoom to the maximum E_{peak} that occurred in the dashed small window. Figure 13 shows the lateral error d_N of the tractor along the reference path.

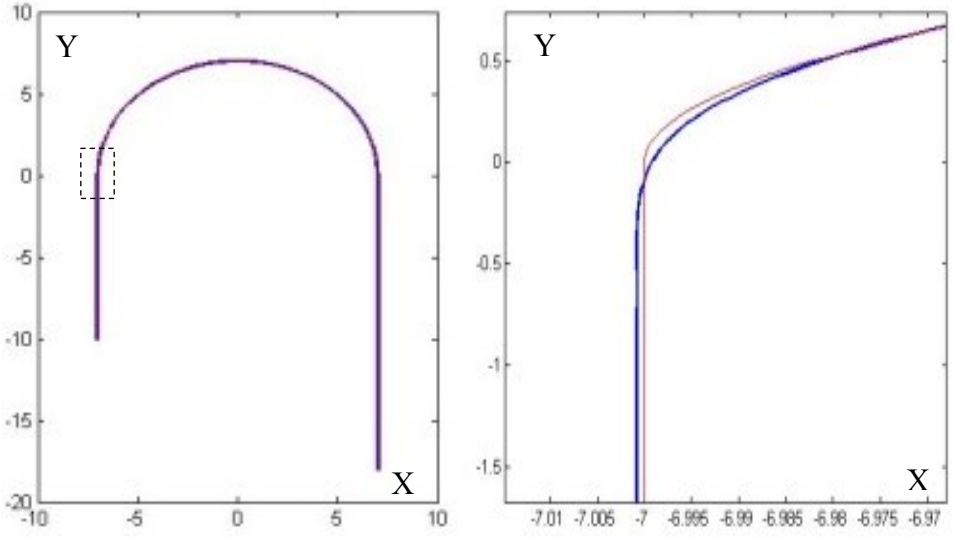

Figure 12: Motion of the Tractor Desire Reference Path

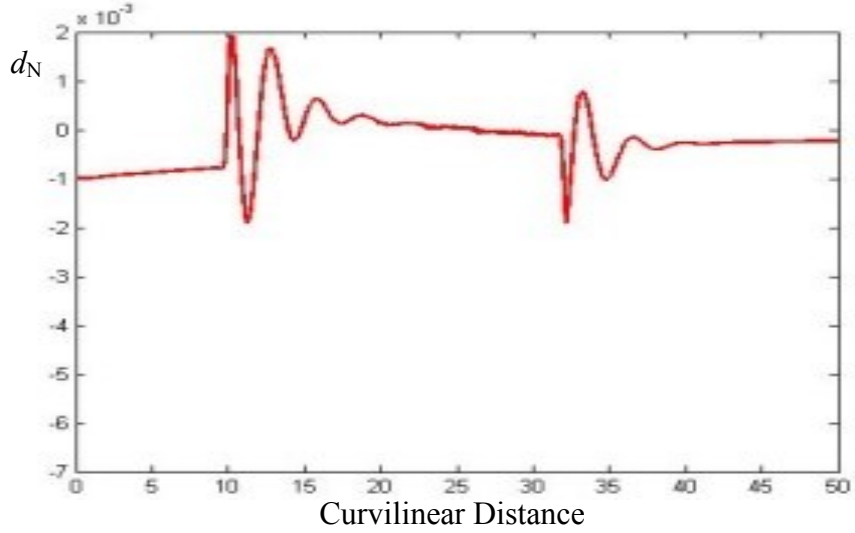

Figure 13: Lateral Error of the Tractor along the Reference Path

5.4 Search in a Narrower Solution Space

In this group of runs, the populations of both algorithms were set to one hundred, and, the number of generations was set to one thousand. Both algorithms were started with random population between K_D = (1, 5), K_2 = (1, 5), L_1 = (-1, 1) and L_2 = (-1,1).

In NSGA-II, the crossover was 0.8 and a mutation operation was 0.2 in searching the optimal solutions. The performance time takes 35 min. For MODE, the crossover factor is set to 0.5, the scaling factor is set to 0.5 in searching this solution, and it takes 40 min.

In Figure 14, the Pareto-front of NSGA-II and MODE are shown on the same plot. The y-axis represents the E_{RMS} and the x-axis represents the E_{peak}. The minimum E_{peak} and the minimum E_{RMS} obtained by NSGA-II are 0.0035m, and 0.00067m, respectively. For MODE, the minimum E_{peak} is 0.0021m, and the minimum E_{RMS} comes out 0.00054m. The large difference in the minimum errors of these algorithms

indicates that the population size and the number of generations shall be larger to get better results.

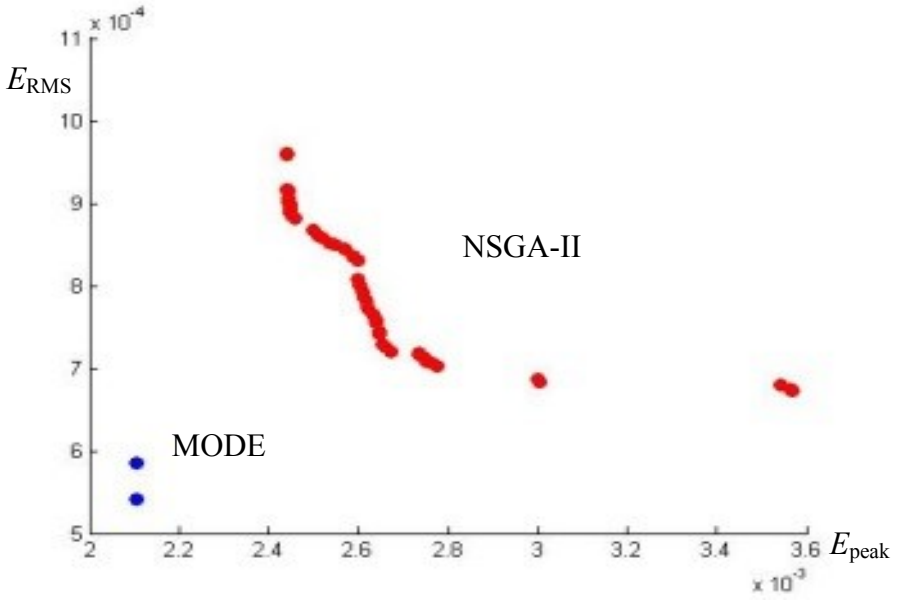

Figure 14: Pareto Front in NSGA-II and MODE with 300 Generations

Table 5: Controller Parameters for Pareto Optimal Points by MODE

Pareto Optimal Solution					Pareto Front	
#	K_D	K_2	L_1	L_2	E_{peak}	E_{RMS}
1	0.9826	5.1164	0.60129	-0.3776356	0.0021053	0.000541194
2	0.7321	4.2383	0.61262	-0.41567040	0.0021039	0.000585480

5.5 Results of Population size 100 with 1000 Generations

Figure 15 shows the results of both algorithms with the populations size100 and number of generations 1000 when genes of the chromosome are bounded in intervals $K_D = (1, 5)$, $K_2 = (1, 5)$, $L_1 = (-1, 1)$ and $L_2 = (-1, 1)$.

The crossover and mutation operation in NSGA-II were 0.8 and 0.2. The operation time was 3:30 min. For MODE, the crossover ratio was set to 0.5, and the scaling factor was set to 0.5. The operation time of MODE took four hours.

In Figure 15, the Pareto-front of NSGA-II and MODE are shown on the same plot. The y-axis denotes the E_{RMS} and the x-axis denotes the E_{peak}. The minimum E_{peak} and, the minimum E_{RMS} achieve by NSGA-II are 0.0019m, and 0.0007m, respectively. MODE delivered the minimum E_{peak} 0.0016m, and the minimum E_{RMS} 0.0004m. The large difference in the minimum error of these algorithms, indicate that the population size and the number of generation shall be larger to get better results.

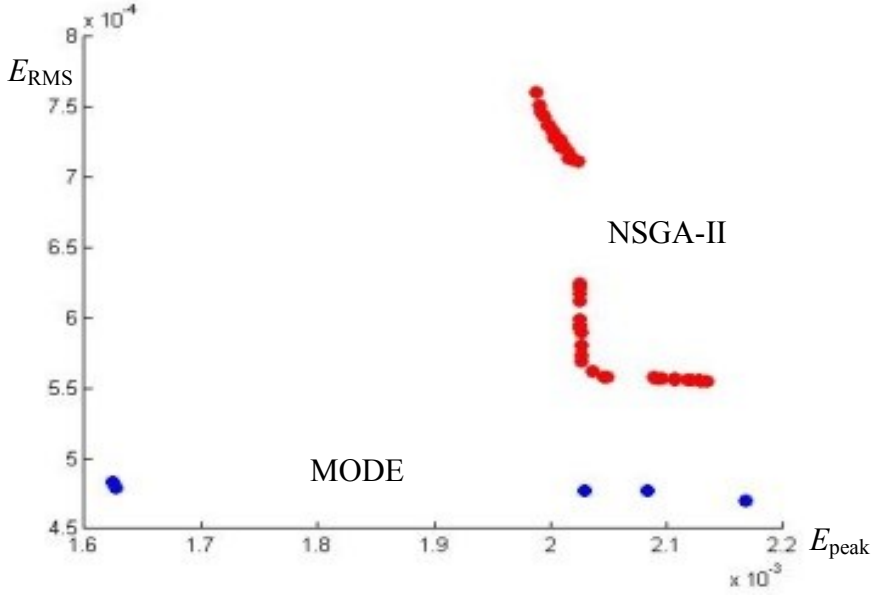

Figure 15: Pareto Front for MODE and NSGA-II for 1000 Generations

5.6 Effect of Pareto Optimal on the Lateral Error along the Path

The best Pareto-optimal solutions obtained from all search runs are shown in Table 6. The resulting lateral deviation for the Pareto-optimal solution #1 is demonstrated by a simulation along the desired reference path in the left plot of Figure 16. The plot at the right side zooms into the dashed small window to show the maximum E_{peak}.

Table 6: Controller Parameters for Pareto Optimal Points

Pareto Optimal Solution					Pareto Front	
#	K_D	K_2	L_1	L_2	E_{peak}	E_{RMS}
1	0.8648	7.4208	0.559544	-0.3081386	0.0016235	0.000483704
2	0.9018	7.3861	0.560413	-0.3091005	0.0016270	0.000478973
3	1.0092	6.5188	0.586364	-0.3208218	0.0020289	0.000477510
4	1.2401	6.5164	0.580384	-0.3141057	0.0020838	0.000477118
5	1.6543	6.6306	0.586108	-0.3204827	0.0021680	0.000470331

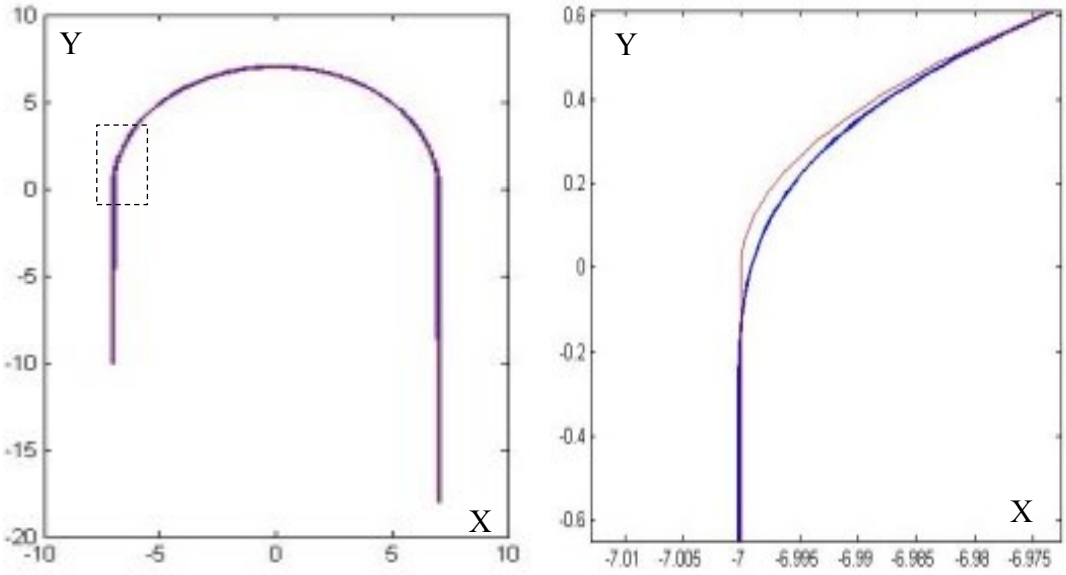

Figure 16: Motion of the Tractor and the Desired Reference Path

Figure 17 compares the effects of the Pareto optimal solutions on the lateral error of the tractor along the reference path. Smaller RMS error is a result of faster

convergence of the tractor motion to the linear or circular section of the path, and there is a compromise between minimum peak and minimum RMS errors. The red curve shows the error of the extreme Pareto optimal point with minimum peak error, which is given at the first row of Table 6. The blue curve belongs to the error with the minimum RMS error, which is given at the last row of the same table. It is clearly visible that red curve has higher error at the beginning of the 10^{th} meter compared to the blue curve. This error indicates the lower convergence rate of red curve compared to the blue curve. The tractor operator may select a lower peak error if the task is critical especially at the transients of the curvatures, or may select a controller setting that provides lower RMS error to improve the performance of tracking the linear sections of the path.

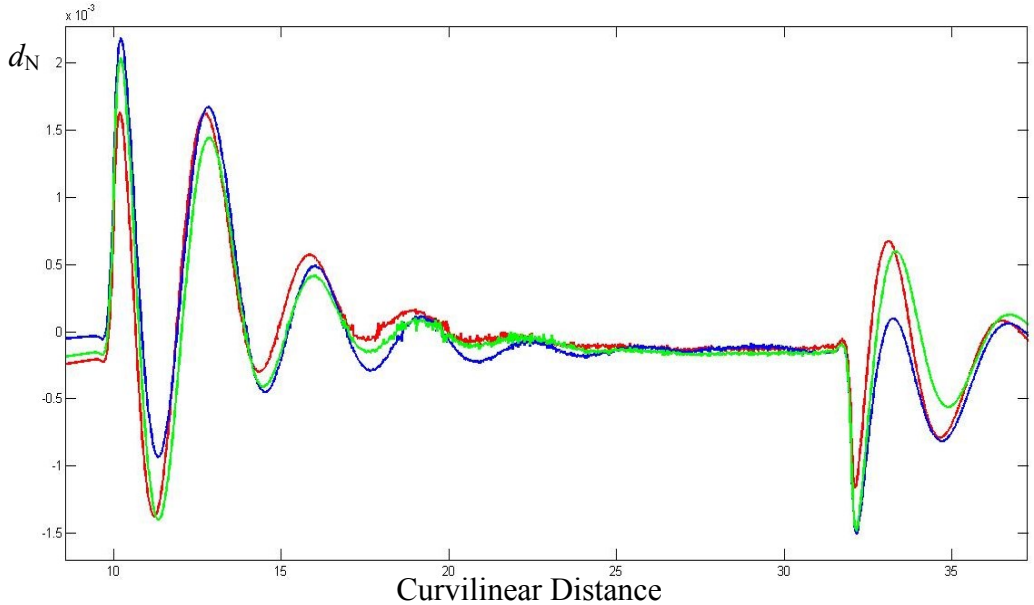

Figure 17: Lateral Error of the Tractor along the Reference Path for the Pareto-optimal solutions #1:Blue, #3:Green, #5:Red

5.7 Summary

This chapter illustrates the result of both algorithms: NSGA-II and MODE, which reduced the lateral errors E_{peak} and E_{RMS}. As it is seen in Table 6, the minimum E_{peak} and E_{RMS} in NSGA-II are 0.0019m, 0.0007m while the E_{RMS} and E_{peak} are 0.0016m, 0.0004m respectively in MODE algorithm.

Chapter 6

CONCLUSIONS

The aim of this study is to reduce the lateral error of the simulated tracking action of an agricultural tractor along a typical agricultural desired path. The most important features of the lateral error along a typical path are the E_{peak}, and E_{RMS}, which are independent components of the error along the path. A multi-objective optimization algorithm is required to search the best parameter settings to have both minimum E_{peak} and E_{RMS}. The best parameter settings form a non-dominated solution surface, which is described by a set of Pareto-front points.

This study applied NSGA-II, and MODE algorithms to determine the Pareto-front surface that compromise both peak and RMS errors. The lateral controller parameters of an actual tractor may be set to the controller parameters corresponding to the Pareto-front points depending on the importance of E_{RMS} or E_{peak} in the agricultural application.

The search gave better non-dominated solution surfaces with typical errors {E_{peak} =0.0016m, E_{RMS} =0.0004 m} compared to reported error {E_{peak} =0.0044m, E_{RMS} =0.0015m} in [4]. Using the results of this study, the tractor operator may select a lower peak error if the task is critical especially at the transients of the curvatures, or may select a controller setting that provides lower RMS error to improve the performance of tracking the linear sections of the path.

6.1 Future works

The real time application of the evolutionary optimization algorithms to enhance the lateral control parameters requires mainly development of these algorithms in a learning structure with a fast convergence rate. Development of such algorithms may be a sounding feature task for the real time enhancement of the control parameters.

REFERENCE

[1] M. Bodur, "Real-time population based optimization for adaptive motion control of robot manipulators," *Engineering Letters, vol. 16, pp. 27-35, Mar 2008*.

[2] M. Bodur, "An adaptive cross-entropy tuning of the pid control for robot manipulators"" *in Proceedings of the 2007 International Conference of Computational Intelligence and Intelligent Systems (ICCIIS-07), pp. 93-98, July 2007*

[3] M. Bodur and M. E. Sezer, "Adaptive control of flexible multilink manipulators," *International Journal of Control, vol. 58, no. 3, pp. 519-536, 1993*

[4] M. Bodur, E. Kiani. H. Hacisevki. "Double look-ahead reference point control for autonomous agricultural vehicles," *Biosystems Engineering, Volume 113, Issue 2, pages 173-186) ,* 2012.

[5] A Pratap, S Agarwal, and T. Meyarivan. Kalyanmoy Deb, "A fast and elitist multiobjective genetic algorithm: NSGA–II," in *IEEE Transactions on Evolutionary Computation*, 2002.

[6] T, Josef, "Differential evolution: competitive setting," in *Proceedings of the International Multiconference on*, 2006.

[7] Babu, B.V. and M.M. Jehan, 2003. Differential evolution for multi-objective optimization. *Proceedings of IEEE Congress on Evolutionary Computation, Dec. 8-12, IEEE Press, Canberra, Australia, pp: 2696-2703.*

[8] Yixin, Li Huiyuan.S, "An improved density estimation method in NSGA2," in *IET Conference Publications*, china, 2012.

[9] De, S. Bhattacharyya, S. Chakraborty, S. Sarkar, B.N. Prabhakar, P.K. Bose, S. "Gray scale image segmentation by nsga-ii based optimusic activation function," in *Communication Systems and Network Technologies (CSNT), International Conference on* , 2012.

[10] Carlos A. Coello Coello; Gary B. Lamont; David A. Van Veldhuisen (2007). Evolutionary algorithms for solving multi-objective problems. *Springer. ISBN 978-0-387-36797-2. Retrieved 1 November 2012*.

[11] F. Cordeiro and A. Silva-Filho, " NSGAII applied to unified second level cache memory hierarchy tuning aiming energy and performance optimization," in *Computing Systems (WSCAD-SCC), 2010 11th Symposium on*.

[12] A. Silva-Filho, C. Bastos-Filho, D. Falcao, F. Cordeiro and R. Castro, "An optimization mechanism intended for two-level cache hierarchy to improve energy and performance using the NSGAII algorithm," in *Computer Architecture and High Performance Computing SBAC-PAD '08. 20th International Symposium on* , 2008.

[13] U. K.Chakraborty, Advance in differantial evolution, mathematics &computer science , *University of Missouri Books, 2008.*

[14] S. Bechikh, N. Belgasmi, L. Ben Said and K. Ghedira, " A novel multi-objective memetic algorithm for continuous optimization," in *Tools with Artificial Intelligence,ICTAI '08. 20th IEEE International Conference on* , 2008.

[15] A. da Cruz, R. Cardoso, E. Wanner and R. Takahashi, "A multiobjective non-linear dynamic programming approach for optimal biological control in soy farming via NSGA-II," in *Evolutionary Computation, CEC 2007. IEEE Congress on*.

[16] S. Das and P. Suganthan, " Differential Evolution: A Survey of the State-of-the-Art," in *Evolutionary Computation, IEEE Transactions on (Volume:15 , Issue:1)* , 2011.

[17] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan "Pareto Optimization of Power System Reconstruction Using NSGA-II Algorithm," in *Power and Energy Engineering Conference (APPEEC), 2010 Asia-Pacific* .

[18] S. Bandaru, R. Tulshyan and K. Deb, "Modified SBX and adaptive mutation for real world single objective optimization," in *Evolutionary Computation (CEC), 2011 IEEE Congress on* , New Orleans, LA.

[19] T. Bhattacharjee and A. Chakraborty, "Congestion management in a deregulated

power system using NSGAII," in *Power India Conference, 2012 IEEE Fifth* .

APPENDIX

1 MatLab Code of Path Generator and Fitness Function

pathmaker.m

```
function pathmaker(x0,y0,dt,speed,firstlength,radius,lastlength)
%pathmaker(0,0,0.005,2,10,7,22)
clc
fh=fopen('pathdef.m','w');
fprintf(fh,'function path=pathdef()\r\n path=[ ');
ds=speed*dt; s=0;x=-radius; y=-firstlength; i=1; a=pi/2;
fprintf(fh,'%f  %f   %f   %f   %i\r\n', s, x, y, a, i );
while(s<firstlength)
   s=s+ds; x=-radius; y=y+ds; i=i+1; a=pi/2;
   fprintf(fh,'%f  %f   %f   %f   %i\r\n',  s, x, y, a, i );
end
while(s<firstlength+radius*pi)
   s=s+ds; sa=s-firstlength; a=pi/2-sa/radius;
   x=-radius*cos(sa/radius); y=radius*sin(sa/radius); i=i+1;
   fprintf(fh,'%f  %f   %f   %f   %i\r\n',  s, x, y, a, i);
end
while(s<firstlength+radius*pi+lastlength)
   s=s+ds; x=radius; y=y-ds; i=i+1; a=-pi/2;
   fprintf(fh,'%f  %f   %f   %f   %i\r\n',  s, x, y, a, i);
end
   fprintf(fh,']; \r\n' );

fclose(fh);
end

%-----end of file
```

fitness2.m

```
function [peakd, rmsd]=fitness2(kN,kD,k1,k2,L1,L2)
%[a,b]=fitness(0,3,0,4,-0.8,0.8)
% Path and Plot
path=pathdef; len_path=size(path,1); startstep=0.001;
plotgrA=0; plotgrB=0; plotgrM=0; % no plot for evol.search
%plotgrA=1; plotgrB=0; plotgrM=1;% plot to get path and dN
% Vehicle Coefficients
 x=1; y=2; w=3;       % Positional terms
% Steering parameter
 s=0; b=0; v=2.0; smax=0.5585; smin=-smax;
% Path parameters L2>L1
 R=abs(path(1,2));
% Bevly & Derrick (2008) coefficients
m=11340; I=18500; Lf=1; Lr=2.0; L=Lf+Lr;
Cr=286479; Cf=137510; Fr=0;
dt=0.01; dtc=0.05; dts=0.001;
ds=path(2,1); equit=0.3; cquit=0;
ibest=1; vbest=[]; iter=0; tcalc=0;

LrmseS=1050;LrmseE=1500; LpeakS=100; LpeakE=2100;
Lcancel=2400; Lpoff=900; % for circular and overall test
% Controller Coefficients
%kN=0; kD=3; k1=0; k2= 4.7;L1=-0.7;L2= 0.73;
KNLine=5.6; KLCirc=2.28;
k1=(KLCirc-k2*L2)/L1; kN=KNLine-k1-k2;

%--initialization----------------------------------------
t=0; j=0; tf=25;
ts=t; s1=0; s2=0; s3=0; % actuator time and states
tc=t+dt; % controller time
% Initial position and heading pv=[R -Lr+25 -pi/2];
  pv=[-R-startstep Lr-10 pi/2];
vv=[0  v   0];    % Initial velocities
av=[0  0   0];    % Initial acceleration
xr=pv(x)-Lr*cos(pv(w)); yr=pv(y)-Lr*sin(pv(w));
clear ov; % observation vector is cleared
idN=2; dsign=1;
while( t<tf)
    t=t+dt; j=j+1; %time & iteration
    %simplification of program notation
    Cw= cos(pv(w)); Sw= sin(pv(w)); Cs= cos(s); Cb= cos(b);
    % CoG (x,y) --> (xr,yr) rear wheels point
    xrp=xr; yrp=yr; xr=pv(x) - Lr*Cw; yr=pv(y) - Lr*Sw;
    if(t==dt), xrp=xr; yrp=yr; end % debug
```

```
    if tc>=t, tc=tc+dtc; % control part
%   control starts with finding pN
    % inputs idN, xr, yr,thetar, path  outputs idN, dN, d
    idNs=idN; idNend=idN+200;
    xk=path(idN-1,2); yk= path(idN-1,3);
    ddNp= (xk-xr)*(xk-xr)+(yk-yr)*(yk-yr); ddNpp=ddNp;
    for kdN=idNs:idNend,
        xk=path(kdN,2); yk= path(kdN,3);
        ddN=(xk-xr)*(xk-xr)+(yk-yr)*(yk-yr);
        idN=max(kdN-1,2);
        if(ddNp<ddN), break; end
        ddNpp=ddNp; ddNp=ddN;
    end
    dNbest=sqrt(ddNp); dNfw=sqrt(ddN); dNbw=sqrt(ddNpp);
    if(dNfw<dNbw), dNnext=dNfw; else dNnext=dNbw; end
    dN=nearestdist(dNbest,dNnext,ds); dN=abs(dN);
    thetaP=path(idN,4); thetaN=thetaP-pv(w);
    thetaP1=path(min(max(idN+round(L1/ds),1),len_path),4);
    theta1=thetaP1-pv(w);
    thetaP2=path(min(max(idN+round(L2/ds),1),len_path),4);
    theta2=thetaP2-pv(w);
    ca=path(idN,1); xn=path(idN,2); yn=path(idN,3);
    % determine sign of distance
    xNd=-xn+path(idN+1,2); yNd=-yn+path(idN+1,3);
    dsign= sign(xNd*(-yr+yn)-yNd*(-xr+xn));
    d= dsign*abs(dN);

    % Control Law for steer angle
    sd=kD*d + kN*thetaN + k1*theta1 + k2*theta2;
   end
  %--control part is over, vehicle simulation starts here---------

    % Quit the case if error exceeds a threshold
     eer=abs(d);  if eer>equit, cquit=1; t=tf;  end

    % Servo Actuator
    while(ts<t), ts=ts+dts;
       s=sd*0.000297 + 2.894*s1 - 2.858997*s2 +0.9647*s3 ;
       s3=s2; s2=s1;s1=s; end
    s=max(min(s,smax),smin);

  % Vehicle equations of motion
    % Friction force is normal to tires
    if s>=0, Ff= -Cf*(s-b- vv(w)*Lf*Cb/v);
    else Ff= Cf*(b + vv(w)*Lf*Cb/v-s); end
    Fc=m*(v*cos(b))^2*tan(s)*sign(b)/L; Fwf=27e3*sin(s);
    ddy=(-Fr-Ff*Cs-Fc+Fwf)/m+v*vv(w);
    av(w)= (-Ff*Cs*Lf + Fr*Lr+Fwf*Lf)/I;
    av(x)=-ddy*Sw; av(y)= ddy*Cw;
    % Absolute Velocities
    vv(x)=vv(x)+av(x)*dt;vv(y)= vv(y)+av(y)*dt; vv(w)=vv(w)+av(w)*dt;
    % Constant Forward Velocity
    vyy=-vv(x)*Sw+vv(y)*Cw; % Lateral y velocity
    vv(x)=-vyy*Sw + v*Cw; vv(y)= vyy*Cw + v*Sw;
    % Side slip angle
    b= atan2(vv(y),vv(x))-pv(w);
    Fr= Cr*(b - vv(w)*Lr*Cb/v); % Friction force is normal to tires
    % Vehicle positions and orientation
    pv(x)=pv(x) + vv(x)*dt;
    pv(y)=pv(y) + vv(y)*dt;
    pv(w)=pv(w) + vv(w)*dt;
    %=================================================================
    %        1 2 3  4  5  6  7   8   9     10    11 Observation Vector
    %        t s Xn Yn d  xr yr  dN thetaN dsign ca
    ov(j,:)=[t s xn yn d  xr yr  dN thetaN dsign ca ];
end % case run completed

 L_ov=size(ov,1);
 if cquit, peakd=equit; rmsd=equit;
 else
  [peaka,ipeak]=max(ov(100:2100,8));
  peakd = abs(ov(ipeak+100,5));
  rmsd  = sqrt(ov(100:1500,5)'*ov(100:1500,5)/(1500-100) ) ;
  poff=ov(900,5);
  dispvec=[ L1 L2 k1 k2 peakd*1000 rmsd*1000 poff*1000 ];
   % plot the testdrive
  if plotgrA,
```

```
122      figure(2);
123      plot(ov(:,6),ov(:,7),'LineWidth',1.5);hold on;
124      plot(ov(:,3),ov(:,4),'r','LineWidth',0.5); end
125    if plotgrM,
126      figure(1);
127      plot(ov(:,11),ov(:,5),'r','LineWidth',1.5); end
128   end
129  return
130
131  function d=nearestdist(d1,d2,d0)
132  e=(d1*d1-d2*d2-d0*d0)/(2*d0); d=sqrt(abs(d2*d2-e*e)); a=-e;
133  return
134
```

2 MatLab Code of MODE

```
clear all;
close all;
clc;
MODEDat.NOBJ = 2;
MODEDat.NRES = 0;
MODEDat.NVAR = 4;
MODEDat.FieldD =[zeros(MODEDat.NVAR,1)...
nes(MODEDat.NVAR,1)];
MODEDat.Initial=[zeros(MODEDat.NVAR,1)...
ones(MODEDat.NVAR,1)];
MODEDat.XPOP= 50*MODEDat.NOBJ;
MODEDat.Esc = 0.5;
MODEDat.Pm  = 0.5;
MODEDat.InitialPop=[];
MODEDat.MAXGEN =300;
MODEDat.MAXFUNEVALS = 15000000*MODEDat.NVAR...
*MODEDat.NOBJ;
MODEDat.SaveResults='yes';
MODEDat.CounterGEN=0
MODEDat.CounterFES=0
OUT=MODE(MODEDat);
function OUT=MODE(MODEDat)
%% Reading parameters from MODEDat
Generaciones  = MODEDat.MAXGEN;    % Maximum number of generations.
Xpop          = MODEDat.XPOP;
Nvar          = MODEDat.NVAR;
Nobj          = MODEDat.NOBJ;
Bounds        = MODEDat.FieldD;
Initial       = MODEDat.Initial;
ScalingFactor = MODEDat.Esc;
CrossOverP    = MODEDat.Pm;
% mop           = MODEDat.mop;

%% Initial random population
Parent = zeros(Xpop,Nvar);
Mutant = zeros(Xpop,Nvar);
Child  = zeros(Xpop,Nvar);
FES    = 0;
for xpop=1:Xpop
Parent(xpop,1)=-1+(1+1).*rand;
Parent(xpop,2)=-1+(1+1).*rand;
Parent(xpop,3)=-1+(1+1).*rand;
Parent(xpop,4)=-1+(1+1).*rand;

end;
Initial(nvar,1)+(Initial(nvar,2)...

if size(MODEDat.InitialPop,1)>=1
Parent(1:size(MODEDat.InitialPop,1),:)=MODEDat.InitialPop;
end
JxParent1=zeros(Xpop,1);
JxParent2=zeros(Xpop,1);
for L=1:Xpop
[A,B] = fitness2(Parent(L,1),Parent(L,2),Parent(L,3),Parent(L,4) ); %
parents cost value
JxParent1(L)=A;
JxParent2(L)=B;
end
FES = FES+Xpop;
%% Evolution process
for n=1:Generaciones
for xpop=1:Xpop
rev=randperm(Xpop);
%% Mutant vector calculation
Mutant(xpop,:)= Parent(rev(1,1),:)+ScalingFactor*...
(Parent(rev(1,2),:)- Parent(rev(1,3),:));
%% Crossover operator
for nvar=1:Nvar
  if rand() > CrossOverP
Child(xpop,nvar) = Parent(xpop,nvar);
  else
Child(xpop,nvar) = Mutant(xpop,nvar);
    End
   end
```

```
end
JxChild1=zeros(Xpop,1);
JxChild2=zeros(Xpop,1);
for L=1:Xpop
[A,B] = fitness2(Child(L,1),Child(L,2),Child(L,3),Child(L,4)); %
parents cost value
JxChild1(L)=A;
JxChild2(L)=B;
end

FES=FES+Xpop;

%% Selection
 for xpop=1:Xpop
   if JxChild1(xpop,:) <= JxParent1(xpop,:) ...
       &&  JxChild2(xpop,:) <= JxParent2(xpop,:)
   Parent(xpop,:) = Child(xpop,:);
   JxParent1(xpop,:) = JxChild1(xpop,:);
   JxParent2(xpop,:) = JxChild2(xpop,:);
   end
    End
PFront(:,1)=JxParent1;
PFront(:,2)=JxParent2;
  PSet=Parent;
   % combined two vector value into one
 JxParent(:,1)= JxParent1;
JxParent(:,2)= JxParent2;
OUT.Xpop           = Parent;   % Population
OUT.Jpop           = JxParent; % Poopulation's Objective Vector
OUT.PSet           = PSet;     % Pareto Set
OUT.PFront         = PFront;   % Pareto Front
OUT.Param          = MODEDat;  % MODE Parameters
MODEDat.CounterGEN = n;
MODEDat.CounterFES = FES;

%      [OUT MODEDat]=PrinterDisplay(OUT,MODEDat); % To print results on
screen

if FES>MODEDat.MAXFUNEVALS || n>MODEDat.MAXGEN
  disp('Termination criteria reached.')
  break;
 end
 end

OUT.Xpop=PSet;
OUT.Jpop=PFront;
[OUT.PFront, OUT.PSet]=DominanceFilter(PFront,PSet); %A Dominance
Filter

f strcmp(MODEDat.SaveResults,'yes')
save(['OUT_' datestr(now,30)],'OUT'); %Results are saved
end

disp('++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++')
disp('Red  asterisks : Set Calculated.')
disp('Black diamonds : Filtered Set.')
if strcmp(MODEDat.SaveResults,'yes')
disp(['Check out OUT_' datestr(now,30) ...
variable on folder for results.'])
end
disp('++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++')

F=OUT.PFront;
for xpop=1:size(OUT.PFront,1)
if Nobj==1
figure(123); hold on;
plot(MODEDat.CounterGEN,log(min(F(:,1))),'dk',
'MarkerFaceColor','k'); grid on; hold on;
elseif Nobj==2
figure(123); hold on
plot(F(xpop,1),F(xpop,2),'dk','MarkerFaceColor','k');...
grid on; hold on;
elseif Nobj==3
figure(123); hold on;
plot3(F(xpop,1),F(xpop,2),F(xpop,3),'dk','MarkerFaceColor','k
```

```
grid on; hold on;
end
end
%
function [OUT Dat]=PrinterDisplay(OUT,Dat)

disp('-----------------------------------------------')
disp(['Generation: ' num2str(Dat.CounterGEN)]);
disp(['FEs: ' num2str(Dat.CounterFES)])
disp(['Pareto Front Size: ' mat2str(size(OUT.PFront,1))]);
disp('-----------------------------------------------')

if mod(Dat.CounterGEN,1)==0
if Dat.NOBJ==3
figure(123);
plot3(OUT.PFront(:,1),OUT.PFront(:,2),OUT.PFront(:,3),'*r');
grid on;
elseif Dat.NOBJ==2
figure(123);
plot(OUT.PFront(:,1),OUT.PFront(:,2),'*r'); grid on;
elseif Dat.NOBJ==1
figure(123);
plot(Dat.CounterGEN,log(min(OUT.PFront(:,1))),'*r'); ...
grid on; hold on;
end
end

function [Frente Conjunto]=DominanceFilter(F,C)
Xpop=size(F,1);
Nobj=size(F,2);
Nvar=size(C,2);
Frente=zeros(Xpop,Nobj);
Conjunto=zeros(Xpop,Nvar);
k=0;
for xpop=1:Xpop
Dominado=0;
for compara=1:Xpop
if F(xpop,:)==F(compara,:)
if xpop > compara
Dominado=1;
break;
end
else
if F(xpop,:)>=F(compara,:)
Dominado=1;
break;
end
end
end
if Dominado==0
k=k+1;
Frente(k,:)=F(xpop,:);
Conjunto(k,:)=C(xpop,:);
end
end
Frente=Frente(1:k,:);
Conjunto=Conjunto(1:k,:);

```

3 MatLab Code of NSGA-II

```
pop=100;
gen=1000;
if isnumeric(pop) == 0 || isnumeric(gen) == 0
error('Both input arguments pop and gen should be integer datatype');
end
% Minimum population size has to be 20 individuals
if pop < 20
error('Minimum population for running this function is 20');
end
if gen < 5
error('Minimum number of generations is 5');
end
% Make sure pop and gen are integers
pop = round(pop);
gen = round(gen);
function [number_of_objectives, number_of_decision_variables,
min_range_of_decesion_variable, max_range_of_decesion_variable] =
objective_description_function()
g = sprintf('Input the number of objective: ');
% Obtain the number of objective function
number_of_objectives = input(g);
if number_of_objectives < 2
error('This is a multi-objective optimization function hence the
minimum number of objectives is two');
end
g = sprintf('\nInput the number of decision variables: ');
% Obtain the number of decision variables
number_of_decision_variables = input(g);
clc
for i = 1 : number_of_decision_variables
clc
g = sprintf('\nInput the minimum value for decision variable %d : ',
i);
% Obtain the minimum possible value for each decision variable
min_range_of_decesion_variable(i) = input(g);
g = sprintf('\nInput the maximum value for decision variable %d : ',
i);
% Obtain the maximum possible value for each decision variable
max_range_of_decesion_variable(i) = input(g);
clc
end
g = sprintf('\n Now edit the function named "evaluate_objective"
x = input(g, 's');
if isempty(x)
x = 'x';
end
while x ~= 'c'
clc
x = input(g, 's');
if isempty(x) x = 'x';
end
end
function f = initialize_variables(N, M, V, min_range, max_range)
min = min_range;
max = max_range;
for j = 1 : V
f(i,j) = min(j) + (max(j) - min(j))*rand(1);
end
peakd=0;
rmsd=0;
[peakd, rmsd]=fitness2(f(i,1),f(i,2),f(i,3),f(i,4));
f(i,V + 1: K) =[peakd, rmsd];
end
function f = initialize_variables(N, M, V, min_range, max_range)
function f = non_domination_sort_mod(x, M, V)
[N, m] = size(x);
clear m
function f  = genetic_operator(parent_chromosome, M, V, mu, mum,
l_limit, u_limit)
[N,m] = size(parent_chromosome);
clear m
p = 1;
was_crossover = 0;
was_mutation = 0;
```

```
for i = 1 : N
% With 90 % probability perform crossover
if rand(1) < 0.9
     child_1 = [];
     child_2 = [];
      parent_1 = round(N*rand(1));
    if parent_1 < 1
       parent_1 = 1;
    End
    % Select the second parent
       parent_2 = round(N*rand(1));
    if parent_2 < 1
     parent_2 = 1
    End
      while
isequal(parent_chromosome(parent_1,:),parent_chromosome(parent_2,:))
    parent_2 = round(N*rand(1));
   if parent_2 < 1
  parent_2 = 1;
    End
   End
     parent_1 = parent_chromosome(parent_1,:);
     parent_2 = parent_chromosome(parent_2,:)

   for j = 1 :
      u(j) = rand(1);
   if u(j) <= 0.5
      bq(j) = (2*u(j))^(1/(mu+1));
else
   bq(j) = (1/(2*(1 - u(j))))^(1/(mu+1));
 End
% Generate the jth element of first child
child_1(j) = ...
0.5*(((1 + bq(j))*parent_1(j)) + (1 - bq(j))*parent_2(j));
% Generate the jth element of second child
child_2(j) = ...
0.5*(((1 - bq(j))*parent_1(j)) + (1 + bq(j))*parent_2(j));
if child_1(j) > u_limit(j)
child_1(j) = u_limit(j);
elseif child_1(j) < l_limit(j)
child_1(j) = l_limit(j);
end
if child_2(j) > u_limit(j)
child_2(j) = u_limit(j);
elseif child_2(j) < l_limit(j)
child_2(j) = l_limit(j)
end
end
peakd=0;
rmsd=0;
[peakd,rmsd]=fitness2(child_1(:,1),child_1(:,2),child_1(:,3),child_1(:,
4));
child_1(:,V + 1: M + V) = [peakd, rmsd];
peakd=0;
rmsd=0;
[peakd,rmsd]=fitness2(child_2(:,1),child_2(:,2),child_2(:,3),child_2(:,
4));
child_2(:,V + 1: M + V) =  [peakd, rmsd];
was_crossover = 1;
was_mutation = 0;
Else
% Select at random the parent
parent_3 = round(N*rand(1));
if parent_3 < 1
parent_3 = 1;
end
child_3 = parent_chromosome(parent_3,:);
for j = 1 : V
r(j) = rand(1);
if r(j) < 0.5
delta(j) = (2*r(j))^(1/(mum+1)) - 1;
else
delta(j) = 1 - (2*(1 - r(j)))^(1/(mum+1));
end
% Generate the corresponding child element.
child_3(j) = child_3(j) + delta(j);
% space.
```

```
143 if child_3(j) > u_limit(j)
144 child_3(j) = u_limit(j);
145 elseif child_3(j) < l_limit(j)
146 child_3(j) = l_limit(j);
147 end
148 end
149 peakd=0;
150 rmsd=0;
151 [peakd,rmsd]=fitness2(child_3(:,1),child_3(:,2),child_3(:,3),c
    hild_3(:,4));
152 child_3(:,V + 1: M + V) = [peakd, rmsd];
153 % Set the mutation flag
154 was_mutation = 1;
155 was_crossover = 0;
156 end
157 if was_crossover
158 child(p,:) = child_1;
159 child(p+1,:) = child_2;
160 was_cossover = 0;
161 p = p + 2;
162 elseif was_mutation
163 child(p,:) = child_3(1,1 : M + V);
164 was_mutation = 0;
165 p = p + 1;
166 end
167 end
168 f = child;
169 front = 1;
170 F(front).f = [];
171 individual = [];
172 function f = tournament_selection(chromosome, pool_size, tour_size)
173 [pop, variables] = size(chromosome);
174 rank = variables - 1;
175 distance = variables;
176 for i = 1 : pool_size
177 % Select n individuals at random, where n = tour_size
178 for j = 1 : tour_size
179 % Select an individual at random
180 candidate(j) = round(pop*rand(1));
181 % Make sure that the array starts from one.
182 if candidate(j) = =
183 candidate(j) = 1;
184 End
185 if j > 1
186 % Make sure that same candidate is not choosen.
187 while ~isempty(find(candidate(1 : j - 1) == candidate(j)))
188 candidate(j) = round(pop*rand(1));
189 if candidate(j) == vc0
190 candidate(j) = 1
191 end
192 end
193 end
194 end
195 for j = 1 : tour_size
196 c_obj_rank(j) = chromosome(candidate(j),rank);
197 c_obj_distance(j) = chromosome(candidate(j),distance);
198 end
199 % Find the candidate with the least rank
200 min_candidate = ...
201 find(c_obj_rank == min(c_obj_rank));
202 if length(min_candidate) ~= 1
203 max_candidate = ...
204 find(c_obj_distance(min_candidate)==max(c_obj_distance(min_candidate)))
    ;
205 if length(max_candidate) ~= 1
206 max_candidate = max_candidate(1);
207 end
208 (i,:) = chromosome(candidate(min_candidate(max_candidate)),:);
209 Else
210 % Add the selected individual to the mating pool
211 f(i,:) = chromosome(candidate(min_candidate(1)),:);
212 end
213 end
214 for i = 1 : N
215 individual(i).n = 0;
216 individual(i).p = [];
217 for j = 1 : N
```

```
dom_less = 0;
dom_equal = 0;
dom_more = 0;
for k = 1 : M
if (x(i,V + k) < x(j,V + k))
dom_less = dom_less + 1;
elseif (x(i,V + k) == x(j,V + k))
dom_equal = dom_equal + 1;
else
dom_more = dom_more + 1;
end
end
if dom_less == 0 && dom_equal ~= M
individual(i).n = individual(i).n + 1;
elseif dom_more == 0 && dom_equal ~= M
individual(i).p = [individual(i).p j]
end
end
if individual(i).n ==
x(i,M + V + 1) = 1;
F(front).f = [F(front).f i];
end
end
% Find the subsequent fronts
while ~isempty(F(front).f)
Q = [];
for i = 1 : length(F(front).f)
if ~isempty(individual(F(front).f(i)).p)
for j = 1 : length(individual(F(front).f(i)).p)
individual(individual(F(front).f(i)).p(j)).n
individual(individual(F(front).f(i)).p(j)).n - 1;
if individual(individual(F(front).f(i)).p(j)).n ==
x(individual(F(front).f(i)).p(j),M + V + 1) = ...
front + 1;
Q = [Q individual(F(front).f(i)).p(j)];
End
End
end
end
front =  front + 1;
F(front).f = Q;
end
[temp,index_of_fronts] = sort(x(:,M + V + 1));
for i = 1 : length(index_of_fronts)
sorted_based_on_front(i,:) = x(index_of_fronts(i),:);
end
current_index = 0;
for front = 1 : (length(F) - 1)
%    objective = [];
distance = 0;
y = [];
previous_index = current_index + 1;
for i = 1 : length(F(front).f)
y(i,:) = sorted_based_on_front(current_index + i,:);
end
current_index = current_index + i;
% Sort each individual based on the objective
sorted_based_on_objective = [];
for i = 1 : M
[sorted_based_on_objective, index_of_objectives] = ...
sort(y(:,V + i));
sorted_based_on_objective = [];
for j = 1 : length(index_of_objectives)
sorted_based_on_objective(j,:) = y(index_of_objectives(j),:);
end
f_max = ...
sorted_based_on_objective(length(index_of_objectives), V + i
f_min = sorted_based_on_objective(1, V + i);
y(index_of_objectives(length(index_of_objectives)),M + V + 1 + i
 Inf;
y(index_of_objectives(1),M + V + 1 + i) = Inf;
for j = 2 : length(index_of_objectives) - 1
next_obj  = sorted_based_on_objective(j + 1,V + i);
previous_obj  = sorted_based_on_objective(j - 1,V + i);
if (f_max - f_min == 0)
y(index_of_objectives(j),M + V + 1 + i) = Inf
else
```

```
295 y(index_of_objectives(j),M + V + 1 + i) = ...
296 (next_obj - previous_obj)/(f_max - f_min);
297 End
298 end
299 end
300 distance = [];
301 distance(:,1) = zeros(length(F(front).f),1);
302 for i = 1 : M
303 distance(:,1) = distance(:,1) + y(:,M + V + 1 + i);
304 end
305 y(:,M + V + 2) = distance;
306 y = y(:,1 : M + V + 2);
307 z(previous_index:current_index,:) = y;
308 end
309 f = z();
310 mu = 20;
311 mum = 20;
312 offspring_chromosome = ...
313 genetic_operator(parent_chromosome, ...
314 M, V, mu, mum, min_range, max_range);
315 [main_pop,temp] = size(chromosome);
316 [offspring_pop,temp] = size(offspring_chromosome);
317 % temp is a dummy variable.
318 clear temp
319 intermediate_chromosome(1:main_pop,:) = chromosome;
320 intermediate_chromosome(main_pop + 1 : main_pop + offspring_pop,1 :
    M+V) = ...offspring_chromosome;
321 intermediate_chromosome = ...
322 non_domination_sort_mod(intermediate_chromosome, M, V);
323 chromosome = replace_chromosome(intermediate_chromosome, M, V, pop);
324 if ~mod(i,100)
325 clc
326 fprintf('%d generations completed\n',i);
327 end
328 end
329 save solution.txt chromosome -ASCII
330 if M == 2
331 plot(chromosome(:,V + 1),chromosome(:,V + 2),'*');
332 elseif M ==3
333 plot3(chromosome(:,V + 1),chromosome(:,V + 2),chromosome(:,V + 3),'*');
334 end
```