Forschungsbericht 2023-18

An Intelligent Time and Performance Efficient Algorithm for Aircraft Design Optimization

Nesrin Çavuş

Deutsches Zentrum für Luft- und Raumfahrt
Institut für Luftverkehr
Hamburg

Deutsches Zentrum
DLR für Luft- und Raumfahrt

Forschungsbericht 2023-18

An Intelligent Time and Performance Efficient Algorithm for Aircraft Design Optimization

Nesrin Çavuş

Deutsches Zentrum für Luft- und Raumfahrt
Institut für Luftverkehr
Hamburg

146 Seiten
60 Bilder
57 Tabellen
127 Literaturstellen

Deutsches Zentrum
DLR für Luft- und Raumfahrt

Herausgeber:

Deutsches Zentrum
für Luft- und Raumfahrt e. V.
Wissenschaftliche Information
Linder Höhe
D-51147 Köln

ISSN 1434-8454
ISRN DLR-FB-2023-18
Erscheinungsjahr 2023

DOI: 10.57676/3axf-2f96

Flugzeugentwurf, Künstliche Intelligenz, Multidisziplinäres Design, Überschallflugzeuge, Unbemannte Kampfflugzeuge, Flugbahnoptimierung

Nesrin ÇAVUŞ
DLR, Institut für Luftverkehr, Hamburg

Ein intelligenter Zeit- und leistungseffizienter Algorithmus zur Optimierung des Flugzeugentwurfs
Technische Universität Hamburg

Die Optimierung des Flugzeugentwurfs erfordert die Beherrschung der komplexen Zusammenhänge mehrerer Disziplinen. Trotz seiner Abhängigkeit von einer Vielzahl unabhängiger Variablen zeichnet sich dieses komplexe Entwurfsproblem durch starke indirekte Verbindungen und eine daraus resultierende geringe Anzahl lokaler Minima aus. Abgesehen von klassischen Optimierungstechniken wie Genetische Algorithmen und Simulated Annealing usw. gibt es jedoch nicht viele Optimierungsalgorithmen, die sich speziell auf diesen anspruchsvollen Bereich fokussieren.

Kürzlich entwickelte intelligente Methoden, die auf selbstlernenden Algorithmen basieren, ermutigten die Suche nach einer diesem Bereich zugeordneten neuen Methode. Tatsächlich wird der in dieser Arbeit entwickelte Hybrid-Algorithmus (Cavus) auf zwei Hauptdesignfälle im Luft- und Raumfahrtbereich angewendet: Flugzeugentwurf- und Flugbahnoptimierung.

Die neue Hybridtechnik verwendet Klassifikation anstelle einer Interpolation der Entwurfspunkte, was für Optimierungsprobleme besser geeignet ist, die eine geringe Anzahl lokaler Minima haben und diskret sind. Es könnte als alternativer Ansatz für Vertrauensregionen bezeichnet werden: Teilregionen, umgekehrte Regionen, emulierte Regionen und aggregierte Regionen. Insbesondere wird der gesamte Designraum gescannt und die Wahrscheinlichkeitsdatensätze ohne Leistungsbeeinträchtigung gesammelt. Mit der zunehmenden Anzahl von Designvariablen wird die Auswahl der sofortigen Testpunkte strukturiert, indem das Wissen oder der Spur aus den vorangegangenen Iterationen genutzt wird. Der implementierte neue Ansatz ist in der Lage, die Anzahl der Versuchspunkte ohne große Kompromisse zu reduzieren. Dies erhöht die Effektivität der geringen Anzahl an Versuchspunkten, die nicht gleichmäßig verteilt werden konnten.

Die Trendanalyse zeigt, dass der Cavus-Algorithmus für die komplexen Designprobleme, die eine hohe Anzahl von Designvariablen, aber eine geringe Anzahl lokaler Minima haben, mit einer proportionalen Anzahl von Prüfpunkten konservativer ist, während es eine Erfolgsquote von bis zu 95-98% erreicht, um die erfolgreichen Muster zu finden.

Aircraft Design, Artificial Intelligence, Multidisciplinary Design, Supersonic Aircraft, Unmanned Combat Aerial Vehicle, Trajectory Optimization

(Published in English)

Nesrin ÇAVUŞ
German Aerospace Center (DLR), Institute of Air Transport, Hamburg

An Intelligent Time and Performance Efficient Algorithm for Aircraft Design Optimization

Hamburg University of Technology

Aircraft Design Optimization requires mastering of the complex interrelationships of multiple disciplines. Despite its dependency on a diverse number of independent variables, this complex design problem has favourable nature as having strong indirect links and as a result a low number of local minimums. However, unlike other classical optimization techniques such as Genetic Algorithm and Simulated Annealing etc., there are not many optimization algorithms focussed specifically on this challenging area.

Recently developed intelligent methods that are based on self-learning algorithms encouraged finding a new method dedicated to this area. Indeed, the hybrid (Cavus) algorithm developed in this thesis is applied two main design cases in aerospace area: aircraft design optimization and trajectory optimization.

The new hybrid technique uses classification rather than interpolation of the design points, which is more suitable for optimization problems that have low number of local minimums and are discrete. It could be called as an alternative approach to trust regions: partial regions, flipped regions, emulated regions and aggregated regions. Especially, it scans the whole design space and collects the probability records without detriment to performance. With the increasing number of design variables, the selection of the instant trial points is structured by using the knowledge, or trail, from the preceding iterations. The implemented new approach is capable of reducing the number of trial points without much compromise. This increases the effectivity of the low number of trial points which could not be evenly distributed.

The trend analysis shows that, for the complex design problems that have a high number of design variables but a low number of local minimums the Cavus algorithm is more conservative with a proportional number of trial points whilst yielding up to 95-98% success rate in finding the successful patterns.

An Intelligent Time and Performance Efficient Algorithm for Aircraft Design Optimization

Vom Promotionsausschuss der
Technischen Universität Hamburg

zur Erlangung des akademischen Grades

Doktor-Ingenieurin (Dr.-Ing.)

genehmigte Dissertation

von
Nesrin Çavuş

aus
İzmit, Türkei

2023

Übersicht der Gutachter

1. Gutachter	Prof. Dr.-Ing. Volker Gollnick
2. Gutachter	Prof. Dr. Ozan Tekinalp
Vorsitzender des Prüfungsausschusses:	Prof. Dr. rer. nat. Sabine Le Borne
Tag der mündlichen Prüfung:	27.03.2023

Abstract

Aircraft Design Optimization requires mastering of the complex interrelationships of multiple disciplines. Despite its dependency on a diverse number of independent variables, this complex design problem has favourable nature as having strong indirect links and as a result a low number of local minimums. However, unlike other classical optimization techniques such as Genetic Algorithm and Simulated Annealing etc., there are not many optimization algorithms focussed specifically on this challenging area.

Recently developed intelligent methods that are based on self-learning algorithms encouraged finding a new method dedicated to this area. Indeed, the hybrid (Cavus) algorithm developed in this thesis is applied to two main design cases in aerospace area: aircraft design optimization and trajectory optimization.

The new hybrid technique uses classification rather than interpolation of the design points, which is more suitable for optimization problems that have low number of local minimums and are discrete. It could be called as an alternative approach to trust regions: partial regions, flipped regions, emulated regions and aggregated regions. Especially, it scans the whole design space and collects the probability records without detriment to performance. With the increasing number of design variables, the selection of the instant trial points is structured by using the knowledge, or trail, from the preceding iterations. The implemented new approach is capable of reducing the number of trial points without much compromise. This increases the effectivity of the low number of trial points which could not be evenly distributed.

The trend analysis shows that, for the complex design problems that have a high number of design variables but a low number of local minimums the Cavus algorithm is more conservative with a proportional number of trial points whilst yielding up to 95-98% success rate in finding the successful patterns.

Keywords: Aircraft Design, Artificial Intelligence, Multidisciplinary Design, Supersonic Aircraft, Unmanned Combat Aerial Vehicle, Trajectory Optimization

To my mother Ayşe Çavuş

Table of Contents

Abstract iii
Dedication v
1. Introduction and Motivation 1
1.1. Objective and Originality of the Thesis 2
1.2. Organization of the Thesis 7
2. Artificial Intelligence / Intelligent Systems 8
2.1. State of the Art Artificial Intelligence 8
2.1.1. What is Artificial Intelligence? 9
2.1.2. History of Artificial Intelligence 11
2.1.3. Artificial Intelligence in Aerospace 13
2.2. Knowledge Based Methods 14
2.3. Computational Intelligence 15
2.3.1. Genetic Algorithm 18
2.3.2. Simulated Annealing 20
2.3.3. Pattern Search 23
2.3.4. Kriging 26
2.3.5. Globex Algorithm 29
2.3.6. Artificial Neural Networks 32
2.4. Probability 34
2.4.1. Probabilistic Neural Networks 40
2.5. Comparison of the Algorithms 42
3. Cavus Algorithm 44
3.1. Trend Analysis for the number of Training Points 58
3.2. Test Case: Rosenbrock Function 59
3.2.1. Method description and the results for Rosenbrock Function for 2 variables . 60
3.2.2. The results for Rosenbrock Function for 7 variables 73
3.2.3. The results for the "Rosenbrock" Function for 14 variables 77
4. Design Cases 81
4.1. Aircraft Design 81
4.1.1. Conceptual design of an unmanned supersonic aircraft 82
4.1.2. Mission Profile 87
4.1.3. Initial Sizing 88
4.1.4. Wing Configuration 89
4.1.5. Fuselage Configuration 91
4.1.6. Propulsion System 92

4.1.7. Horizontal and Vertical Tail Configuration 93
4.1.8. Landing Gears 94
4.1.9. Aerodynamics 94
4.1.10. Weight and Stability 95
4.1.11. Performance 96
4.1.12. Structural Load 100
4.1.13. Cost Model 101
4.1.14. Verification of the Aircraft Design Part 102
4.2. Trajectory Optimization 106
5. Results 108
5.1. Design Case I (Unmanned Supersonic Aircraft) 108
5.2. Design Case II (Trajectory Optimization) 115
6. Conclusion 118
Bibliography 120
Appendix 130
PNN Classification 130

List of Figures

Figure 1.1 The Complete Aircraft Design Framework [7] 2
Figure 1.2 Common sequence of analysis modules in aircraft design [8] 3
Figure 1.3 Fuselage Preliminary Design Process [3] 4
Figure 1.4 General input-output relations in aerospace engineering problems 6
Figure 2.1 Categories of Intelligent system software [18] 9
Figure 2.2 Approaches of AI [23] 10
Figure 2.3 Relations of components for different AI systems [23] 11
Figure 2.4 An agent interacting with an environment [24] 15
Figure 2.5 Core Genetic Algorithm [45] 20
Figure 2.6 Simulated Annealing [44] 21
Figure 2.7 Pseudo code of the Pattern Search Algorithm [44] 24
Figure 2.8 All the possible subsets of the steps for coordinate search in $R2$ [17] 25
Figure 2.9 The pattern in $R2$ with a step length control parameter Δk [17] 26
Figure 2.10 Determination of the main search direction in the case of a two variable problem [16] 30
Figure 2.11 Assumed artificial parabola for a search of a minimum [16] 31
Figure 2.12 Two-dimensional situation near a boundary [16] 32
Figure 2.13 Probability density and distribution functions of a continuous random variable X (a) density function; (b) distribution function [43] 37
Figure 2.14 Probability theory [54] 37
Figure 2.15 Probabilistic Neural Networks [14] 40
Figure 2.16 The smoothing effect of σ on an estimated PDF from 5 samples [14] 41
Figure 3.1 Multidisciplinary Design Optimization Flowchart for Cavus Algorithm 46
Figure 3.2 Flowchart of the Cavus Algorithm 48
Figure 3.3 Patterns for one variable (*k=3*) 49
Figure 3.4 Patterns for two variables (*k=3*) 50
Figure 3.5 Patterns for one variable (*k=5*) 50
Figure 3.6 Patterns for two variables (*k=5*) 51
Figure 3.7 Logical Description of the Cavus Algorithm 56
Figure 3.8 Selecting promising patterns with PNN 57
Figure 3.9 Rosenbrock Function (3D view) 60
Figure 3.10 Rosenbrock Function (2D view) 61
Figure 3.11 First set of data points 62
Figure 3.12 PNN classification 65
Figure 3.13 Second set of data points 66
Figure 3.14 Second set of the data points grouped with the related pattern numbers 67
Figure 3.15 Third set of data points 69

Figure 3.16 Fourth set of data points 69
Figure 3.17 Fifth set of data points 70
Figure 3.18 Sixth set of data points 70
Figure 3.19 Seventh set of data points 71
Figure 3.20 Eighth set of data points.....71
Figure 3.21 Ninth set of data points 72
Figure 3.22 Ninth set of data points in detail view.....72
Figure 3.23 Ranks of the first training set 74
Figure 3.24 Comparison of the algorithms for Rosenbrock function (Results).....80
Figure 3.25 Comparison of the algorithms for Rosenbrock function (Function evaluations) 80
Figure 4.1 Flowchart of the Aircraft Design Algorithm 84
Figure 4.2 Mission segments 87
Figure 4.3 NACA 64A210 89
Figure 4.4 Approximation for integral fuel tank volume, available in a linear lofted wing [84] [86].....90
Figure 4.5 Fuselage sections [84] 91
Figure 4.6 Inlet locations - buried engines (Nose) [2] 92
Figure 4.7 Conventional tail [2].....93
Figure 4.8 Triple slotted flap [2].....94
Figure 4.9 Centre of gravity-exaggerated views [87] 96
Figure 4.10 Top View of the resulting UCAV design 105
Figure 4.11 Top View of reference F-16 design.....105
Figure 4.12 Map of the Trajectory Optimization 107
Figure 5.1 Top and side views of the resultant UCAVs 112
Figure 5.2 Optimal Way Points 115
Figure 5.3 Resultant Optimal Way Points 117

List of Tables

Table 2.1 Methods of Operations Research [43] 16
Table 2.2 Unconstrained Minimization Methods [43] 17
Table 2.3 Constrained Optimization Techniques [43] 17
Table 2.4 Comparison of Random search and Simulated Annealing [45] 22
Table 2.5 Application Evaluation of ANN Models [42] 33
Table 2.6 Types of probability distributions (analytical models) [43] 37
Table 2.7 Main types of probabilistic graphical models [47] 39
Table 2.8 Comparison of the Algorithms 43
Table 3.1 Optimization results for the fixed number of training points (n=10) 58
Table 3.2 Optimization results for the changing number of training points ($n=k^m$) 59
Table 3.3 Design variables with their boundaries and values for the first training set 61
Table 3.4 Successful patterns 63
Table 3.5 Unsuccessful patterns 63
Table 3.6 Total possible patterns 63
Table 3.7 Successful patterns with classes 64
Table 3.8 Possible successful patterns 64
Table 3.9 Possible unsuccessful patterns 64
Table 3.10 Promising patterns with their bounds 65
Table 3.11 The unsystematically distributed points for each pattern and the results 67
Table 3.12 Normalized min and mean values 68
Table 3.13 Actual patterns with the promising patterns for the next run 68
Table 3.14 Design variables with the results for the first training set (m=7) 73
Table 3.15 The promising patterns for the next run 74
Table 3.16 The rank of the promising patterns 75
Table 3.17 The starting point for Globex Algorithm 75
Table 3.18 The starting step sizes for Globex Algorithm 76
Table 3.19 The results for the "Rosenbrock" Function with 7 variables 76
Table 3.20 The promising patterns for the next run $\mathrm{m} = 14$ 77
Table 3.21 The rank of the promising patterns $\mathrm{m} = 14$ 78
Table 3.22 The starting point for Globex Algorithm 78
Table 3.23 The starting step sizes for Globex Algorithm 78
Table 3.24 The resultant variable values for Rosenbrock Function with 14 variables 78
Table 3.25 The results for Rosenbrock Function with 14 variables 79
Table 3.26 Comparison of the algorithms with the changing variable numbers 79
Table 4.1 Aircraft Design part inputs and outputs 85
Table 4.2 F-16 Control Surface Actuator Models [88] 95
Table 4.3 Design Inputs for the F-16 aircraft 102

Table 4.4 Examples of Airplane Program Production Runs [84] 103
Table 4.5 Assumed Inputs 103
Table 4.6 UCAV Comparison Table with F-16 104
Table 5.1 UCAV Constraints 109
Table 5.2 Constants in UCAV optimizations 109
Table 5.3 Design variables and their boundaries 110
Table 5.4 Variables and optimization results for UCAV 111
Table 5.5 Sample UAVs with their Specifications 113
Table 5.6 The starting point for Globex Algorithm 114
Table 5.7 The starting step sizes for Globex Algorithm 114
Table 5.8 The Results for the Aircraft Design Case 114
Table 5.9 Lower and Upper Bounds of the Waypoints 115
Table 5.10 The result of the Cavus algorithm 115
Table 5.11 The starting point for the Trajectory Calculation 115
Table 5.12 The result of the Fmincon algorithm for the Trajectory optimization 116
Table 5.13 The starting point of the Globex Algorithm for the Trajectory optimization . 116
Table 5.14 The starting step sizes of the Globex Algorithm for the Trajectory optimization 116
Table 5.15 The result of the Globex algorithm 116
Table 5.16 The result of the Genetic Algorithm 116
Table 5.17 Comparison of the results for the Trajectory Optimization 116

Nomenclature

Aircraft Design Part:

a	Speed of sound
a_t	Tail lift curve slope
A	Area, Aft, Avionics
ac	Aerodynamic center
ADALINE	Adaptive Linear Element
ANN	Artificial Neural Networks
AR	Aspect ratio
b	Span
BPR	Bypass Ratio
c	Chord, Combat (action)
cg	Center of gravity
cr	Cruise
C_d	Airfoil drag coefficient
C_{d0}	Airfoil drag coefficient at zero angle of attack
C_l	Airfoil lift coefficient
C_{l0}	Airfoil lift coefficient at zero angle of attack
$C_{l\alpha}$	Airfoil lift curve slope
$C_{m\alpha}$	Airfoil pitching moment curve slope
c_r	Root chord length
c_t	Tip chord length
$\bar{c}$	Mean aerodynamic chord length
C_D	Drag coefficient
$C_{D\alpha}$	Airplane drag curve slope
C_{D0}	Drag coefficient at zero angle of attack
$C_{DL\&P}$	Drag coefficient with leakages and protuberances
C_f	Flat-plate skin-friction drag coefficient
C_L	Lift coefficient
$C_{L\alpha}$	Airplane lift curve slope
d	Action time, Dash
D	Drag, Diameter, Development
DAPCA	Development and Procurement Cost of Aircraft model
e	Oswald efficiency factor, Engineering
E	Endurance, Engine(production)
eng	Engine
exp	Exposed
F	Flight test

FF	Fitness Function
FTA	Number of flight test aircraft
fus	Fuselage
F_M	Total force on the two main wheels
F_N	Force on nose wheel
g	Gravitational acceleration
H	Height, hour
HT	Horizontal tail
KAUS	Knowledge Acquisition and Utilization System
K_{vs}	Variable sweep constant
L	Lift, Length, Landing
LE	Leading edge
LG	Landing gear
ltr	Loiter
l_t	Tail arm
M	Moment, Mach number, Mid-body, Manufacturing
max	Maximum
min	Minimum
misc	Miscellaneous
MCMOSA	Multiple Cooling Multi Objective Simulated Annealing Algorithm
MLL	Multi-Layer Logic
MLP	Multi-Layer Perceptron
n	Load factor
n_z	Ultimate load factor
net	Net
N	Nose
plf	Planform
q	Dynamic pressure, Quality control
Q	Interference factor
R	Radius, range, hourly rate
r	Root
Re	Reynolds number
Ref	Reference
ROC	Rate of climb
s	Steady state
S	Reference area
sh	Sears-Haack body
SFC	Specific fuel consumption

s_L	Landing distance
s_{TO}	Take-off distance
t	Time, Thickness, Tooling
T	Thrust
TO	Take-off
TOG	Take-off ground roll
T_{av}	Thrust available
T_i	Turbine inlet temperature
UCAV	Unmanned Combat Aerial Vehicle
V	Velocity
VT	Vertical tail
V_f	Fuel volume
w	Width, Wing
W	Weight
wb	Wing-body
wet	Wetted
W_0	Take-off gross weight
W_{dg}	Design gross weight
W_e	Empty weight
W_f	Fuel weight
xx	Number of action turns
x_{acwb}	Aerodynamic center of the wing-body
x_n	Neutral point
$\bar{x}$	Mean aerodynamic chord x location
$\bar{y}$	Mean aerodynamic chord y location
$\bar{z}$	Mean aerodynamic chord z location
θ	Pitch angle
ρ	Density
Δ	Change
μ	Friction coefficient
λ	Taper ratio
Λ	Sweep angle
τ	Ratio of tip and root thickness ratios
Γ	Dihedral angle
β	Prandtl Glauert
σ	Density ratio
ω	Rate of turn

#	Number of
0	Sea level, starting point, binary value
∞	Free stream

Optimization Part:

$f(x)$	Probability density function
k	Number of intervals
m	Number of variables
n	Number of training points
pt	Number of patterns for one variable
p^m	Number of total patterns

Knowledge should mean a full grasp of knowledge:
Knowledge means to know yourself, heart and soul.
If you have failed to understand yourself,
Then all of your reading has missed its call.[1]
YUNUS EMRE

1. Introduction and Motivation

'Know yourself', as Socrates said, may be the best way to start to explain why people began from themselves intuitively to search for techniques to ease their life. Because their bodies, the environment in which they leave, so the universe function in an excellent order. Although human beings are far to understand the real structure of this excellence, they are gifted to be attracted by, and have the ability to start to investigate. But they cannot encompass/comprehend anything beyond as much as allowed [1]. That is why this thesis is based on "Artificial Intelligence" which tries to mimic the functioning of the human intelligence, i.e. another excellence. Artificial Intelligence may be the unpredictably large research area for us. Although it is still under development, it has affected and eased our life gradually. Thus, in this study its effect in the field of aerospace is emphasized.

Another motivation item of this study can be explained through "Probability", which is defined as the likelihood of an event to occur. If events are more than one, the sum of the probabilities of all possible occurrences is accepted to be equal to "1"; which inherently wipes out the "coincidence theory".

In this thesis, a dedicated optimization algorithm for a conceptual aircraft design was developed based on these two motivation items. Many classical mathematical optimization techniques are applied in this area [2]. These classical methods use so-called random walk or creeping, which need high number of function evaluations due to probable poor starting points, or mathematically true but illogical selections. Besides, due to the structure of the aircraft design, the derivative-based methods which are much more powerful than classical methods are very difficult and may not worth to apply. However, the aircraft design has many advantages over other optimization problems and deserves a dedicated optimization method to converge to better results with less function evaluations. An aircraft is composed of systems of systems and has a closed loop process in which the energy is conserved. So that, the possible degree of success of the unknowns can be extracted from the knowns of the design space based on their probabilities. At that point, this study differs from other researches in this domain; by treating the constrained design space as a whole and deciding on the next trial point with a classification algorithm, instead of any so-called random walk or interpolating algorithms which are commonly applied until now. Indeed it uses two keys to have better selection of design points; learning from previous data, i.e. artificial intelligence, and the coherence of our universe, i.e. (here, using) the rules of probability. As a result, any poor attempts are diminished before they are tested. This dedicated method for a conceptual aircraft design problem can be applied to any design problem which can be defined as a closed loop process, also.

[1] *Translated from Turkish: İlim ilim bilmektir, İlim kendin bilmektir. Sen kendini bilmezsin, Ya nice okumaktır.*

1.1. Objective and Originality of the Thesis

With increasing demand on high quality, high performance products and limited resources it is always required to have efficient selections on the components. This efficient selection process is called as optimization for a design problem, and these high quality high performance products and limited resources are defined as the objectives and the constraints, respectively. Then the components whose values change throughout the selection process are named as the independent variables of a design problem. According to the design problem, the designer may choose one or more independent variables. If there is just one objective (dependent variable) this process is called *single objective optimization*, if there are more than one objectives then it is a *multiobjective optimization* problem. When the design problem compromises different disciplines then the optimization process called *multidisciplinary design optimization* (MDO). The focus of this thesis is aircraft conceptual design optimization. Indeed, Aircraft design is a complex process that requires different fidelity levels of multidiscipline, accordingly different frameworks, expertise of engineers and significant efforts [3], [4]. It consists of many highly coupled systems, subsystems and related design parameters [5], [6].

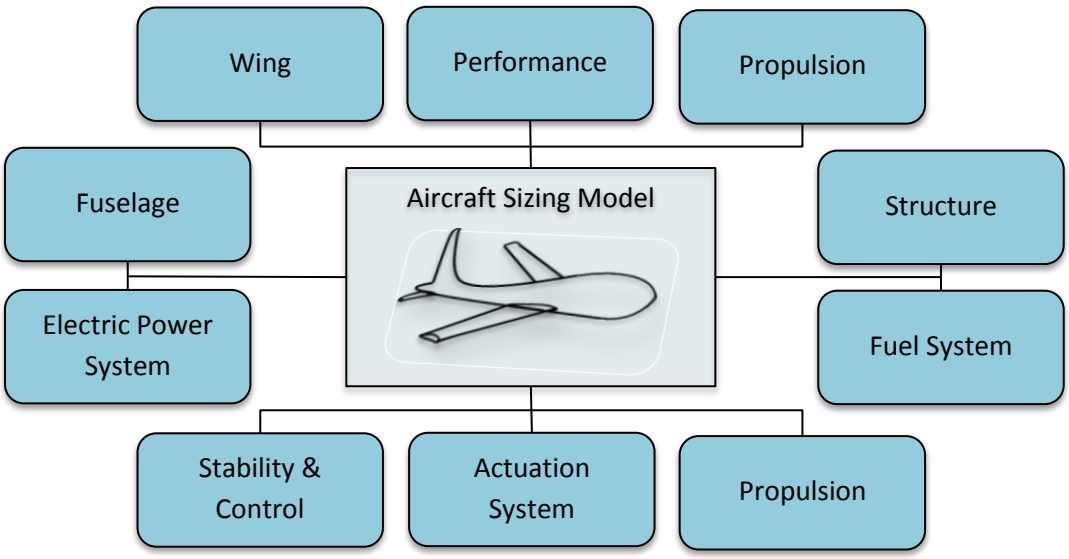


Figure 1.1 The Complete Aircraft Design Framework [7]

A multidisciplinary design tool for an aircraft design is presented by Amadori et al. in [7] to meet the requirements of modern complex product development and it is framework is illustrated in Figure 1.1 with all the modules and their connections. A straightforward iterative and sequential layout combining multiple design modules are shown by Zill [8] in Figure 1.2. However, each of these design modules also includes complex relations both internally and externally. Besides, at the conceptual analysis stage a huge number of (unknown) parameters are used with the empirical equations of classical sizing approach [9].

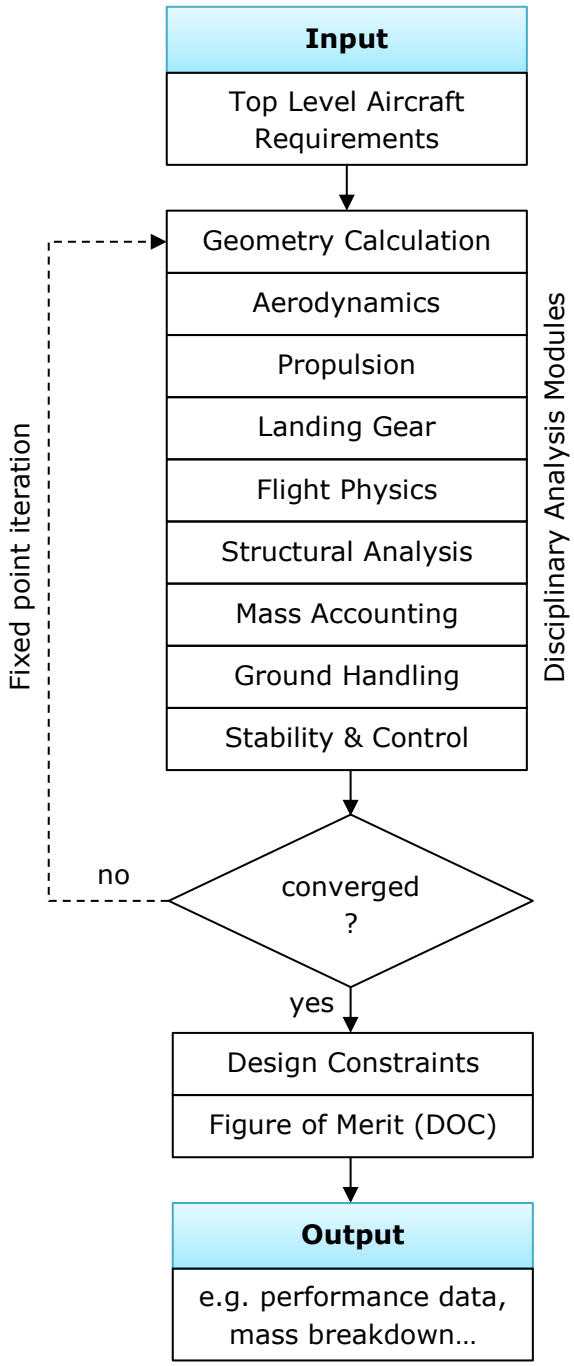


Figure 1.2 Common sequence of analysis modules in aircraft design [8]

To know explicitly or to trace these implicit parameters are not the case, because they are based on the statistical data. Thus, the classical aircraft sizing process can be

defined as a *statistical knowledge-based engineering method* that uses small amount of parameters, which can be explicitly defined, and a large number of statistic data [9].

To manage the complexity and to capture the 'know-how' the artificial intelligence techniques are employed; the computer aided design, like computational analysis, is used in all forms to find the superior product with an iterative design process [3]. An example to the iteration process involved in an aircraft design is given by Saggu [3] in Figure 1.3. In the figure the complex relations of a fuselage-wing box design are shown.

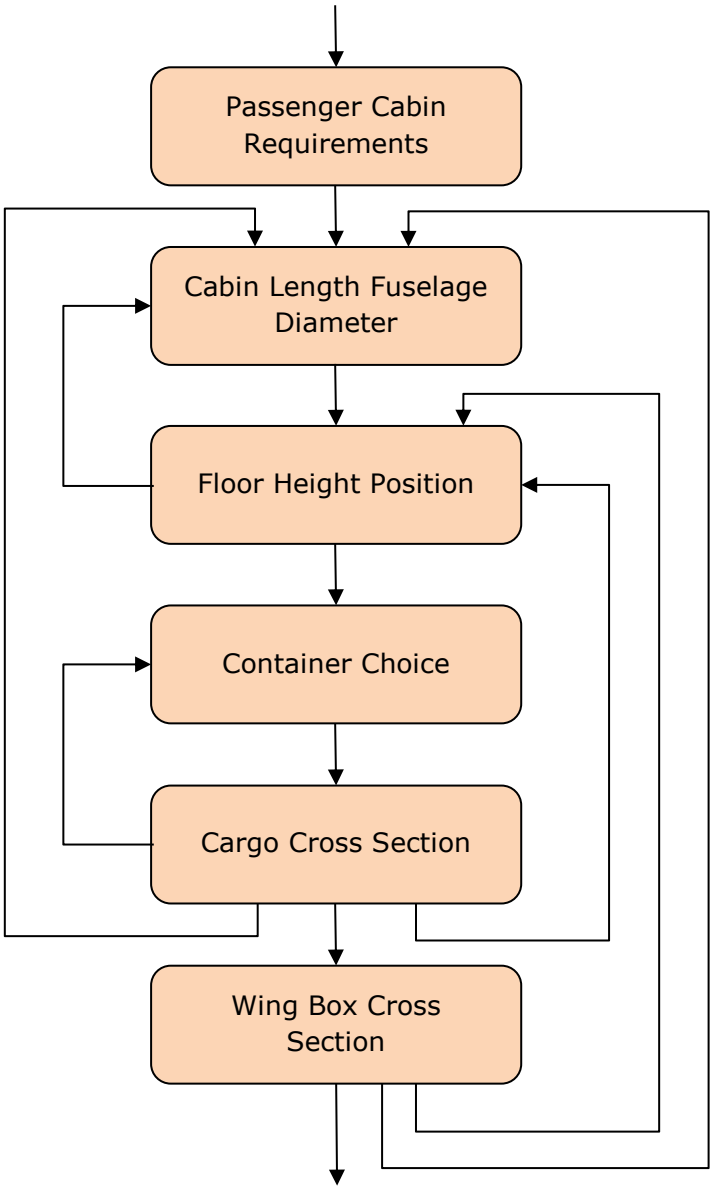


Figure 1.3 Fuselage Preliminary Design Process [3]

The amount of both dependent and independent variables have great influence on the complexity of the design process, and hence this burden impresses the time to reach the results and also the quality of the results. Gelsey et al. [10] used a gradient-based constraint optimization algorithm for a parametric conceptual aircraft design process. An automated search procedure with model assumption violation techniques is applied. The quality of candidate designs for complex artifacts is evaluated by computational simulation at each step. *Boolean* and *model constraints strategies* are used during this

evaluation process. Instead of using the experts' domain knowledge in the loop, it is aimed to have effective information exchange between a simulator and an automated search procedure while violating the constraints and to find feasible design points. A cost improvement of two orders of magnitude is observed.

Besides complexity, the huge number of design variables is another concern in aircraft design. A tailless unmanned air vehicle with 44 variables is optimized by Sobieski and Kroo [11]. A response surface estimation technique with simple trust region algorithm is implemented at this the collaborative optimization study. It is proved that, when a complete subsystem optimization is required the response surface optimization is relatively inexpensive method. Moreover, computational expense of generating a response surface is reduced almost 50% as recognizing extra point information implicitly in each sub-model.

Many different optimization techniques are used in different problem areas. The type of optimization technique to be used in a special area is another concern that leads runtime and results in a manner to be better or worse. As an example, one method could fit a problem in which it is desired to make some predictions on economics, while other method could fit better to a mechanical problem. Therefore, it is very hard to say for any technique that is the best technique for all of the situations one may meet.

In aerospace engineering, to understand the complex design problems and reach the viable and accurate design solutions the guidance of probabilistic design techniques and simulation are required by the designers increasingly [12].

In the detailed work of Bashir and Hasan [13], the basic operations of Genetic Algorithm are presented with Rosenbrock function. In this work, it is seen that after crossover, mutation and selection operations, the two new chromosomes have the same variable values. Moreover, the previous iteration has also the same variable values in one of the chromosomes. This is a good example of poor selections in Genetic Algorithm. This kind of excess function evaluations can be encountered during the process of population based algorithms. When these excess function evaluations occur as optimizing an aircraft design, the wasted run time does not be similar to Rosenbrock function but even it could be hours or days. This is due to the multiplication factor of the time spent while calculating one design solution.

Therefore, the purpose of this study is to propose a novel algorithm which can be used for engineering design optimization problems, that has low number of local minimums but a high number of variables relating complex structures. Even though, the success of the algorithm is examined here by 14 independent variables, it can be applied to more number of variables. The target is to reduce the function evaluations by reducing the poor parts of the design space based on the probabilities extracted from previous trials. Due to the fact that, this algorithm is neither restricted by step sizes nor diverged by so-called random walks, it has better results compared to other algorithms applied in this study.

In this research, the design cases are selected from the applications of aerospace engineering area. Aircraft design optimization and an aircraft mission have complex dependencies; but they have clear standard segments and each segment with its requirements – inputs and outputs – are well-known; hence these platforms are practical to be selected. The general input-output relations are illustrated in Figure 1.4.

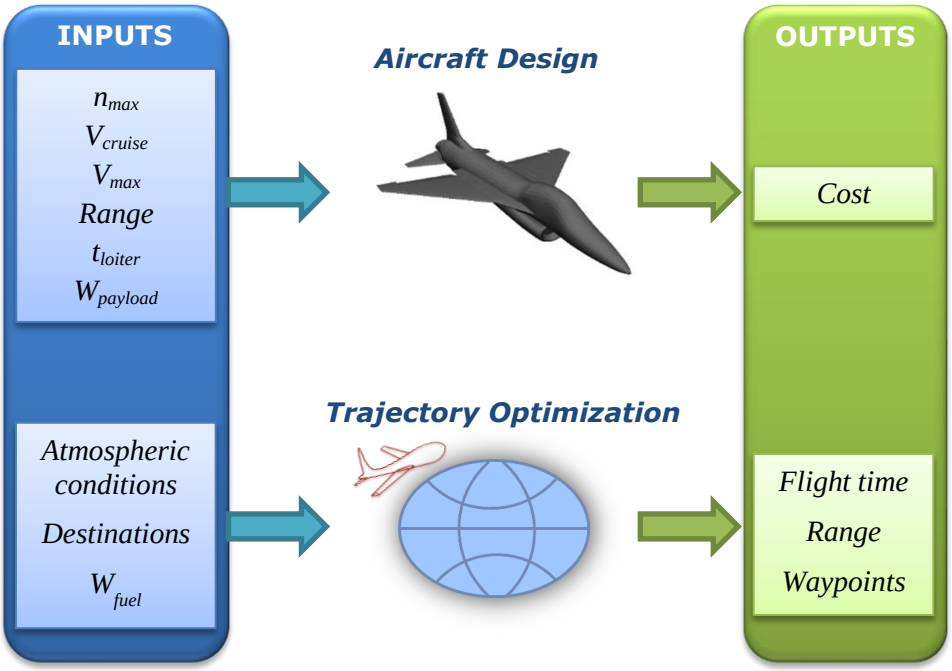


Figure 1.4 General input-output relations in aerospace engineering problems

As well as, these closed loop systems are more conservative than many statistical problems; thus it is worth to use directed search methods. However, when the number of independent design variables and the complexity increase, the statistics based methods are helpful at intermediate steps of an optimization algorithm [12].

Further, optimization process of an aircraft can be structured with two main sub functions, optimization part and aircraft design part, which also includes many sub-optimization loops. The selection of the independent variables and the constraints with their ranges has a direct effect on the performance of an optimization process. Any poor selection of a design point on a design space cost extra time, which may be totally useless. Therefore, not only the optimization part itself, also the aircraft design part expends the runtime, which can be thought as multiplication of these times. Thus, the total process should be improved not to select unfavourable design points in order to save the run time and have better convergence for optimum design.

As the structure of the analysis is concerned, widely used methods like regression and Kriging do interpolations between set of points. These sets are composed of dependent and independent variables. The idea in this study is to use instantaneous associations between dependent and independent variables. Besides considering the approximated linear/nonlinear relations, the behaviour of the dependent variable can be stored while shifting one design point to another. With storing instantaneous behaviours of variables, the redundant candidate design points can be avoided during the optimization process.

In this study, with a given or produced training data set, the relations between independent variables and dependent variables are examined; and then search areas are selected based on the successful changes on the objective function values. Stored actions on (or differences between) the variables are considered as the patterns, which are classified according to the function values as successful or unsuccessful. Resultant successful patterns are used to extend and deepen the design search space with the help

of Probabilistic Neural Networks, which was introduced by Specht [14]. This kind of Neural Networks is used for its success on classification and pattern recognition. The method is given in detail in section 0.

The presented study here can be clustered among the single level optimization techniques of aircraft design optimization strategies as mentioned by Vanderplaatz in [15], and it can belong to the group of direct search methods which does not depend on the derivatives like the optimization method presented by Jacob [16]. Also, the technique can be categorized to the pattern search methods presented by Torczon [17].

In this regard, the originality of the work in aircraft design optimization environment stands on three principles. At first, the method depends on neither the design points nor the gradients alone; instead it uses the bilateral combinations of the design points to perceive the bounded design space. Second, the successive design points are not generated from the current points; instead they are systematically improved by the clustered parts of the design space, which are gained from the current information. Third, depending on the success of the neighbouring pattern selection as classifying the patterns, a novel dimension reduction technique is used to increase the computational performance.

The developed algorithm can be applied to a wide range of design problems which are linear, nonlinear, quadratic, convex or non-convex. In this study, Rosenbrock, a non-convex function, also a conceptual design of an aircraft and a trajectory of a passenger aircraft are taken as test cases for the proposed algorithm.

Indeed, the objective of the thesis is to develop an intelligent algorithm which can be a bridge between aerospace and mathematics domains by handling one of the artificial intelligence techniques via an unconventional aspect. The memory usage, a serious drawback of the artificial intelligence algorithms, is overcome with a novel case specific dimension reduction technique. As a result, the novel algorithm the advantage reducing the required minimum number of function evaluations for aerospace design optimization problems.

1.2. Organization of the Thesis

At first, artificial intelligence and its state of the art aerospace engineering are dealt with. Then, the method is described and illustrated in detail with Rosenbrock function. Due to having many local minimums and wide search space Rosenbrock function is one of the popular test methods for single objective optimization that has the flexibility to increase the independent design variables, also. The effectiveness of the algorithm is shown with Rosenbrock function with two, seven, ten and also fourteen variables. A trend analysis is also done for selecting the number of training points during the optimization process, which also proves the efficiency of the algorithm while increasing the variable numbers.

Then two different test cases from aerospace engineering, i.e. the conceptual design of a supersonic unmanned aircraft and trajectory optimization, were applied for the multidisciplinary single optimization case.

2. Artificial Intelligence / Intelligent Systems

2.1. State of the Art Artificial Intelligence

Knowledge based systems, computational intelligence and hybrids are specified as the tools of artificial intelligence by Hopgood in [18]. In order to use the advantages of both methods, a hybrid method is introduced in this study which is composed of agents and Probabilistic Neural Networks algorithm. Agents work on gridded search space to find out and sort the shifting actions of the values, and Probabilistic Neural Networks is used for classification of the successful and promising patterns.

Recently, the use of artificial intelligence tools continues to increase in aircraft multidisciplinary design optimization due to their successful implementations. A Knowledge Based Engineering approach to support aircraft multidisciplinary optimization was used by La Rocca and van Tooren [19] to develop both conventional and novel geometries. Finite element analysis models are generated and time reduction is gained with the automated method.

Another method, Concurrent Learning was used with Adaptive Neural Networks based approximate models by Gursoy and Yavrucuk [20] for a nonlinear fixed-wing aircraft model. Non-iterative two models were used to estimate the direct adaptive limit and the control margins. Network weights were updated from both past and current information, with which resultant values were better calculated.

To decrease the aircraft design cycle time, two different artificial intelligence algorithms are used by Oroumieh et al. in [21]; Neural Networks and Fuzzy Logic. Aircraft weight, engine thrust and wing area were determined with the applied methods at the early phase of the aircraft design process. A specific class of light business jets is selected as a design case to approximate the take-off wing loading and take-off thrust loading. The actual results are approximated about ten percent for the preliminary design phase.

A fuzzy logic based artificial intelligence algorithm was applied to an unmanned combat aerial vehicle control system by Ernest et al. [22]. The success of the algorithm was proved in high fidelity simulation environment. The algorithm was found to be highly responsive to complex situations and uncertainties.

As introduced above, artificial intelligence methods with knowledge based techniques are used by many designers to improve the design solutions and the required run time. In this study a hybrid technique is used to increase the efficiency of the optimization process by reducing the required number of function evaluations to converge the optimum point.

The method studied here has the advantage on other guided random search techniques with assigning the directions on the search space for the variables while searching and classifying the promising grids. With this method the training sets are used to decide on the next search space with the produced values of the cumulated runs. In the following sections the methodology is explained more in detail.

2.1.1. What is Artificial Intelligence?

Artificial Intelligence is the science of mimicking human mental capabilities in a computer, which are reasoning, understanding, perceiving, recognizing, recalling, imagination, and more [18].

In the next section, the contributions of scientist to Artificial Intelligence are described in brief. The different techniques used in AI are categorized and as illustrated in Figure 2.1 by Hopgood [18] in three nested types; those are Knowledge-based Systems, Computational intelligence and hybrids.

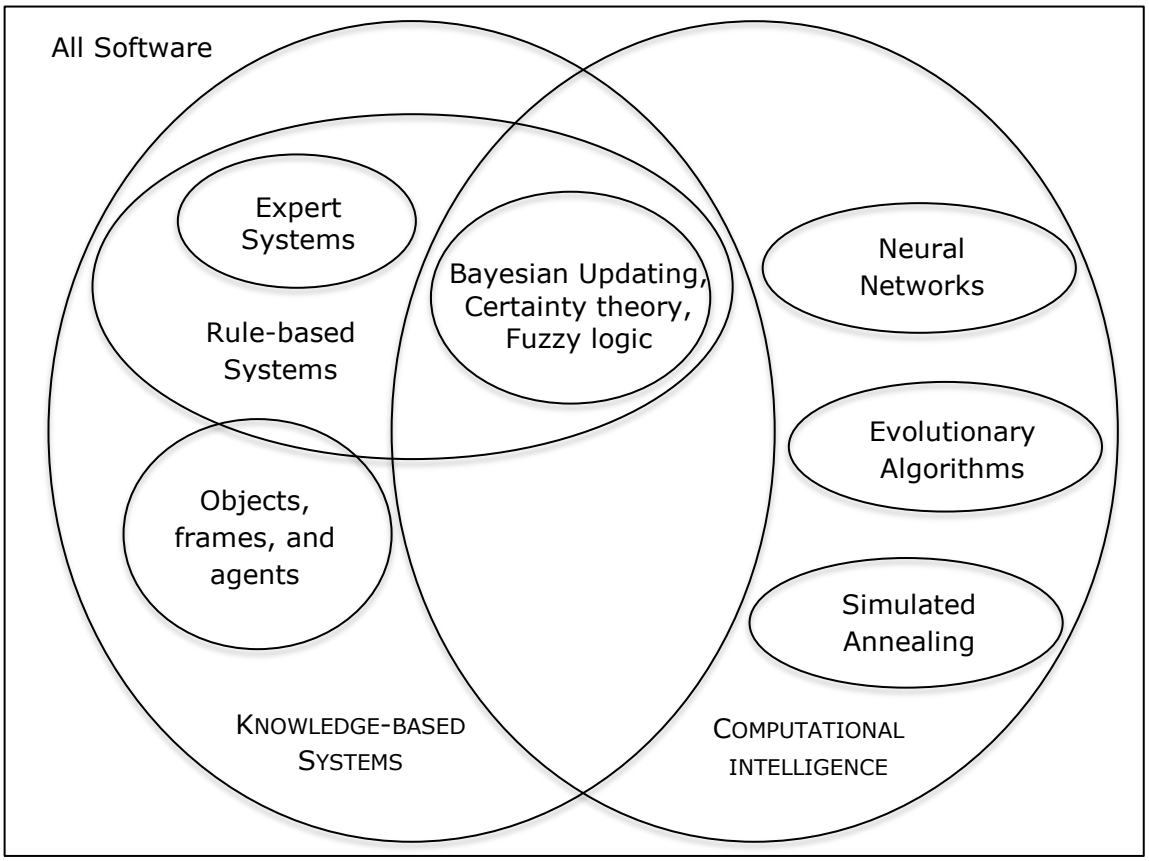


Figure 2.1 Categories of Intelligent system software [18]

According to Goodfellow et al. [23] the true challenge of artificial intelligence (AI) is to solve problems that people solve easily and intuitively as automatic like recognizing spoken words and figures, which are also hard for people to describe formally. They give

a solution to this difficulty as defining concepts which have relations to each other. Then, this hierarchy of concepts which is structured from simple to complicated ones allows computers to learn from experience and understand the world. Because of this layered formation which combines many layers this approach is called as deep learning in AI. A Venn diagram shown in Figure 2.2 is given by Goodfellow et al. [23] to illustrate the applications and relations of approaches with AI technology.

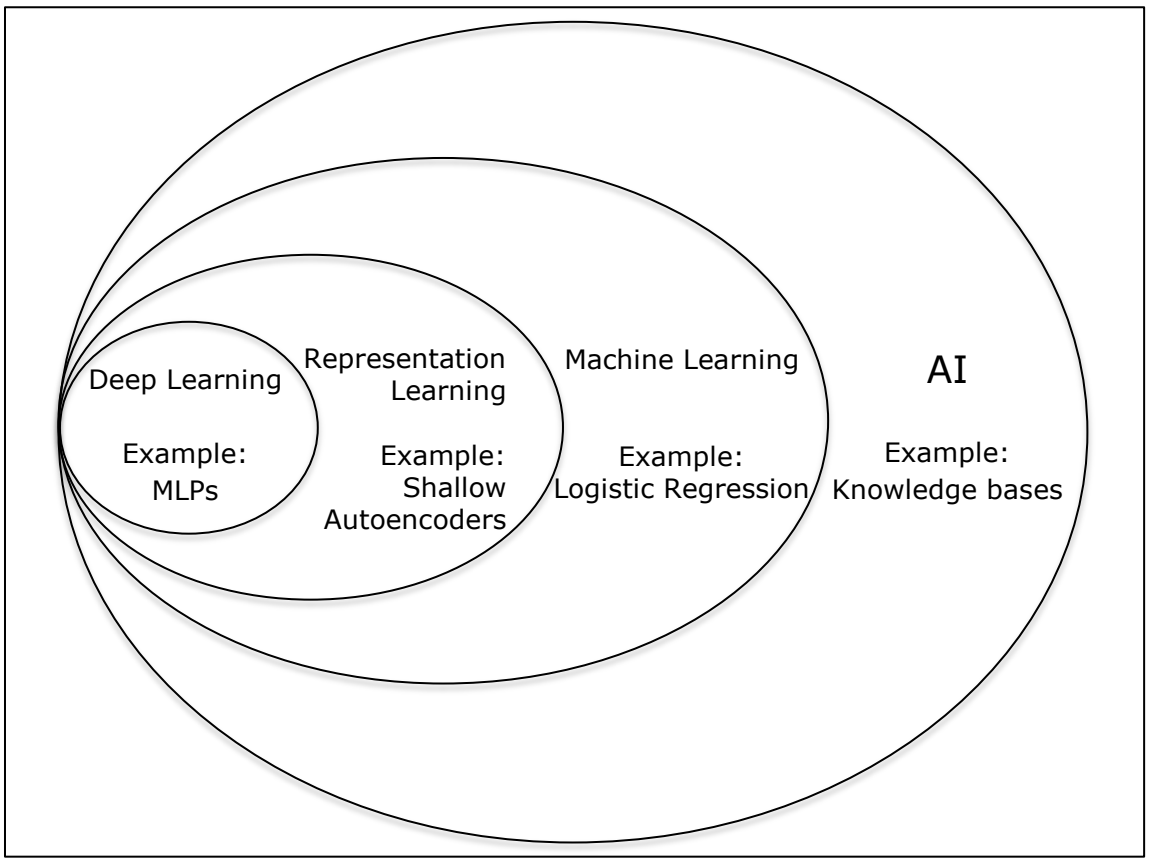


Figure 2.2 Approaches of AI [23]

In order to behave in an intelligent way, computers need to capture both formal and informal knowledge, which people are faced within their daily life. However, due to its intuitive and subjective form of informal knowledge, it is really hard to describe it in formal ways to make the computer reason on the given knowledge. Knowledge based approaches, which are kind of AI, use logical inference rules to catch the informal knowledge. Low success of using this method led researchers to find another approach known as machine learning with which computers acquire their own knowledge by finding patterns from raw data. Logistic regression is one of the algorithms that can be given as an example to machine learning. It is used when the outcome has two classes or is categorical. Another type of machine learning algorithms is known as "representation learning" which represents the input data into featured set of data to solve tasks. Autoencoders are good examples of this type which are the combinations of encoders accompanied with their decoders. An encoder converts the input data into different representations for example with feature extraction then dimension reduction which increases the performance of the computer while handling this information; and its

decoder converts the represented data back into its original format. However, finding the correct features to represent the complex task is a great hurdle. To accomplish this difficulty, deep learning algorithms are used which use simple concepts to build complex concepts. A good example may be given as the feedforward deep network or multilayer perception (MLP) which is composed of different and sequential simpler mathematical functions to represent the data. With nested hierarchy of concepts in deep learning, more abstract representations can be executed in terms of less abstract ones that allows computer systems to gain experience with data [23].

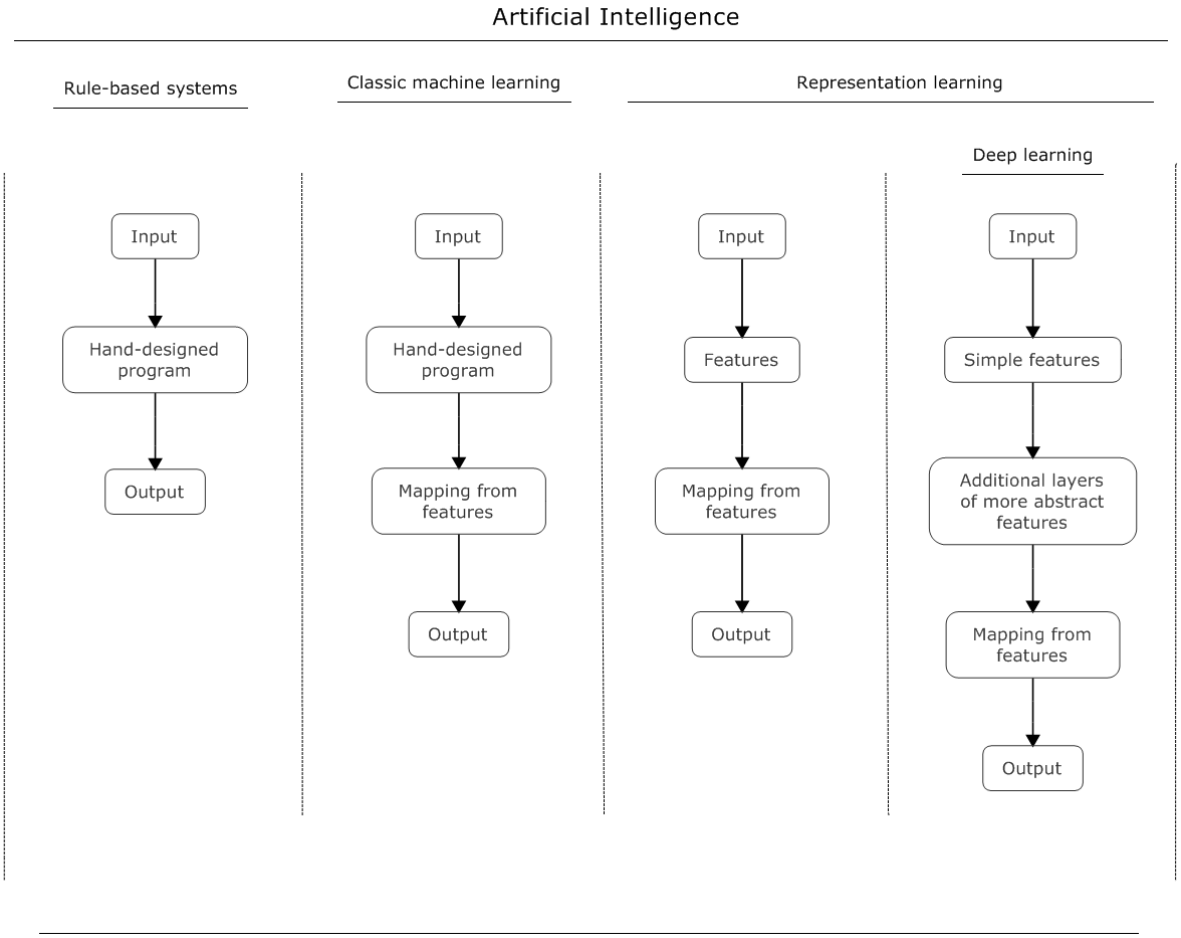


Figure 2.3 Relations of components for different AI systems [23]

2.1.2. History of Artificial Intelligence

Like stating the beginning of history of flight/aviation, the beginning of the history of artificial intelligence is hard to say with pinpoints accuracy. Many researchers in different areas -like biology and mathematics- contribute to this discovery [24].

Warren McCulloch and Walter Pitts efforts to investigate the first artificial neuron in 1943, which is based on simple threshold logic, might be treated as one of the main breaking points. This computational model, which is in fact inspired from the functions of brain neurons, might be the first concrete step for the Artificial Neural Networks [25].

This investigation encouraged the researches to work on computational intelligence further.

If a neuron could be imitated why could not the intelligence? Alan Turing opened a path with his successful attempt to test intelligent behaviour of a computer with the Turing Test [26]. In this test a computer and a person answer the questions of an interrogator as being in separate rooms, and the interrogator tries to distinguish which one is a computer which one is a human. If the interrogator fails to distinguish the computer as, whether it is a human or not, then the computer passes the test with its ability to mimic cognitive tasks of a human [27].

The term 'Artificial Intelligence' was first introduced by John McCarthy. With his deep interest to common sense reasoning, he proposed a program called Advice Taker, which solves problems by manipulating sentences in formal languages. His objective was to introduce the method of representing information by logic in computer memory, since he believed that AI cannot be thought without logic, i.e. it is unavoidable for AI. Furthermore, he stated his objective as making programmes that learn from their experiences [28], which can be seen as the basics of today's Knowledge Based Methods. He also presented the high level programming language LISP in the late 1950s. He used Symbolic Expressions to represent Symbolic Functions and applied the universal Symbolic Function for the theoretical role of a universal Turing Machine and the practical role of the interpreter [29].

In 1964 Danny Bobrow demonstrated in his Ph.D. thesis that a computer can solve algebra word problems with interpreting natural language. The problem solving system that he used finds the set of kernel sentences and converts them to a set of simultaneous equations, then tries to solve this set of equations for the values of requested unknowns. If it is able to solve then it gives the answer in English, if the inputs are not enough to solve, it asks for more information with indicating the nature of the required information [30], [31]. This problem solving system that is shown in Bobrow's thesis was the first step to communicate with computers through natural language. Further, in 1965 Joseph Weizenbaum built an interactive problem solver, ELIZA, which can carry on a dialog in English. The first robot that can reason with natural language processing and accordingly take physical action was Shakey, which was developed at Stanford Research Institute around 1969. After that, in 1973 at the University of Edinburgh an experimental robot, Freddy, was produced which uses sensors and video camera to recognise the objects then take actions. In 1979, Stanford Cart was used autonomously which was then known as the first computer controlled autonomous vehicle. With these successful attempts in AI, researchers were encouraged to continue developing algorithms in AI and they made many contributions like case-based reasoning, multi-agent planning, scheduling, data mining and virtual reality through 90's. When Garry Kasparov, the world chess champion, was beaten by IBM's Deep Blue Chess Program, artificial intelligence became popular not just among the researchers but all over the world population in 1997. Then, with the famous robot Kismet that was built in MIT in 2000 and could express emotions with its artificial face, vocal and motor capabilities, the era was opened for building commercially available interactive robots [31].

2.1.3. Artificial Intelligence in Aerospace

Artificial intelligence, especially the knowledge-based methods and neural networks have been used more than four decades in different branches of aerospace environment, when Neural Networks is considered as a base and method for doing nonlinear regression [32]. In the study by Gallaher et al. [33], least-squares regression approximations were used to obtain predictor information to find out the future airplane position and orientation, may be given as a first example. In this study, the standard method for obtaining the predictor information, which uses complete fast time model of the aircraft, was compared with an alternative approach that uses just thirteen predictor variables representing changes in positons and rates of change of positions for determining the six degrees of freedom of aircraft motion. The specific task was approach-to-landing task for a general, light, twin-engine aircraft, a Singer-Link General Aviation Trainer (GAT-2). High multiple correlation coefficients obtained to assess the goodness of fit of each of the regression equations indicate that the used regression approach produces very accurate prediction equations and is an alternative to using the complete fast-time model [33].

In the study done by Kroo and Takai [34] an aircraft design program is combined with a rule-based advice and warning system. Aircraft design program includes modules for calculation of aerodynamics, structures, propulsion, and operating costs. The subroutines and order of execution are selected based on human experience in an appropriate order, when the related result is required or currently updated for the computation. The rule-based expert system is used to assist the user in selecting intelligent design solutions and appropriate analysis procedures with accessing the simple set of IF-THEN rules which can be depicted as run-time knowledge activation. After each routine the database is examined and the problems are identified, then the user is informed with literal expressions like "The nose fitness is too low for this Mach number" and the expert advice system is activated with this input. Afterwards, the second part of the knowledge-based system advices a solution like "reduce Mach number or increase nose length". The expert advice system uses a forward and backward chaining inference engine, which uses fuzzy reasoning (reasoning with uncertainty), to diagnose the problems, posting the warning strings and making the suggestions. The knowledge-based system with its expert warnings and advice utility is proved as a helpful tool to understanding the design changes with their effects and as a useful debugging tool for further analysis in aircraft advanced design [34].

Intelligent wing design support system defined by Takasu et al. [35] was a mile stone for automatic design systems. In this system, they built the hierarchically constructed wing design support system by Multi-Layer Logic (MLL) [36] [37] data structure and Knowledge Acquisition and Utilization System (KAUS) [38]. They used mainly two sub-processes for wing design: aerodynamic design which includes wing section design and three-dimensional design, and structural design that consists of spar, rib and skin designs. First, they represented a wing model hierarchically with MLL data structures in order to build and modify the structural model with knowledge at the later stages of the design. Then, KAUS is used for knowledge processing that is configured with a knowledge base, a procedure base and a database parts. Eventually, the system extracts the results of structural analysis, finds the weak points and suggests modifications accordingly. With this work, it is shown that AI technology is very helpful while developing wing design support system [35].

One of the nonlinear methods introduced by Dovi and Wrenn [39] is Envelope Function Formulation [40], which is also adaptable for multiobjective optimization problems [41]. A typical wide body transport aircraft with 256 passengers was used to investigate the benefit of Envelope Function Formulation over two methods, the Penalty Function and the Global Criterion methods. The technique converts a constraint optimization problem to an unconstraint one by replacing the constraint and objective function boundaries in n-dimensional space with a single surface. A complex mission performance analysis was done with four design variables: aspect ratio; area, quarter chord sweep and thickness to chord ratio of the wing. Ramp weight, mission fuel, lift to drag ratio at constant cruise Mach number and range with fix ramp weight were selected as the objectives. First two objectives are subjected to minimize and other two are to maximize. Four primary modules were used in conceptual design: weight, aerodynamics, mission performance and takeoff and landing.

All of the used methods were feasible within the design space. Function evaluations were different for each case, and could not be said one of them is better than others. However, this study is a good example for showing the changing success rate of optimization problems with different cases. It is highlighted that, while evaluating the performance of a method with its computational efficiency, the ease of use, data requirement and programming should also be considered, since they are the prior attributes of the methods [39].

Artificial Neural Networks was used to evaluate the transportation engineering predictions by Faghri and Hua [42]. A case study of trip generation forecasting was done using one traditional method, regression analysis, and two Artificial Neural Networks methods, back propagation and adaptive linear element (ADALINE). Regression analysis was selected since it is widely used in that area with the category analysis. The difference between the regression analysis and the ADALINE depends on the way of using the coefficients and weights in the optimization. For the regression method it is aimed to find the minimum error for the survey data, which can be considered as the training data sets for the ADALINE, while for the ADALINE it is to find the best value of the weights that obtain good results for testing the data. As a result, they have found out that Artificial Neural Networks models perform better than the linear regression analysis for that design case. When comparing the two Artificial Neural Networks methods, the result of ADALINE is slightly better than back propagation method. On the other hand it needs four-time more iterations than back propagation, which are 10,000 iterations to minimize the errors on the testing data sets in the training. That means the training of back propagation model is much better than the ADALINE [42].

2.2. Knowledge Based Methods

In the Knowledge Based type artificial intelligence algorithms reasoning, perception and acting are performed by the *agents*. An agent acts in an *environment*, in which other agents can act also, works together or against. A robot can be an illustrative example of an agent which has sensors and actuators to perceive its physical environment. An agent could be a computer program with attributes like vectors that acts in a computer environment only, which is called as a software agent. These attributes can be defined individually according to the desired resultant action. Figure 2.4 illustrates an agent with

its inputs and outputs. Two deterministic agents with the same inputs should act in the same way [24].

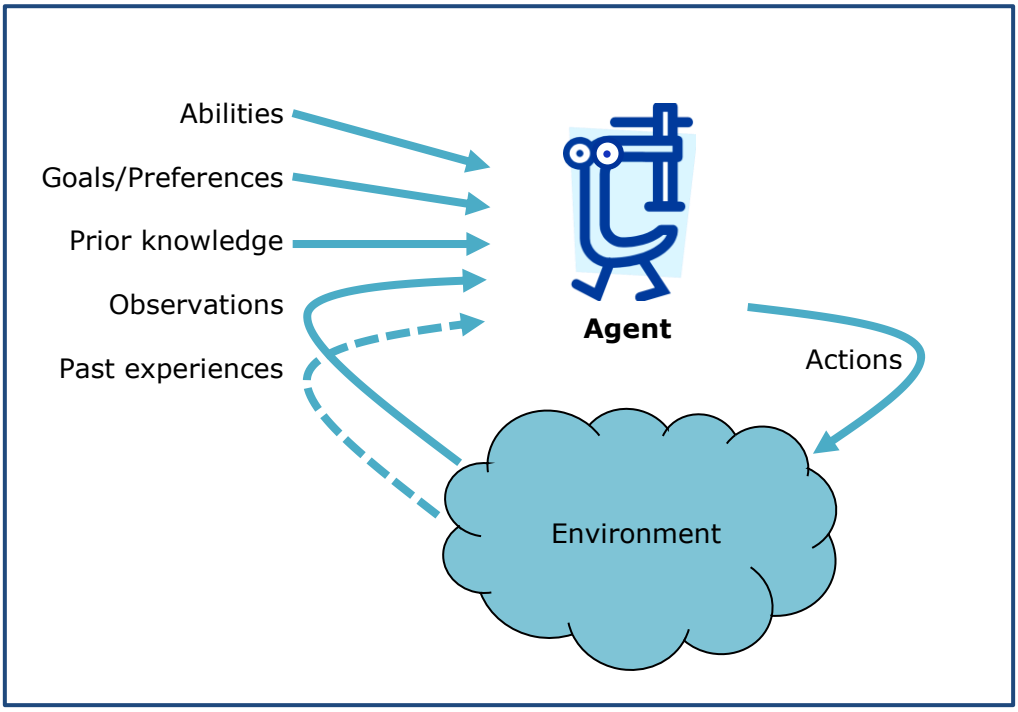


Figure 2.4 An agent interacting with an environment [24]

Agents can act in different environments. The complexity of the environment can be defined with nine dimensions: modularity, representation scheme, planning horizon, sensing uncertainty, effect uncertainty, preference, number of agents, learning and computational limits. These dimensions are used to present a *design space* of artificial intelligence. Different design points can be obtained by changing the values or the characteristics of those dimensions. More design choices must be add in order to build intelligent agents [24].

Design spaces are best defined with individuals and relations. Agents must build probabilistic models before they learn about individuals, or unsystematically distributed variables. After learning the probabilities, the probabilities do not depend on the individuals, but the agent can learn also the new individuals. A *probabilistic relational model (PRM)* is a model in which the probabilities are defined with the relations not with the actual individuals. The probability parameters are shared by different individuals [24]. In this thesis, a kind of probabilistic relational model is used.

2.3. Computational Intelligence

In Knowledge Based Systems, reasoning capability of intelligence is tried to be mimicked and utilized to reach the solution. This is done by extracting the information from the data bases. On the other hand, methods of computational intelligence employed

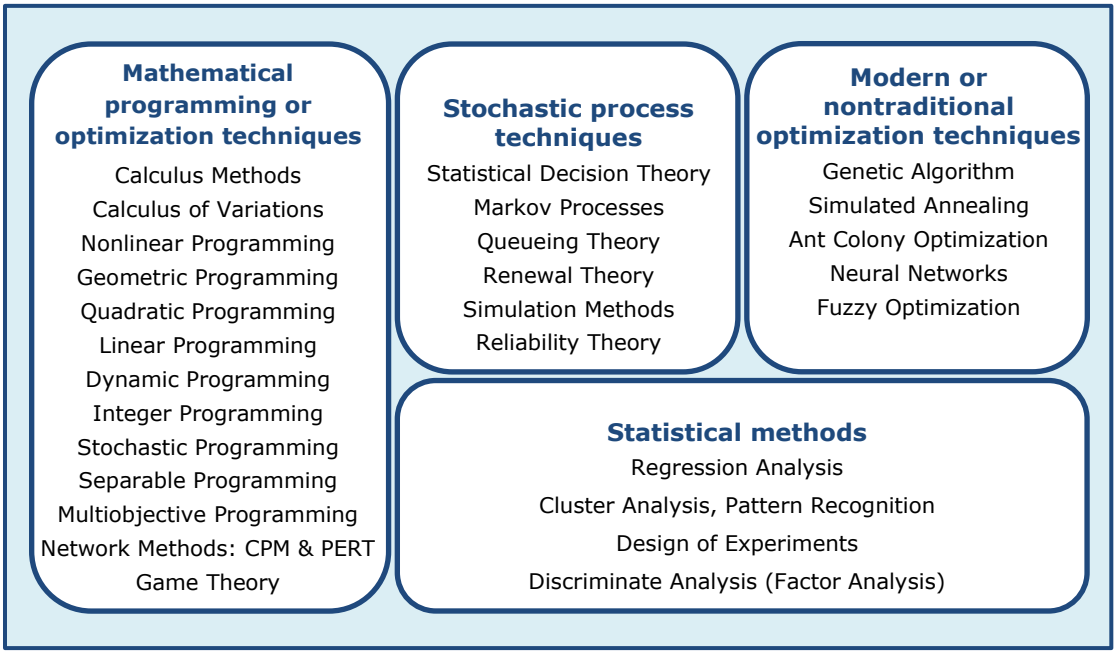
in optimization problems are more structured and use the behavioural properties of the natural objects.

Most of these techniques require the function values but not the derivatives as in the gradient descent algorithms. Genetic Algorithm and Simulated Annealing are stochastic methods and are used for discrete optimization problems. *Genetic Algorithm* uses the natural selection and genetics principles, whereas *Simulated Annealing* is based on the simulation of thermal annealing of critically heated solids to find the global minimum. The *particle swarm optimization* is inspired by the behaviours of colonies like birds, insects and fish. In *ant colony optimization* the behaviour of the ant colonies which try to find the shortest path from a nest to a food source is imitated. If the information at hand cannot be clearly stated or is not discrete then *fuzzy optimization methods* are used [43].

Especially for highly nonlinear problems the *Neural Networks based methods* are used, in which the conduction of information in neurons are tried to be artificially simulated. However, the imitated procedure does not even approach being similar to that in the natural neural networks, that uses synapses of neurons to transmit the electrical and also chemical signals in order to communicate [43].

The more common techniques are listed and grouped on Table 2.1 by Rao [43]; however these techniques can be clustered under different branches based on their various characteristics.

Table 2.1 Methods of Operations Research [43]



In this thesis, among the modern optimization techniques listed in Table 2.1, Genetic Algorithm and Simulated Annealing are used to compare and verify the results of two design cases in aerospace, because of their high application along with their ease of usage and success in the field. Besides, the advantage of probability theory and dimension reduction techniques were also taken.

Optimization methods can be categorized into two classes also, depending on the availability of using constraints: if a constraint function exists it is named as "*constrained method"*, otherwise it is named as "*unconstrained method"*.

When the optimization technique does not need the partial derivatives of the objective function it is called *nongradient method* or *zeroth-order method*. *Direct search methods* are nongradient methods. On the other hand, in some cases the first and second partial derivatives provide more information and help to improve the results. *Descent methods* are the methods which use those derivatives, thus they are known as *gradient methods*. All the unconstrained minimization methods are iterative and have different rate of convergence [43]. Table 2.2 includes more common unconstrained minimization techniques.

Table 2.2 Unconstrained Minimization Methods [43]

Direct search methods[a]	**Descent methods[b]**
Random search method	Steepest descent (Cauchy) method
Grid search method	Fletcher-Reeves method
Univariate method	Newton's method
Pattern search methods	Marquardt method
Powell's method	Quasi-Newton methods
	Davidon-Fletcher-Powell method
	Broyden-Fletcher-Goldfarb-Shanno method
Simplex method	

[a] *Do not require the derivatives of the function*
[b] *Require the derivatives of the function*

Constrained methods can be clustered into two broad categories: Direct methods which use constraints explicitly, and indirect methods which use constraints implicitly as a sequence of unconstrained minimization problem [43]. Some of the constrained optimization techniques are listed at Table 2.3.

Table 2.3 Constrained Optimization Techniques [43]

Direct search methods	**Indirect search methods**
Random search method	Transformation of variables technique
Heuristic search method	Sequential unconstrained minimization techniques
Complex method	*Interior penalty function method*
Objective and constraint approximation methods	*Exterior penalty function method*
Sequential linear programming method	*Augmented Lagrange multiplier method*
Sequential quadratic programming method	
Methods of feasible directions	
Zoutendijk's method	
Rosen's gradient projection method	
Generalized reduced gradient method	

According to the Table 2.2 and Table 2.3, the algorithm developed in this thesis falls into the group of Direct Search Methods, to which the pattern search methods belong and which do not require the derivatives of the function.

Besides, there are systematic methods for constrained optimization: pruning of dominated assignments, domain splitting and variable elimination. These techniques can simplify the problem but do not always solve the problem [24].

Ok what is the conclusion of this section? What are the pros and cons? What is a deficiency, which can be resolved by your scientific contribution?

In this section, the extensively used techniques are listed with different perspectives in groups to have better understanding of the way that they function. Accordingly, the technique developed in this thesis employs one of the modern optimization methods "Probabilistic Neural Networks"; which uses the benefit of probability theory while clustering the instant data. Assigning the candidate design points to the clusters according to their probability of being poor or good designs saves the computational effort and it is one of the advantages of the algorithm. Also, the used distinctive dimension reduction technique brings advantage for the optimization process. So thus, the pure unsystematical walk technique is replaced by a less stochastic method but while taking the advantage of probabilities. This eases to reach the objective with less number of function evaluations.

In the next chapters, some popular techniques are explained more in detail.

2.3.1. Genetic Algorithm

Evolutionary algorithms together with Simulated Annealing are types of guided random search techniques. The most famous algorithm is the "Genetic Algorithm" among the evolutionary algorithms. It uses reproduction, crossover and mutation properties of chromosomes while treating each value of a design variable as one of the chromosomes in natural genetics and representing it with set of binary numbers. For example, mutation can be defined as the stochastic perturbation of the values to gain new variable values in a population. In Genetic Algorithm instead of derivatives, the value of the objective function is used for the further search processing. Since it uses a population of trial points, it differs from the Simulated Annealing which has one starting point. Because of this property, Genetic Algorithm has advantages on other methods while finding the global optimum on nonconvex and nonlinear design spaces [43]. However, when the number of variables is increased, the calculation time rises and the execution of the process can last even days or weeks regardless of the design surface uniformity.

Genetic Algorithm has been successfully applied to optimization problems in engineering design and transportation problems for several years. It can be used discontinuous and non-differentiable problems because of its use of stochastic information. It is used more for discrete problems than continuous problems. Unlike the methods which use search direction Genetic Algorithm uses population, and this causes to miss any adjustments for the neighbouring solutions [44].

In a Genetic Algorithm each new element of the population is a combination of a selected pair. The new element or the offspring has some variable values from one of the

parent and the rest from the other parent. This operation is called *crossover*, and the most common crossover method is *one-point crossover* in which this transition process occurs at an unsystematical index. This crossover process occurs until reaching the population number [24].

The chromosomes are selected for the next generation according to their fitness values. However, when the fittest chromosomes have too much priority this can cause the reduction in the diversity and early convergence to points that are not globally optimal. Fitness proportionate selection is one of the most popular selection methods [45].

The selection operator is also called *reproduction* operator and selects the good strings of the population, that have higher fitness values. Thus, if F_i is the fitness of the i^{th} chromosome (or string) in the population of size n, the probability of selecting the i^{th} string for the mating pool (p_i) is given by Rao [43]:

$$p_i = \frac{F_i}{\sum_{j=1}^{n} F_j}; \quad i = 1, 2, \dots, n \tag{2.1}$$

When $\bar{F}$ represents the average fitness of the population:

$$\bar{F} = \frac{1}{n}\sum_{j=1}^{n} F_j \tag{2.2}$$

Then the cumulative probability of the string i is:

$$P_i = \sum_{j=1}^{i} p_j \tag{2.3}$$

When the equations (2.1) and (2.3) are considered together, it is expected that the string with a higher fitness value is selected more times than others strings if it has a larger range of cumulative probability. So that, with the reproduction operation the string with a higher fitness value will be copied directly to the next mating pool more frequently [43].

Another operator of the Genetic Algorithm is *mutation*. With this, one or more unsystematically selected variables are altered. The mutation operation helps to overcome the convergence to the local minimums. The resulting chromosomes are placed in the new population [13]. Genetic Algorithm combines uphill tendency with stochastic perturbation exploration, but the effectivity of the algorithm mostly depends on the crossover operation [46].

Spall [45] explained step by step the core Genetic Algorithm as the following:

> *Step 0:* **Initialization** Generate randomly an initial n number of population and evaluate the fitness function for each string.
>
> *Step 1:* **Parent selection** Select the strings which have higher fitness values; higher fitness value means more often being selected as parents
>
> *Step 2:* **Crossover** Perform crossover for each pair of parents at a randomly selected splice point or points with the decided crossover probability.
>
> *Step 3:* **Replacement and mutation** Alter the individual bits with the decided mutation probability.
>
> *Step 4:* **Fitness and end test** Compute the fitness values of the new population of strings. Stop the process if the termination criterion is met otherwise go to *Step 1*.

Figure 2.5 Core Genetic Algorithm [45]

In this thesis, the unsystematical starting points and the binary coding are also used for the optimization process as in the Genetic Algorithm, since the introduced lower and upper bounds require starting points and the binary coding is sufficient to extract the targeted information from the intervals. However, instead of using crossover or mutation on the variables while depending on the value of the objective function, the hybrid technique uses the successful changes on the objective function and the corresponding direction sets of the variables on the particular intervals. Then, these direction sets are classified and used simultaneously (as a training set) in the Probabilistic Neural Networks to decide on the next probable directions and intervals. While doing that, instead of unsystematical or instant crossover/mutation the knowledge gathered from the training set is used. When the number of variables increases the string length (or, the number of digits) also increases. Besides, it was proved with a trend analysis that the Probabilistic Neural Networks becomes more conservative to any change on the string and eliminate the noises. This provides determination on the successful routes while progressing onto the untried but more probable intervals.

2.3.2. Simulated Annealing

The Simulated Annealing is such an algorithm that combines the efficiency of hill climbing and the completeness of unsystematical walk algorithms. In other words, instead of using the best move (the best neighbour point), it uses an unsystematical move. This unsystematical move supplies the algorithm the ability to escape from a local minimum in general and improves the completeness of the search. On the other hand, this unsystematical move can be time and calculation power consuming. At that point, this con inspires a novel technique like the one explained in this thesis to overcome substantially this loss. Besides, Simulated Annealing is popular for large scale

optimization tasks and has been used widely for factory scheduling [46]. It is well known as a global optimizer for both discrete and continuous optimization problems [45].

The principle behind the Simulated Annealing is to use the behaviour of substances as they cool as they are processed which were examined with scientific applications. However, while they are cooling, the temperature is not the only determining property of the demonstrated behaviour, the rate of cooling while reaching the lower energy state is important and must be slow. While doing analogy with an optimization problem, a minimized value of the loss function is matched with the minimum energy state for a system. If the system is cooled so rapidly then the reached state may not be the state that has the minimum energy state as in the rapid substance cooling, or polycrystalline. So, annealing is defined the process of cooling at a slow rate. As in the physical cooling process Simulated Annealing has temporary energy increases in the loss function [45]. Because of its discrete nature, Simulated Annealing is not affected by the continuity or differentiability of the functions. Despite the increase in computational effort, the convexity status of the feasible space does not affect the convergence [43]. In order to terminate the process, as in the Genetic Algorithm a convergence criteria should be defined in the program. This may be the minimum change in the temperature or loss function or number of function evaluations.

Venkataraman [44] summarized the basic steps of the Simulated Annealing Algorithm as:

Step 1: Choose a starting design point X_0. Calculate $f_0 = f(X_0)$. (Need stopping criteria)

Step 2: Choose a random point on the surface of a unit n-dimensional hypersphere to establish a search direction S.

Step 3: Using a step size α, calculate $f_1 = f(x_0 + \alpha S)$; and $\Delta f = f_1 - f_0$

Step 4: If $\Delta f \leq 0$; then $p = 1$, else $p = e^{-\beta \Delta f}$

Step 5: A random number r, $0 \leq r < 1$, is generated. If: $r \leq p$, then the step is accepted and the design vector is updated. Else: no change is made to the design. Go to *Step 1*.

Figure 2.6 Simulated Annealing [44]

A reasonably large number of iterations is needed to reach the global optimum. The two parameters α and β are problem dependent; and generally $\beta = -k/T$, which is the Boltzmann probability distribution. T is the annealing temperature, and k is the Boltzmann constant [44]. A configuration which has higher energy is selected based on the energy difference Δf. The algorithm starts with a high value for T, then it is reduced with each iteration. This reduction operation on temperature is called as "cooling". The probability of approaching to the higher energy states is higher initially, but at the end of the process the energy state approaches to zero [47]. A worse solution can also be accepted due to the conditional probability; as a result the "hill climbing" can be visible at those stages. The value of p is proposed to be $0.5 \leq p \leq 0.9$ [44].

Indeed, Simulated Annealing picks a neighbour point unsystematically and then selects it if there is an improvement on the objective function. If there is no improvement, it accepts or rejects depending on the parameters above. Thus, Simulated Annealing requires a good annealing schedule for a better convergence to the global optimum. Otherwise, the high number of iterations to escape from a local optimum is discarded and the number of function evaluations highly increases. "Finding a good annealing schedule is an art", [24].

Actually, there are variations of Simulated Annealing algorithm depending on the sampling method to generate a new candidate point and the implementation of the annealing scheduling [45].

Some of the properties of the Simulated Annealing algorithm are listed by Rao [43] as following:

- The computational effort may increase with a worse initial starting point, but the final solution is not affected.
- The continuity or the differentiability of the functions does not affect the transition or convergence characteristics of the algorithm; that is due the fact that the constraint evaluation and the discrete nature of the algorithm.
- The convexity of the search space has no influence on the convergence.
- The design variables can have values other than positive also.
- The algorithm is applicable to solve mixed-integer, discrete or continuous problems.
- As in the case of Genetic Algorithm, an equivalent unconstraint function can be used for the problems that involve behaviour constraints.

Thus, the main deficiency of the Simulated Annealing algorithm is that the selection of the starting point may increase the computational effort and the run time, accordingly. This drawback may be diminished with coupling it with a method like any surrogate model. Actually, the idea in this thesis to develop a method, that works like an unconventional surrogate model which works with patterns.

Table 2.4 Comparison of Random search and Simulated Annealing [45]

N	Random Search		Simulated Annealing		
	Localized Random Search	Enhanced Localized Random Search	Initial T=0.01	Initial T=0.10	Initial T=1.00
100	0.00053	0.328	1.86	0.091	0.763
1000	2.8×10^{-5}	1.1×10^{-5}	0.0092	0.067	0.506
10000	2.7×10^{-6}	2.5×10^{-7}	0.00038	0.0024	0.018

Two random search techniques, *Localized Random Search* and *Enhanced Localized Random Search*, are compared with Simulated Annealing Algorithm for its performance by Spall in [45]. The case is the simple quartic polynomial loss function:

$$f(x) = x_1^4 + x_1^2 + x_1 x_2 + x_2^2 \tag{2.4}$$

To generate the variables an unsystematical distribution is used for the both random algorithms and the Simulated Annealing algorithm. To increase the performance of the Simulated Annealing algorithm some efforts were made like using a standard temperature decay factor, 0.98. Table 2.4 shows the results of the comparison for different number of function evaluations, N, and different initial temperatures, T, for Simulated Annealing. Even though the results are improved with increasing function evaluations, random search algorithms find better results than Simulated Annealing. These results prove that the Simulated Annealing algorithm is not always superior to the simple random search algorithms [45] and needs tuning for any improvement.

The developed algorithmic approach in this thesis can contribute solving this deficiency by not using the creeping or the pure unsystematical search, but by handling the simultaneous results to explore the patterns of the whole design space. Actually, the developed algorithm uses the information extracted from the previous iterations to map the whole design space and reduces the poor candidate data points which may be handled if stochastically perturbed values are just used. This learning process can ease to reach the global optimum with less number of function evaluations and save time. The detailed explanation is found at section 3 Cavus Algorithm.

2.3.3. Pattern Search

Pattern search methods are the methods that use pattern directions as the search directions [43]. In the *Univariate method*, the design variables have a coordinate and each direction is considered as a search direction. A unit vector is used to present the direction on the coordinate. The search is done through each variable orderly on those search directions [44]. In other words, the minimum is searched along the directions parallel to the coordinate axes in the univariate method [43]. Each cycle has iterations for each directions of a set of variables. This method is also known as *Cyclic Coordinate Descent method*, and has a minor difference from *Pattern Search* method. Whereas Univariate method has a zigzag movement while approaching the solution, Pattern search method has an additional iteration for each cycle to improve the process. At this additional iteration the previous search directions and the optimal value of the stepsize for that direction are summed. The next cycle of iteration begins after one-dimensional optimal stepsize is calculated [44].

A series of exploratory moves around the current iterate and updating the current iterate with associated information before selecting a new iteration point are the bases of the pattern search methods. These exploratory moves have two requirements in any particular pattern search method to maintain the properties for the convergence [17] :

1. At iteration k, the direction and the length of step s_k are determined by the pattern P_k and by the step length parameter Δ_k , respectively.
2. If any of the $2n$ trial steps has a function value less than the current iterate then the simple decrease on the function must be found by a step s_k which is produced by these exploratory moves.

Pseudo code of the Pattern Search Algorithm is given by Venkataraman [44] as the following:

Step 0: Choose starting point X_1, and N_c (number of cycles)

$f_c(1) = f(X_1)$; $X_c(1) = X_1$

$\varepsilon_1, \varepsilon_2$: tolerances for stopping criteria

Set $j = 1$ (initialize cycle count)

Step 1: For each cycle j

For $i = 1, n$ (number of variables)

$S_i = \hat{e}_i$ (assign univariate step)

$X_{i+1} = X_i + \alpha_i S_i$, α_i is determined by minimizing $f(X_{i+1})$

End of For loop

$S_j = \sum_{i=1}^{n} \alpha_i^* S_i \equiv X_{n+1} - X_1$ (Pattern step)

$X_j = X_{n+1} + \alpha_j S_j$ (best stepsize)

$X_c(j+1) \leftarrow X_j$; $f_c(j+1) = f(X_j)$ (store cycle values)

Step 2: $\Delta f = f_c(j+1) - f_c(j)$; $\Delta X = X_c(j+1) - X_c(j)$

$If\ |\Delta f| \leq \varepsilon_1\ ;\ stop$

$If\ \Delta X^T \Delta X \leq \varepsilon_2\ ;\ stop$

$If\ j = N_c\ ;\ stop$

$X_1 \leftarrow X_j$; $f(X_1) \leftarrow f(X_j)$

$j \leftarrow j + 1$

$Go\ to$ *Step 1*

Figure 2.7 Pseudo code of the Pattern Search Algorithm [44]

There are many kinds of developed pattern search methods. The *coordinate search* may be the simplest one among the others [17].

If x_k is the current iterate, at iteration k the trial point is defined as [17]:

$$x_k^i = x_k + s_k^i \tag{2.5}$$

Figure 2.8 illustrates all the possible actions during the exploratory movements of the coordinate search method. The black solid circles indicate the successful movements where the objective function decreases. Whereas, the open or empty circles indicate the evaluated functions but unperformed steps due to lack of improvement on the function value. So the progress is gained when $f(x_{k+1}) < f(x_k)$ [17].

If after $2n$ evaluations ($x_k^1,\ x_k^{1'},\ x_k^2,\ x_k^{2'}$), no decrease is observed on the function value at the current step x_k, then $x_k = x_{k+1}$ and the stepsize s_k are reduced for the next iteration. This case is shown in Figure 2.8 as the last scenario [17].

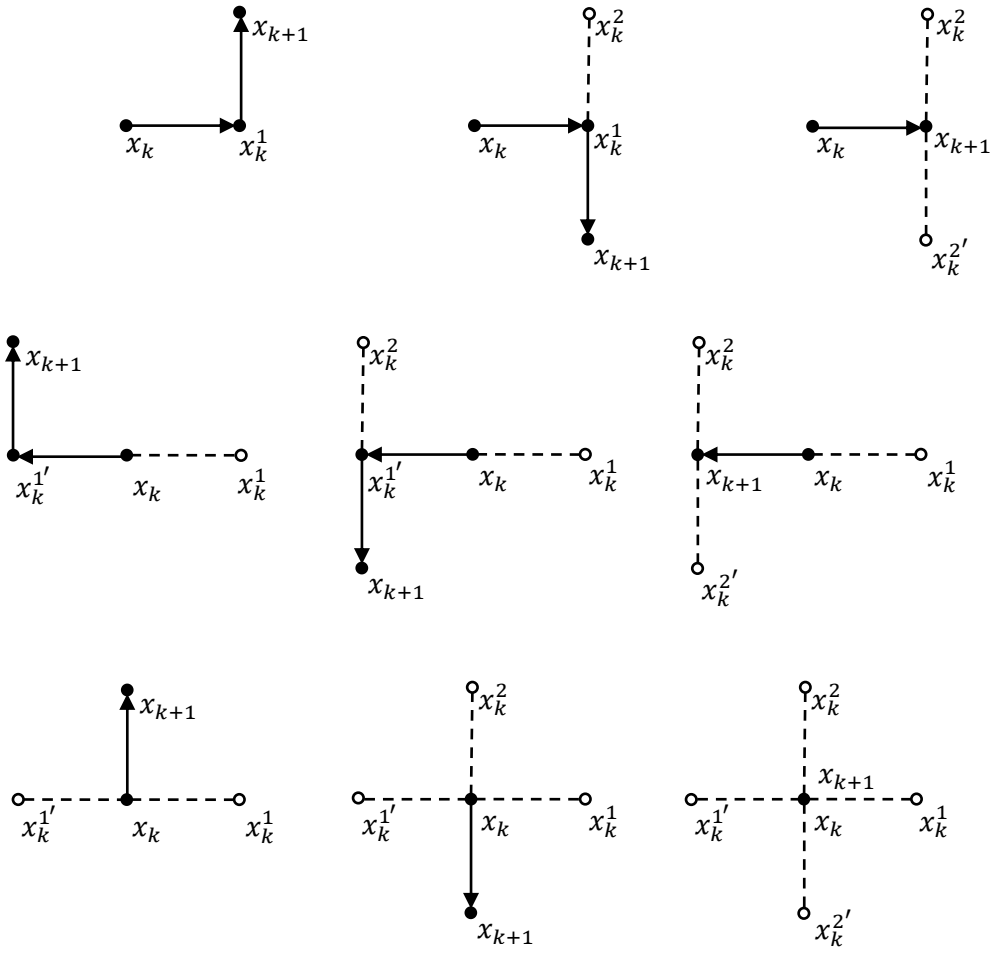


Figure 2.8 All the possible subsets of the steps for coordinate search in R^2 [17]

All of these possible subsets can be gathered in a generating matrix $C_k = C$ with representing all the possible combinations of $\{-1, 0, 1\}$. Indeed, C has $p = 3^n$ columns and for $n = 2$ [17]:

$$C = \begin{bmatrix} 1 & 0 & -1 & 0 & 1 & 1 & -1 & -1 & 0 \\ 0 & 1 & 0 & -1 & 1 & -1 & -1 & 1 & 0 \end{bmatrix}$$

Thus for a given step length Δ_k, all the possible trial points can be seen in Figure 2.9 [17].

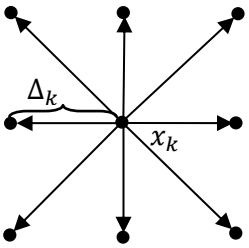


Figure 2.9 The pattern in R^2 with a step length control parameter Δ_k [17]

Figure 2.9 includes all the possible trial points shown in separate successive movements in Figure 2.8 [17].

In [17] the convergence analysis of the pattern search algorithm is done by Torczon and described further in detail. They have concluded that, pattern search methods are descent methods and gradient related methods. They also proved that, due to inadequate step length control mechanism the search does not terminate before convergence. Not permitting arbitrary step lengths along arbitrary search directions is the key characteristic of the method and the base of its success. Moreover, it is robustness against its proponents is also demonstrated and as a result they comment on the prospective further improvement capabilities with guarantied global convergence. Hence, it has been done many researches on this topic since then [48].

2.3.4. Kriging

Kriging is a kind of regression method which uses interpolations governed by Gaussian process and is originally used in geostatistics. It is used to estimate the value of a function by interpolating the values of the neighbourhood data points. Due to the fact that, it scans throughout the design space and predicts the intermediate function values with the values of previously derived function values, it is used in spatial analysis. To estimate the function value at a new point, the neighbouring function values are combined with weights. For that reason, the new point is compared with the known data points to find out these weights.

This technique is first developed by Krige [49] and then applied to the computer experiments as an approximation technique by Sacks et al. [50] and to geostatistics by Matheron [51].

Sacks et al. [50] summarized the method as the following:

With a given design, S, where $S = \{s_1, \cdots, s_n\}$, and data $y_s = \{y_{s_1}, \cdots, y_{s_n}\}'$, then the linear predictor of $y(x)$ at an untried x becomes $\hat{y}(x) = c'(x)y_s$.

$Y_s = [Y(s_1), \cdots, Y(s_n)]'$, the corresponding so-called random quantity is used in place of y_s.

y_x is predicted with a Bayesian approach:

$$\hat{y}(x) = E[Y(x)|\, y_s]$$

(2.6)

The best linear unbiased predictor of the response at an untried input is:

$$f(x) = [f_1(x), \cdots, f_k(x)]'$$

(2.7)

For the k functions in regression:

$$F = \begin{pmatrix} f'(s_1) \\ \vdots \\ f'(s_n) \end{pmatrix}$$

(2.8)

The $n\ x\ k$ expanded design matrix:

$$R = \{R(s_i, s_j)\}, \quad 1 \le i \le n; 1 \le j \le n$$

(2.9)

Where,

$$r(x) = [R(s_1, x), \cdots, R(s_n, x)]'$$

(2.10)

and the unbiasedness constraint is $F'c(x) = f(x)$.

The best linear unbiased predictor can be written by resolving the partitioned matrix as:

$$\hat{y}(x) = f'(x)\hat{\beta} + r'(x)R^{-1}(Y_s - F\hat{\beta})$$

(2.11)

Where $\hat{\beta} = (F'R^{-1}F)^{-1}F'R^{-1}Y_s$ is the usual generalized least squares estimate of β.

At a new location x_p to estimate the unknown value $\hat{y}$, all n sample points are used with a weighted linear combination as in the following equation [8]:

$$\hat{y}(x_p) = \sum_{i=1}^{n} w_i(x_p)\, y(x_i)$$

(2.12)

w_i is the weighting factor and it is changing throughout the design space as a function of x_p. To correlate the two points x_i and x_j different methods can be used. In order to describe the Kriging as simple as possible, Gaussian function is introduced to the function [8], then:

$$R(x_i, x_j) = \prod_{m=1}^{k} e^{-\Theta_m (x_{i,m} - x_{j,m})^2} = exp\left[- \sum_{m=1}^{k} \Theta_m (x_{i,m} - x_{j,m})^2 \right]$$

(2.13)

The new point x_p is correlated with other sample points x_n with a correlation vector:

$$\boldsymbol{r} = \boldsymbol{R} \cdot \boldsymbol{w}$$

(2.14)

Inverting the above equation, the weight vector becomes:

$$\boldsymbol{w} = \boldsymbol{R}^{-1} \cdot \boldsymbol{r}$$

(2.15)

The unknown value $\hat{y}$ can be obtained with the results of the n sample points by reformulating the equation (2.12), where

$$\boldsymbol{y} = \begin{bmatrix} y(x_1) \\ \vdots \\ y(x_n) \\ 0 \end{bmatrix}$$

and

$$\hat{y}(x_p) = (\boldsymbol{R}^{-1} \cdot \boldsymbol{r})^T \cdot \boldsymbol{y} = \boldsymbol{w}^T \cdot \boldsymbol{y}$$

(2.16)

The accuracy of the Kriging metamodel increases with adding more sample data points, besides it results in large correlation matrices to be stored. Kriging is appropriate for discontinuous functions as well. Therefore it is also suitable for multivariate design problems, and a global surrogate model prior to the fitting process is not needed [8].

Zill [8] applied this method in aircraft design optimization to calculate more accurate results during the transition stage from conceptual to preliminary design phases. In that study, Gaussian function is used to correlate the two data points. As a result, a software

frame is developed which automatically updates the scaling function with high fidelity model data to improve the scaled low-fidelity model while converging to the true high fidelity optimum point.

In this thesis, the new data point is also correlated with other data points to map the design space. However, due to the nature of Kriging the interpolations are used to predict the low fidelity data points by Zill in [8], whereas the Cavus algorithm presented here employs discrete values and classifications to predict the probability of success of the vectored changes on the variables in combined intervals. In other words, the Cavus algorithm can be grouped into probability based pattern search methods. Because of that, with the Cavus algorithm it is not surprizing to converge an area which is far away to and not in between the unsystematically distributed training points.

2.3.5. Globex Algorithm

The GLOBEX algorithm is developed and presented by Jacob [52]. The method searches for the global optimum of a limited multivariable function without the knowledge of its derivatives. It can be used for any number of design variables and with any number of inequality constraints, since the used multivariable function can be calculated indirectly or analytically and the boundaries can also be changed during the computation [16]. The method is the superior version of the optimization method which searches for the local optimum named as EXTREM. If the objective function has more than one local optimum then EXTREM finds the closest extreme value to the selected starting point [16].

In the first optimization step of the technique, estimates are determined by a sequence of normally distributed perturbed numbers. The user defined starting point is used as the mean vector value of these normally distributed perturbed numbers. Additionally, the user defined starting step sizes define the mean square deviations of these normally distributed points. If these points lie in the given limits then the partial optimization starts at each of these points with EXTREM [16].

In the second optimization step, with taking into account the possible restrictions the unsystematically estimated values are determined around the best function value found so far. A partial optimization is again started at each of these points with the possible constraints. If a more favourable function value is found, this point becomes the new value for the further search and the mean quadratic deviations are multiplied by 0.9 [16].

The best of all values determined in these two sections is stored and used as an initial value in the third optimization section for the main optimization, GLOBEX [52].

The properties of the GLOBEX algorithm are [52]:

- The probability to find the global optimum increases with the number of unsystematical estimates produced.
- The GLOBEX algorithm can be used for complex functions like *narrow, crooked valleys* at or near multiple boundaries.

The technique is described in detail by Jacob in [16] and [52]. Here, its main context and strengths are also dealt with to present the general idea.

Method

The algorithm is following these tasks while searching the extremum of a bounded multivariable function [16]:

1. Choose the search directions
2. Find the optimum along a line
3. Define the search step sizes
4. Check the constraints

When the number of variables is n, the array of the step sizes is given in the form $DX(n)$ and this defines the first main line in other words the optimal search direction. The initial point is also given by the user and presented with an array, $\overrightarrow{X_\iota}$. As in the Figure 2.10, the extremal point $\overrightarrow{X_{\iota+1}}$ is determined approximately along the first line. After this first iteration a second line is calculated by a Gram-Schmidt orthogonalization process, which is going through the point $\overrightarrow{X_{\iota+1}}$ and orthogonal to the main direction. At the second iteration, along this second direction $\overrightarrow{X_{\iota+2}}$ is determined and a third line is sketched. This line is orthogonal to the two first directions [16].

The first stage is accomplished when the n^{th} iteration is completed and the extremal point $\overrightarrow{X_{\iota+n}}$ is reached along the n^{th} line, which is orthogonal to all of the previous directions. The new main direction is found by joining the initial point $\overrightarrow{X_\iota}$ and $\overrightarrow{X_{\iota+n}}$, the extremal point of the last iteration of the previous stage. The procedure may begin with assigning $\overrightarrow{X_\iota} = \overrightarrow{X_{\iota+n}}$, and then the main and secondary search directions can be found accordingly [16].

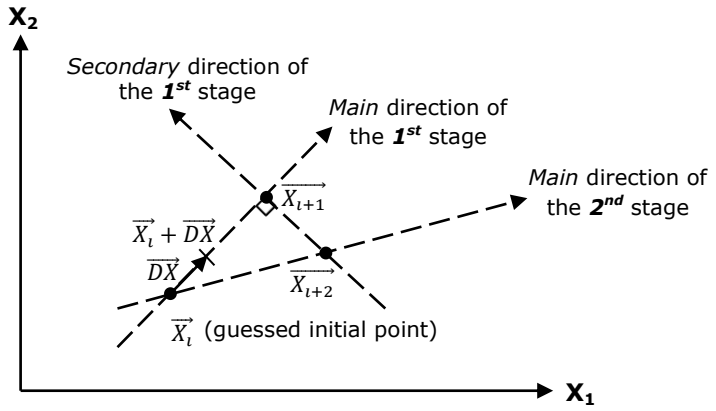


Figure 2.10 Determination of the main search direction in the case of a two variable problem [16]

Afterwards, the three function values F_1 at $\overrightarrow{X_\iota} - \overrightarrow{DX}$, F_2 at $\overrightarrow{X_\iota}$, and F_3 at $\overrightarrow{X_\iota} + \overrightarrow{DX}$ are evaluated to find the extremum of an artificial parabola which is going through these function values with interpolation or extrapolation [16]. From the Figure 2.11 :

$$\overrightarrow{X_{\iota+1}} = \overrightarrow{X_\iota} + \frac{\overrightarrow{DX}}{|F_1 - 2F_2 + F_3|}\frac{F_3 - F_1}{2M_M} \tag{2.17}$$

where $M_M = +1$ for the search of a maximum

$= -1$ for the search of a minimum

The progression of the new point $\overrightarrow{X_{\iota+1}}$ is limited by the algorithm to a maximum of 20 step sizes $\overrightarrow{DX}$. Even if the function value F_{opt} at $\overrightarrow{X_{\iota+1}}$ is worse than the F_2 the next iteration is started from that point. On the other hand, if the difference between $F_{opt}$and F_2 is greater than 4 times the absolute value of the difference between the function values of last two iterations, another function value is calculated with dividing the step size of $\overrightarrow{X_{\iota+1}}$ by 2. Allowing the worse function values and the related points to be used for the next iterations helps to escape from a sharp corner of a narrow valley or a top of a ridge in some cases [16].

Step sizes are defined according to these principles [16]:

- If the distance between the new and the old point is smaller than $1/4$ of the current step size along that line, then the actual step size is divided by 4.
- If the distance between the new and the old point is greater than 20 times the actual step size, then the step size is multiplied by 2.

With these basic rules, the search algorithm reduces the steps sizes while approaching the optimum point and as well escapes from the tight curves by increasing the step sizes.

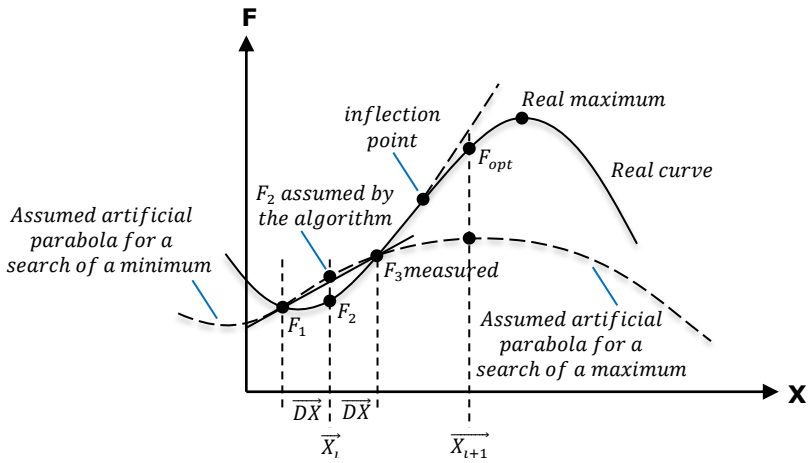


Figure 2.11 Assumed artificial parabola for a search of a minimum [16]

If the boundaries are exceeded during the iterations then [16] [52]:

- The step size is divided by 4 and the new point $\overrightarrow{X_{\iota+1}}$ is placed other side of the old point $\overrightarrow{X_\iota}$, which is away from the boundary, Figure 2.12.

- The step size of the new point is divided by 10 and the new progression will be in the direction of boundary, so far the previous trial is found by an extrapolation or interpolation.

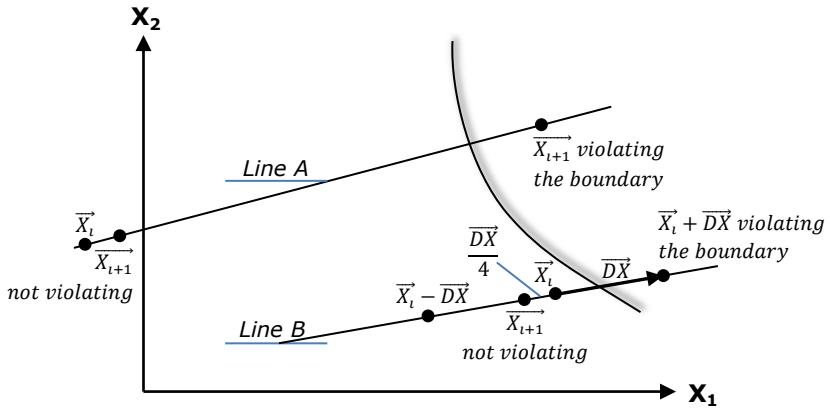


Figure 2.12 Two-dimensional situation near a boundary [16]

Besides, these are the general limitations of the GLOBEX algorithm [16] [52]:

- The starting point should not be selected outside of the boundaries, which is indeed a general prerequisite for most of the methods
- Even though the algorithm is capable of handling the changing boundaries, the current values of the variables must not exceed the limits
- The termination criteria should be severe enough to avoid from primitive results or divergences.
- The step sizes should be reduced sufficiently to follow even the contour of the edges.

2.3.6. Artificial Neural Networks

Artificial Neural Networks, which are inspired by biological neural networks, learn from previous data (training sets) and they are capable to solve nonlinear complex problems. Throughout their discovery and improvements, Artificial Neural Networks have popular and also stagnant time intervals. To teach the algorithm, large data sets so that highly expensive computational power is needed. With the improvement on computational power (GPU, distributed systems etc.) and with finding new unsupervised methods to improve the weights of the algorithm -thus the learning structure-, Artificial Neural Networks have made a visible return.

Artificial Neural Networks (ANN) is classified into four categories by Faghri and Hua in [42] as:

- Mapping ANN
- Recurrent ANN
- Temporal ANN
- Hybrid ANN

Mapping ANNs finds the outputs by summing the products of all inputs and corresponding weights. Linear Associator, Learning Matrix, ADALINE and MADALINE (Multiple ADALINE), Back Propagation, Self-Organizing Mapping and Adaptive Resonance Theory (ART) are the types of Mapping ANNs. Between these, ADALINE and MADALINE use a least mean square error-correcting learning rule. Because during the learning process the error information is propagated back from the output layer to input layer to reduce the error, the best known ANNs was called as Back Propagation ANN. Self-Organizing mapping ANNs is one of the interesting and efficient ANNs to sort items into categories or to cluster. The Kohonen Layer is one of the well-known types of these ANNs uses winner-takes-all strategy with firing only the successful unit [42].

In *Recurrent ANNs*, the outputs are linked to the inputs and their values are sent and used as successive inputs. Hopfield, Brain-Sate-in-a-Box (BSB), Bidirectional Associative Memory (BAM), Boltzmann Machine and Recurrent Back Propagation are the types of this ANNs. Associative ANNs associate inputs and outputs. One of them is Hopfield networks which associate the inputs to the outputs that resemble more to the input patterns. Besides, Bidirectional Associative Memory also associates input and output but which can be different from each other but related somehow. Boltzmann Machines is stochastic version of the Hopfield network uses Simulated Annealing to find the weights. Recurrent back-propagation ANNs can recognize time-dependent input-output data which can be said as dynamic patterns [42].

In *Temporal ANNs*, the output and input vectors are dynamic and it can be represented by a differential equation.

*Hybrid ANN*s use both a supervised and an unsupervised learning in one network.

Faghri and Hua [42] summarized the application level and areas of these types of Artificial Neural Networks in Table 2.5.

Table 2.5 Application Evaluation of ANN Models [42]

Category \ Model	Adaline/ Madaline	ART	BAM	Hopfield	Boltzmann Machine	Back propagation	BSB	Linear Associator	Learning Matrix	Kohonen Networks
Recognition	●			●	●	●				
Control	✱	✱	✱	✱		●				○
Forecasting/Prediction	●					●				
Classification		●				●	✱			●
Diagnosis		✱				✱	●			
Optimization				●	✱	●				
Noise Filtering	✱					●				
Image Processing	✱	✱	✱	✱	✱	✱	✱			
Association			✱	●	○		○	○	✱	
Decision Making		✱								
Temporal Processing	✱	✱				●				

Key: ● (strong applicability); ✱ (moderate applicability); ○ (applicable); □ (difficult to evaluate)

Among the widespread application areas, some of the ANNs like Hopfield and Back Propagation are also used for optimization problems as shown in the Table 2.5. This encourages developing new algorithms that use ANN's strengths as in this thesis.

AI based codes are regarded as intelligent by some researches as far as they do some tasks like manipulating mathematical formulas, prove theorems and understand some amount of natural language. Nevertheless, for some researches these programs are limited with learning and this affects the level of understanding. Even though the experimental proof of success of these programs may not be acceptable by more mathematically inclined people, the proof mechanism defines the limits and the special conditions of the working algorithm [53].

After formulating the problem, various possible action sequences are employed to reach the solution. Different search strategies can be used depending on the problem type and the history of the data [32].

In regard to that, at the beginning of the aircraft design process researchers may have huge numbers of data from wind tunnel tests as well as from flight tests at least for the conventional aircraft. All of these data with engineering sense of experienced engineers, which is inseparable throughout a design process, have great value on all of the design phases. Learning from data, which is data mining, with the combination of engineering sense has very precious fine tuning effect on a design with knowledge based methods. As a result, some of these complex interactions can also be estimated with relative magnitudes just from the data at hand at the beginning of the design process, which helps to accelerate the calculations and force the results to converge to better values. Aircraft design with dependent and independent variables with known interactions to each other in an optimization process can be improved with mimicking human intelligence, i.e. artificial intelligence, for the early as well as ensuing stages of design.

In this thesis, it is aimed to use Probabilistic Neural Networks in a hybrid method. It is selected because of its ease of handling binary values and affecting the computational speed accordingly. This method is then to be used for the optimization problems of aerospace environment, which are closed loop processes and a probabilistic method supports to investigate the search space efficiently. Before introducing the algorithm, Probabilistic Neural Networks and its application areas are presented. Despite the fact that, Probabilistic Neural Networks is a kind of Artificial Neural Networks, it is better to describe it after mentioning the extensive issue that is *probability* at the next section.

2.4. Probability

Determining what is true in the world is based on our observations of the world [24], and our cognation. In other words, our knowledge is limited with our discoveries until this time. Because, for most of the circumstances we cannot have a mass of data from the required aspects, we decide on the outcome with some knowledge and assumptions, which depend on our previous observations and our cognation. This prediction process is called *reasoning under uncertainty*. To make a good decision, an agent does not consider only the previous knowledge and assumptions accordingly; as well it must take into account the other possible situations and their probabilities. Thus, reasoning under uncertainty is related with both decision theory and probability theory [24].

Also as stated by Bishop [54], uncertainty is the key parameter for the pattern recognition. The probability theory when combined with the decision theory serves a consistent framework to do estimations even for defect or vague information [54].

The systems that use artificial intelligence do not always have enough information to make reasoning and decisions. In many situations, they are incomplete or even unreliable. Then, decisions should be made by the artificial intelligence system under uncertain conditions; it means that the system has to make *decisions under uncertainty* [24], [47].

When the optimization problem deals with stochastic variables rather than deterministic values, probabilistic methods are used. The unsystematically disturbed variables can be the dimensions of parts of a mechanical system which have tolerances, or the loads of an aircraft under changing flight conditions [43].

Considering that X is a discrete unsystematically distributed variable, the probabilities for all possible values of X are $\mathrm{P(X)}$ that is the *probability distribution* of X. If all the possible values of X have the same probability, then this is called as *uniform probability distribution*. There are several probability distributions. One more example can be *binominal distribution* used when there are n independent trials. Each trial has two possible outcomes (success or failure) and the probability of success is constant over all trials and presented by Sucar [47] as:

$$P(r|n,\pi) = \binom{n}{r} \pi^r (1-\pi)^{n-r} \tag{2.18}$$

Where,

$$\binom{n}{r} = \frac{n!}{r!\,(n-r)!} \tag{2.19}$$

Three properties of probability are listed below and all conventional probability theory can be derived from these rules [47]:

1. $P(X)$ is a continuous monotonic function in $[0,1]$: Closer to 1 means that the event is more likely to happen, else, closer to 0 means that it is less likely to happen
2. Product rule: $\mathrm{P(X,Y|Z) = P(X|Z)P(Y|X,Z)}$: Probabilities of dependent events are related with their conditional probabilities
3. Sum rule: $\mathrm{P(X|}Y\mathrm{) + P(\neg X|Y) = 1}$: The probability of an event to happen and not to happen are complements

Where,

X, Y, Z are binary variables and $\mathrm{P(X)}$ is the probability of X.

$\mathrm{P(X|Z)}$ is the *conditional probability*; the probability of X given Z.

$\mathrm{P(X,Y|Z)}$ is the probability of X AND Y *(logical conjunction)* given Z.

$\mathrm{P(\neg X|Y)}$ is the probability of NOT X *(logical negation)* given Y.

Thus, if two events, X and Y, are independent from each other, the simultaneous occurrence of these events is found by:

$$\mathrm{P(X,Y) = P(X)P(Y)}$$

(2.20)

The function that calculates the probability of a variable X while $\mathrm{X = x_i}$ is named as the *probability mass function* and given by Rao [43]:

$$f(x_i) = P(X = x_i)$$

(2.21)

If the probability of the variable X is liked to be defined for the cases that X is equal or less than a number, x, then the probability is *cumulative probability* and the function is called the *cumulative distribution function* [43]:

$$F(X) = P(X \leq x) = \sum_i f(x_i)$$

(2.22)

Probability density function of an unsystematically distributed variable (which is the continuous case) is given by Rao [43]:

$$f(x)dx = P(x \leq X \leq x + dx)$$

(2.23)

The *distribution function* of X is defined by Rao in [43] as:

$$F(x) = \int_{-\infty}^{x} f(x')\,dx'$$

(2.24)

Where the *normalization condition* is presented by Rao [43] as:

$$F(-\infty) = 0 \;\; and \;\; F(\infty) = 1$$

(2.25)

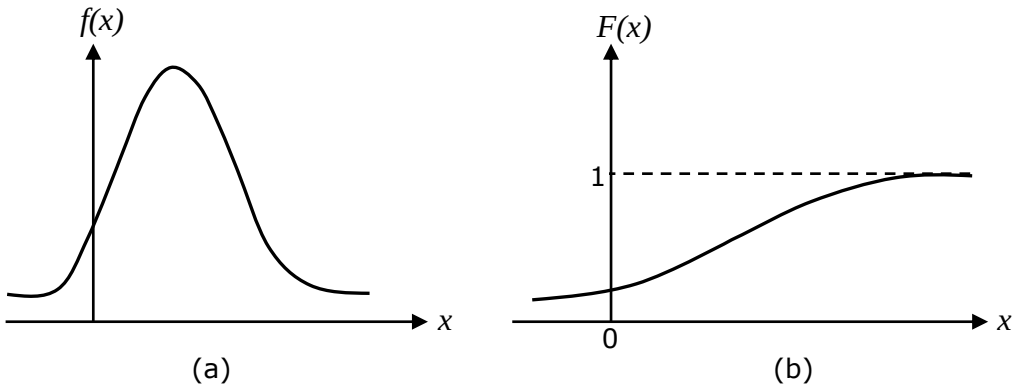


Figure 2.13 Probability density and distribution functions of a continuous random variable X (a) density function; (b) distribution function [43]

The central tendency or average and variability of a variable are termed as the *mean value* and the *standard deviation*, respectively. There are several distributions for discrete and continuous cases. Some are listed by Rao [43] on the Table 2.6:

Table 2.6 Types of probability distributions (analytical models) [43]

Discrete Case	**Continuous case**
Discrete uniform distribution Binomial Geometric Multinomial Poisson Hypergeometric Negative binomial (or Pascal's)	Uniform distribution Normal or Gaussian Gama Exponential Beta Rayleigh Weibull

Before starting to explain the Cavus algorithm, an example is taken from Bishop [54] to describe the probability theory and its basic rules for a better understanding.

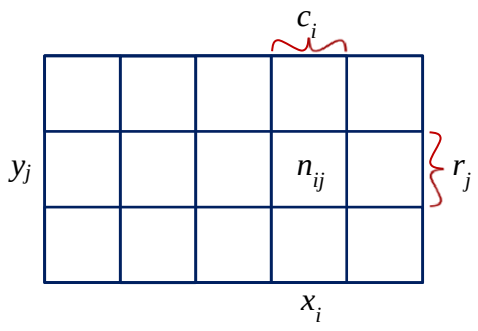


Figure 2.14 Probability theory [54]

Considering two unsystematically distributed variables X and Y, where X takes the values x_i for $i = 1 \dots K$ and Y takes the values y_j for $j = 1 \dots L$ on the Figure 2.14, and the total number of trials is N. The number of trials, where $X = x_i$ as $Y = y_j$, is then n_{ij}. On the figure, the number of trials that $X = x_i$ independent from the value of Y is c_i, and similarly the number of trials $Y = y_j$ independent from X is r_j. The *joint probability*, where $X = x_i$ as $Y = y_j$, is denoted by $P(X = x_i,\ Y = y_j)$. These are the points that fall in the cell i,j and the joint probability is found as [54]:

$$P\big(X = x_i, Y = y_j\big) = \frac{n_{ij}}{N}$$

(2.26)

Then, the total points fall in the column c_i is $\mathrm{X} = \mathrm{x_i}$, that are irrespective of the value of Y, and the probability is [54]:

$$P(X = x_i) = \frac{c_i}{N}$$

(2.27)

Also, the total points fall in the column r_j is $Y = y_j$, that are irrespective of the value of X, and the probability is [54]:

$$P\big(\ Y = y_j\big) = \frac{r_j}{N}$$

(2.28)

In addition to that, the total number of trial points on column i is calculated as [54]:

$$c_i = \sum\nolimits_j n_{ij}$$

(2.29)

Thus, from Eq.(2.26) and Eq.(2.27):

$$P(X = x_i) = \sum_{j=1}^{L} P(X = x_i, Y = y_j)$$

(2.30)

Actually, Eq. (2.30) is the *sum rule* of probability. When we consider that the trial points fall in cell i,j and their probability on column i, it is written as $P\big(Y = y_j \big| X = x_i\big)$ and called the *conditional probability* of $Y = y_j$ given $X = x_i$ and calculated as [54]:

$$P\big(Y = y_j \big| X = x_i\big) = \frac{n_{ij}}{c_i}$$

(2.31)

After combining the equations Eq.(2.26), Eq.(2.27) and Eq.(2.31), the *product rule* of probability is shown as [54]:

$$P\big(X = x_i, Y = y_j\big) = \frac{n_{ij}}{N} = \frac{n_{ij}}{c_i}.\frac{c_i}{N} = P\big(Y = y_j\big|X = x_i\big)P(X = x_i) \tag{2.32}$$

The *rule of symmetry* is another property of probability, which is [54]:

$$P(X,Y) = P(Y,X) \tag{2.33}$$

When we handle the rule of symmetry and the product rule together, we get the relationship between the conditional probabilities, and this relationship is called as *Bayes theorem* [54]:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \tag{2.34}$$

Bayes theorem is very effective in pattern recognition and machine learning [54].

Any problem that involves uncertainty can be solved by probability theory. With the era of *big data*, the automated methods for data analysis are being developed. These automated methods are known as *machine learning methods* and that are used to estimate the following or unknown part of the data from the available data. While making decision, the probabilistic methods play a big role. Thus, those machine learning algorithms automatically detect the pattern of the available data and predict the pattern of the data under uncertainty [55].

Table 2.7 Main types of probabilistic graphical models [47]

Type	**D**irected/ **U**ndirected	**S**tatic/ **D**ynamic	**P**robabilistic/ **D**ecisional
Bayesian classifiers	D/U	S	P
Markov chains	D	D	P
Hidden Markov models	D	D	P
Markov random fields	U	S	P
Bayesian networks	D	S	P
Dynamic Bayesian networks	D	D	P
Influence diagrams	D	S	D
Markov decision processes (MDPs)	D	D	D
Partially observable MDPs	D	D	D

2.4.1. Probabilistic Neural Networks

Probabilistic Neural Networks (PNN) use non parametric probability density function (PDF) estimation for classification with a structure of Neural Network. The training process is fast but on the other hand it needs lot of memory. While the training set increases it can approach Bayes optimal, besides it becomes sensitive to outliers [56].

The objective of the Probabilistic Neural Networks is to classify any new data into one of the classes introduced before. To estimate the class of the new data point, the PNN uses a probability density function for each of the classes [57]. After calculating the probabilities the data is classified to the class which has the highest value. The PNN has four layers and can map any input pattern, whether that has continuous or binary variables, to any number of classifications. By defining a set of weight, which equals to the new training vector, a modification on the decision boundaries is possible with the new data. Parallel processing is also possible with PNN. Moreover, nonlinear multivariate regression surfaces and subsequent probabilities of an event can be calculated, as well with a small change on the structure it can be used as an associative memory [14].

PNN can be illustrated as the following:

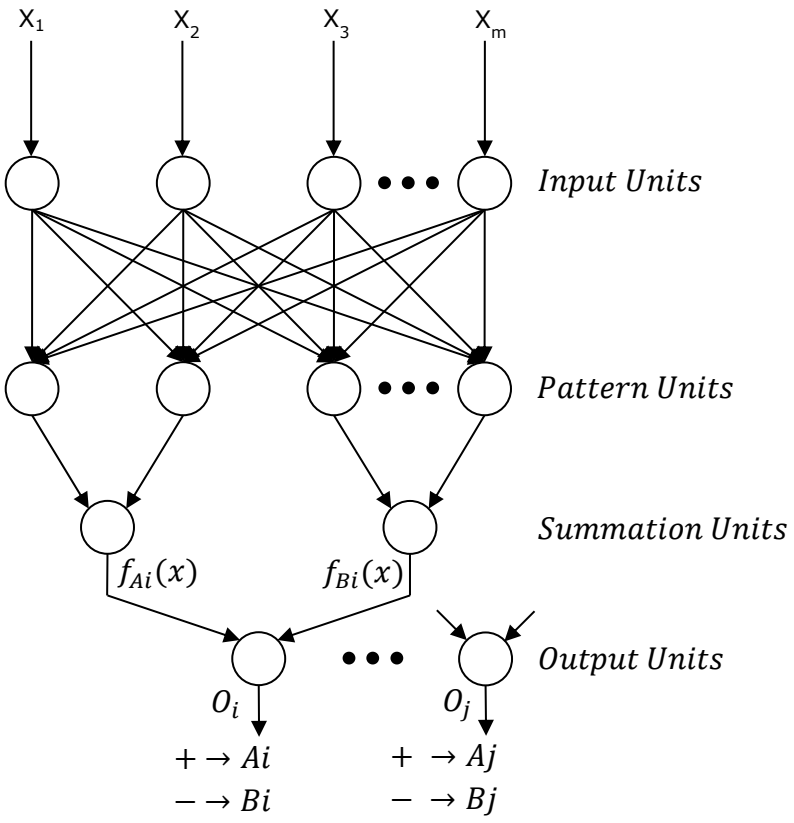


Figure 2.15 Probabilistic Neural Networks [14]

Input layer has a number of neurons equal to the number of the variables of the problem and each has a connection with the neurons of the next layer. This hidden layer covers the pattern units and it has as many neurons as the samples in the training set.

Each training sample has a Gaussian function centred at. Afterwards, the hidden layers at the same classes are connected in the relative summation units, with this way additional pairs of categories can be added to the output vector [14], [58].

At first, the input vector X is multiplied (dot product) with weight vector W, then a nonlinear operation is performed before transferring the activation level to the summation unit. In Probabilistic Neural Networks instead of using sigmoid activation function used for back-propagation, an exponential function is used. When inputs, x, and weights, W, are normalized to unit length this exponential function, or in other words the *activation function*, becomes [14]:

$$activation\ function = e^{[-\frac{(W_i - X)^t (W_i - X)}{2\sigma^2}]} \tag{2.35}$$

Here σ is the *smoothing parameter* and it has a significant effect on the PDF. While small σ produces distinct modes, larger σ eases the interpolation between points. Further, very large σ converts the shape of PDF to Gaussian [14].

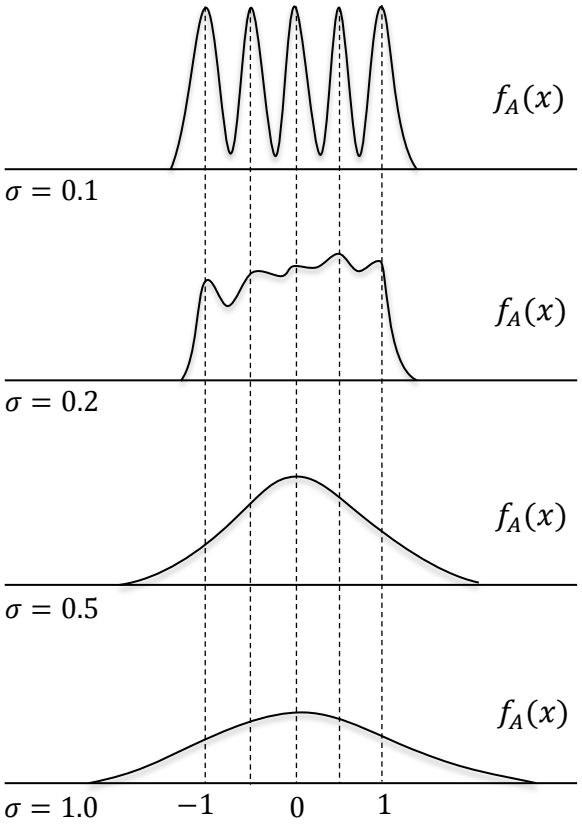


Figure 2.16 The smoothing effect of σ on an estimated PDF from 5 samples [14]

The summation units sum the inputs coming from the pattern units and these sums are simply the sum of small multivariate Gaussian distributions, which are centred at each of the training samples. Indeed, with changing smoothing parameter it can approximate any density function. Figure 2.16 shows the effect of smoothing parameter on an estimated PDF for 5 training samples. With a small value of smoothing parameter, $\sigma = 0.1$, each training points is illustrated with 5 distinct modes. When the value is increased to $\sigma = 0.2$, high degree of interpolation occurs between the samples. With a value of $\sigma = 0.5$, the shape approximates the Gaussian and has a single mode. The smoothing parameter with a value of $\sigma = 1.0$ has a flattening effect on the distribution and also extending the tails [14].

When $\sigma \to 0$ the decision boundary forms like a very nonlinear boundary as the nearest neighbour classifier, also when $\sigma \to \infty$ then the decision boundary becomes a hyperplane [14], [59]. Optimal separation in general is not possible with the limiting values of σ. The network of the PNN is similar to that of the nearest neighbour decision rule [60] proposed by Hecht-Nielsen [61]. Therefore, in accordance with the density of training samples it is better to select a degree of averaging of nearest neighbours rather than a single nearest neighbours. Further, this value of the smoothing parameter is also affected by the dimension of the problem and the number of training patterns. Yet to find a reasonable value is not very hard, since the misclassification does not vary substantially [14], [62].

The main properties of the PNN are presented by Specht in [14], it can:

- be used for mapping, classification or direct estimation of a posteriori probabilities
- be used as an associative memory: If some of the variables are unknown then the missing values are found by scanning all the possible values which maximize the PDF.
- be used to estimate a category of a new pattern even after introducing each category with one representing training pattern. Not only the decision boundary becomes complex but also the generalization improves with additional patterns
- tolerate noisy samples
- be used for scattered data
- be used for time-varying data
- be as faster as $200{,}000$ times than back-propagation [56] and it has parallel structure

On the other hand, it requires high memory space to store the model [63].

2.5. Comparison of the Algorithms

Table 2.8 is the performance table for some of the previously discussed algorithms, which were also used for comparison in the following sections. These comparisons here are illustration purposes only and the inferenced ideas from the references [13], [14], [32], [24], [43], [44], [45], [46], [47], [52], [54], [55], [64]. In the table, the properties in the last two highlighted lines actually indicate the deficiencies, and the rest points the superiorities of the ANN algorithms.

Table 2.8 Comparison of the Algorithms

Property	Genetic Algorithm	Simulated Annealing	Globex	ANN
Convergence speed	fast	slow	very fast	fast
Initial population dependency	high	high	high	low
Global search capability	high	low	low	high
Convergence to local minimum	high	high	high	medium
Continuous Problems	high	high	high	high
Discrete Problems	medium	low	high	high
Memory usage	medium	low	very low	high
Computational effort	high	medium	low	high

Source: illustration purposes only

Besides the fact that ANN algorithms demand high capacity of memory and high computational effort, they are preferred because of their high global search capability and convergence speed to the optimum. Likewise the example problems in this thesis, if the design problem requires rather more time to calculate the result of the objective function than the time for the optimization algorithm itself, and as well if the problem has discrete structure, then it is more favourable to use ANN algorithms.

Although, the main bottleneck of the ANN algorithms is their need for high computer capacity, there are many ongoing researches about it because of the facts that:

- Powerful hardware are also under development [14]
- The new discoveries with these algorithms ease the tasks of the scientists already and they have still undiscovered aspects, so they are open to any improvements

In this thesis, with the developed algorithm it is aimed to integrate one of the ANNs to gain its pros (like handling nonlinear complex problems efficiently) while diminish the cons (as the case, high memory usage) by treating the search space as patterns and reducing the dimensions accordingly. Although the ANN algorithms need powerful computers, they have the capacity to handle the huge number of unstructured data and to process functions in parallel that most of the traditional methods cannot manage. As a result, with its hybrid structure the developed algorithm has better optimization results as well as lower computational effort and time.

3. Cavus Algorithm

An optimization problem is composed of dependent and independent variables. Deciding on the variables, which have as possible as lower level multicollinearity, and on the objectives is the first challenging part while constructing algorithm. If there is no training set, the *guided random search techniques* as described in Chapter 2 -Genetic Algorithm and Simulated Annealing- can be used. However, the *run time is directly affected with the increasing number of variables, and the quality of the results by the size of the search space*. If there is a training dataset, numerical techniques like Multiple Nonlinear Regression and Kriging can be combined with Multiple Gradient Descent Algorithms to optimize the design. Due to huge number of calculation steps and interpolating mechanism, the convergence quality and the run time change drastically.

When it comes to driving information and efficiency, the best way of searching and using the data becomes the leverage or key for the robustness of problem solving. Artificial Intelligence (AI) algorithms combine mathematical models of the computer science and big datasets to enable problem solving efficiently. Predictive algorithms of AI are able to discover the patterns and detect the anomalies of the system.

For an optimization problem, the total input-output calculations of every possible input are CPU intensive and cost time. If a good relation between input and output can be drawn, and well-fitted surrogate models can be extracted from the relation, the computational effort and the time can be reduced. When the initial sets of data exist or are produced by a design code the Machine Learning algorithms can be trained and then used to test the further data. The idea here is to develop an algorithm to use for surrogate modelling with integrating an AI method, while the surrogate models are used for approximating the outcome of a function without utilizing the exact inputs, and the relationships between the inputs and outputs can be assessed well with an AI algorithm. That means the developed model is used as a black box to guess the patterns more likely to be successful. Then only then the real calculation would be done when the pattern has better probability, which is considered together with the inherited success and the neighbouring patterns. But for the exploration, the developed algorithm is used to define the whole pattern space which corresponds to the design space indirectly. Especially because of its correlation characteristics, the method may best fit to the problems that have deterministic characteristics rather than the problems that have stochastic properties. In other words, the closed loop processes are well application areas. Like an aircraft design with an aircraft mission, that uses known amount of fuel to convert the chemical energy to potential and kinetic energy.

Although the aircraft are designed with predefined requirements, the whole design process includes lots of unknowns and uncertainties that have not been defined with exact formulations. Thus, before the first flight, some hours of flight simulations are

performed and corrections are done as much as possible. However, for a safe flight wind tunnel tests are done at first and a rough flight envelope is defined also based on calculations, accordingly. Then, real test flights are done to improve this envelope and to fulfil the certification requirements for further safe flights. Throughout this design cycle, the calculations are improved by experienced engineers in parallel. All of these processes are time and budget consuming. This know-how accumulation, transfer and the best use of data become vital. Even a small improvement at the pre-conceptual design phase saves time and budget a lot, and moreover even life.

A very detailed design is not aimed at the early design phase of an aircraft, because every detail cannot be exactly structured straight off at that stage, but a good optimization algorithm that guesses better design solutions is always on demand for aerospace domain. Classical evolutionary optimization algorithms that are mostly used in aerospace may be not be much efficient as discussed and shown in the following sections. Because its consistent characteristics, surrogate models that may fit best to the design area and be explored and trained by an Machine Learning algorithm is to be more efficient then the classical methods. The basic methods at this area are correlation algorithms and they may be the good starting points. Actually, the correlations between input and output for an aircraft design case were already worked in [65] and [66]. From these studies, the inference was made as that, in an aircraft design there is logical relations between inputs and outputs, among which reasonable patterns can be extracted. Accordingly, the related patterns were produced and illustrated in [67].

In this study, instead of using instant variable values and interpolation between them, the respective changes of design variables from one design solution to others are concerned and stored while searching the subsequent design points. This comparison stage helps the algorithm to shape and reshape the search surface in parallel with surrogate patterns at every iteration, and feeds the algorithm with more promising trial points, accordingly. The success of the approach is examined and verified in Section 3.1. Probabilistic Neural Networks which is mentioned at Part 0 is used to find out the desired changes on the search space. Thus, this method classifies the gradients in terms of each optimization parameter and objective function changes at an intermediate step. Then, it continues with more probable sections of the design space by assigning new training points as using a dimension reduction method in parallel. Here, the trial points are also called as the training points. This should not result in any confusion; the reason behind is that the trial points employed in one iteration are also used in the training patterns for the successive iteration. The related multidisciplinary design optimization flowchart is shown in Figure 3.1.

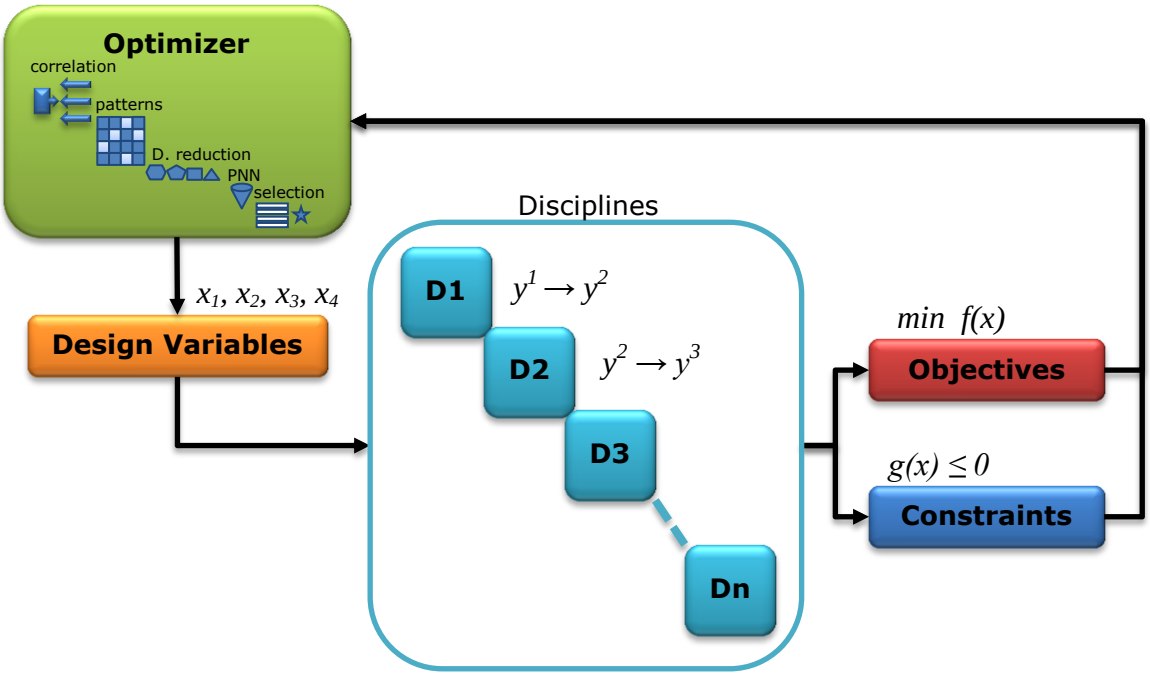


Figure 3.1 Multidisciplinary Design Optimization Flowchart for Cavus Algorithm

The idea of the Cavus algorithm was introduced step by step with the studies in [68], [69], [70], and [71]. In the first two studies, Multiple Cooling Multi Objective Simulated Annealing Algorithm (MCMOSA) is used to optimize a supersonic multirole aircraft. The unsystematically distributed nature of the MCMOSA algorithm was improved with Pearson Product correlation and a basic knowledge-based method was integrated accordingly to have better optimization results. Indeed, besides linear fitness function, elliptic and ellipsoidal fitness functions are used in MCMOSA algorithm. With these approaches, the pareto front is aimed to be captured by the geometrical functions. The idea is to draw a perfect pareto front and then reduce the distance between this perfect pareto and cost function results. Since, the perfect pareto is known geometrically, a better design point corresponded to each point on the pareto would be guessed with the help of elliptic/ellipsoidal fitness functions. As in Simulated Annealing the cost function is replaced by the energy of the system in MCMOSA. The specific temperature is used while computing the change in the energy of the system. If the energy is reduced then the trial point is accepted. However, the trial point may also be accepted without considering the energy reduction but having the probability of reduction for the next steps. In MCMOSA, instead of one fitness function a population of fitness functions are minimized together. Another originality of the algorithm is the assignment of a specific temperature parameter to each fitness function [72], [73]. The so-called random walk of Simulated Annealing in [68], which is one of the guided random search techniques, used in the optimisation part is improved with an introduced knowledge-based technique in [66].

For the aircraft design problem, this method gives better results just with adding an intermediate step that analyses the environment and assigns vectored increments to the design variables. The run time is decreased, while the iteration count stayed the same. For the limited loop number the results of the trained algorithm are closer to the Pareto front and distributed more firmly.

Besides the fact that, this improvement serves better accuracy and slightly better run time, the function of evaluations are not affected due to the fact that the termination criteria was to fulfil the limited number of loops. Indeed, with this study, [66], it was observed that the relevance, strength and the direction of the relation between the variables and the objectives are very effective, which should be examined in detail. Actually, the main deficiency of this kind of algorithms may be their so-called random walk nature which increases the number of function evaluations till converging to a local optimum.

Meanwhile, the coupled algorithms with reliability based function Neural Networks, probability and evidence theory are used by researchers and better results are gained. Likewise the general approaches, they also use the interpolations between the design points. In engineering, especially in conceptual design phase, due to the lack of knowledge, assumptions are made and some fix numbers are used to overcome the uncertainties. Alternatively, to handle these epistemic uncertainties the appropriate probabilistic models are searched instead of using unsystematically distributed variables with assumed probabilistic models [74], [75]. Even though, these techniques are promising for the complex systems, they need high computational effort [76].

Considering all of these approaches with their pros and cons, the novel optimization technique used in this study is developed. It depends on rule-based agents that search for and act to find out the promising design space. The Probabilistic Neural Networks algorithm is integrated in the observation phase, which is applied here for its success on pattern recognition and classification problems. In this hybrid method the advantages of both gradient based and evolutionary algorithms were used. As in the gradient based approaches, it uses the relative change of independent and dependent variables between each design point to trace the whole search space; and also as in the evolutionary algorithms it uses the roles of population in the related range. As stated before, in an optimization process the run time and the convergence characteristics are affected by increasing the number of variables drastically. This disadvantage is superseded with a hybrid technique in this study by reducing the number of poor design points which are anticipated from the previous experiences, i.e. training points.

The method works first as an observer and correlates the input and output pairs, then as a classifier and then as an estimator for the optimization part; and this improves the unsystematical selection of the variables. In view of the fact that, the time expended in design part is in most cases more than the time expended in optimization part, which calls the design part for each design point. This also means that, total run time is increased by the number of candidate design points multiplied by the time to execute one design point in design part. In the Cavus algorithm, the time to spend for calling and finding the results of an objective function in barren parts of the design space is eliminated by a classification process. That also helps to shorten the time expended in design part with reducing the poor candidate design points to be calculated. This gained time is used to find out more promising design points, so that improves the convergence with fewer amounts of function evaluations.

As illustrated in Figure 3.1 first the correlation matrixes should be structured. Then, as stated in Figure 3.2, the number of variables and the first set of training points are introduced to the program at first. The number of variables may be decided to be as much as based on the design case, and if they are adaptive to any change between the boundaries during the optimization process.

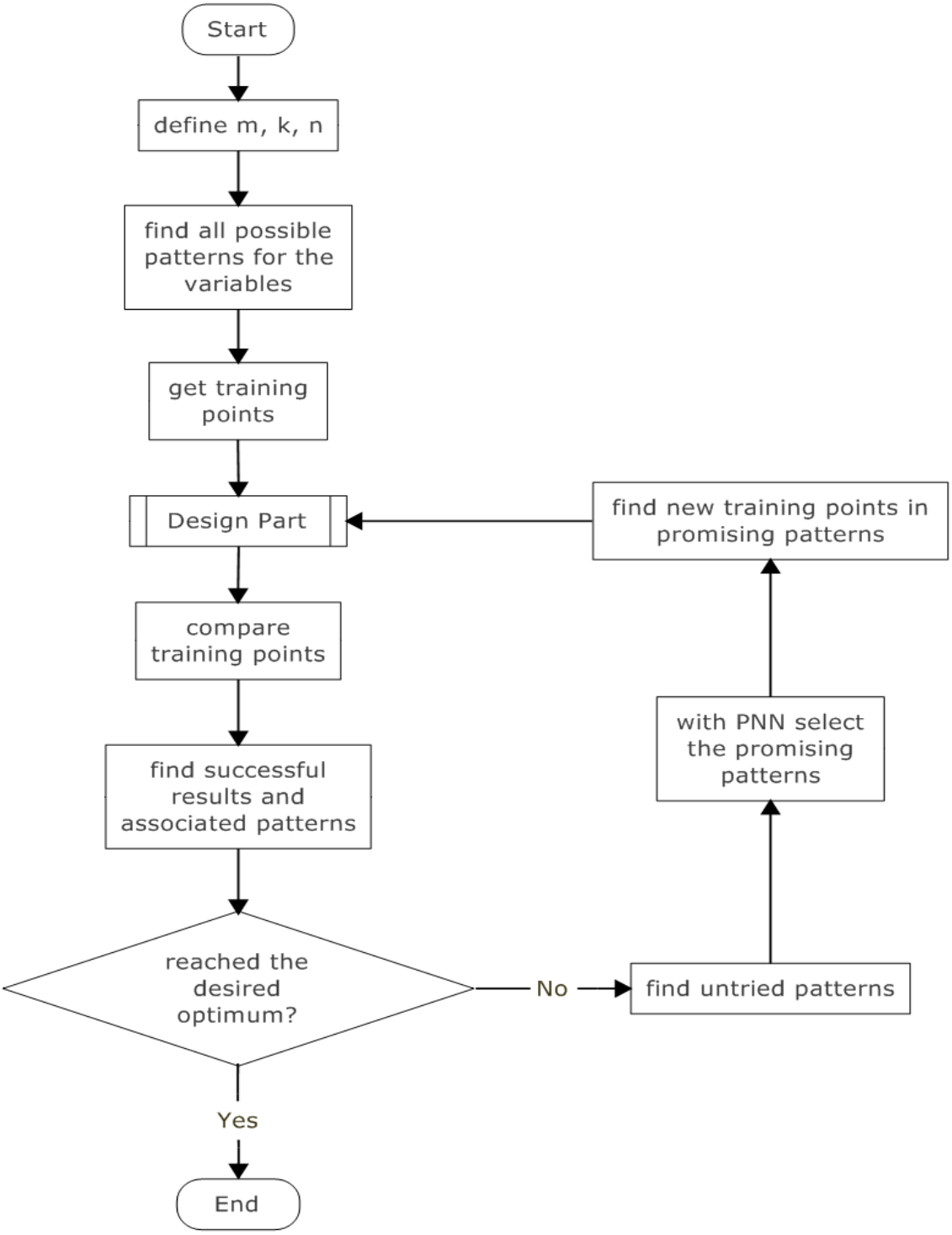


Figure 3.2 Flowchart of the Cavus Algorithm

The number of intervals may be decided to be as much as also to increase the accuracy. On the other hand, the computation time and the memory usage increase accordingly, thus it is better to keep it smaller and even 1. Since the number of training points is another concern for the success of the AI algorithms, two different cases are tested. For the design cases selected in this study, training points are kept small enough and for each pattern n training points are selected unsystematically. Depending on the

problem characteristics the number of training points can also be selected proportional to the number of independent variables. Likewise, with increasing variable numbers the number of training points can be increased accordingly as having a value between $k\,x\,m$ and k^m, where the number of intervals of each variable on a pattern is k and the number of independent variables is m.

Indeed, the number of training points is selected based on the study at the section 3.1, and they are applied on each grid, then the unsystematically selected variable values are sent to the design part. Then the results are compared with each other to build up a correlation matrix, which shows the changes in the independent design variable values and the objective values. This means, if the training point number is n, the resultant combinations will be $n\,x\,(n-1)$ with considering the increment and also decrement effect. At that stage, the increment is symbolized as 1, and the decrement as -1. If there is no change between the compared values it can be taken as 0. With bipolar (and also 0) values and the related intervals, each combination of training points can be processed as *patterns*. The mentioned numbering system will be used at the next steps for handling the correlations of the patterns. If the objective is to minimize the fitness function the correlation patterns with the objective correlation value -1 are taken as the *successful patterns*, others are left as *unsuccessful patterns*. Additionally, depending on the lower and upper bounds of the dependent and independent variables, the search space for each variable is divided in k intervals, and the interval boundaries are stored. Each variable has number of patterns, pt, calculated as in Eq. (3.1):

$$pt = 2 * (k + \sum_{i=1}^{k} (k - i))$$

(3.1)

If the number of design variables is m, the amount of total potential patterns is pt^m. As an example, for 1 variable and 3 intervals the possible patterns are illustrated sequentially in Figure 3.3; for 2 variables and 3 intervals the patterns are illustrated in Figure 3.4.

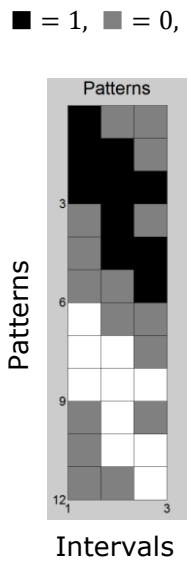


Figure 3.3 Patterns for one variable (*k=3*)

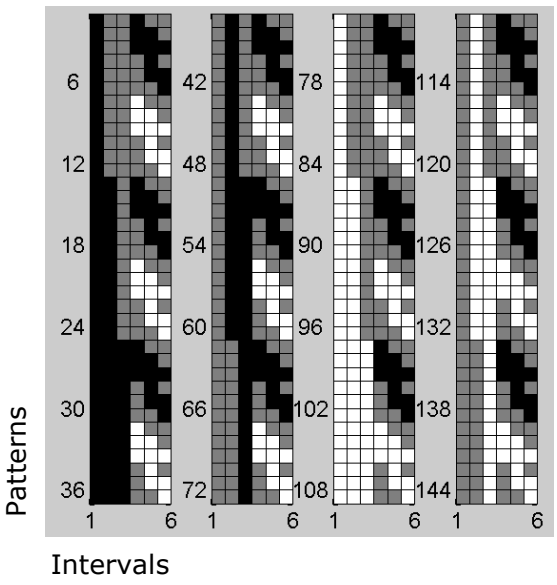


Figure 3.4 Patterns for two variables (*k=3*)

For 1 variable and 5 intervals the possible patterns are illustrated sequentially in Figure 3.5, thus there are 30 patterns; for 2 variables and 5 intervals there are 900 patterns which are illustrated by matching 30 patterns of each variable in Figure 3.6. Afterwards, for the first training dataset the unattempted patterns can be extracted from the total pattern sets.

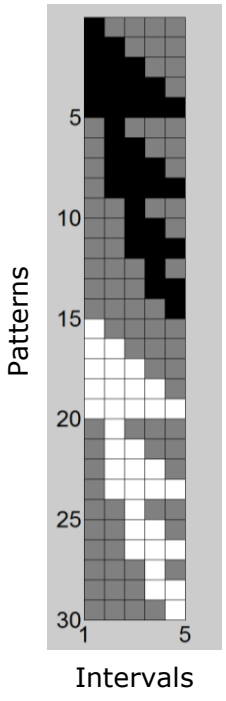


Figure 3.5 Patterns for one variable (*k=5*)

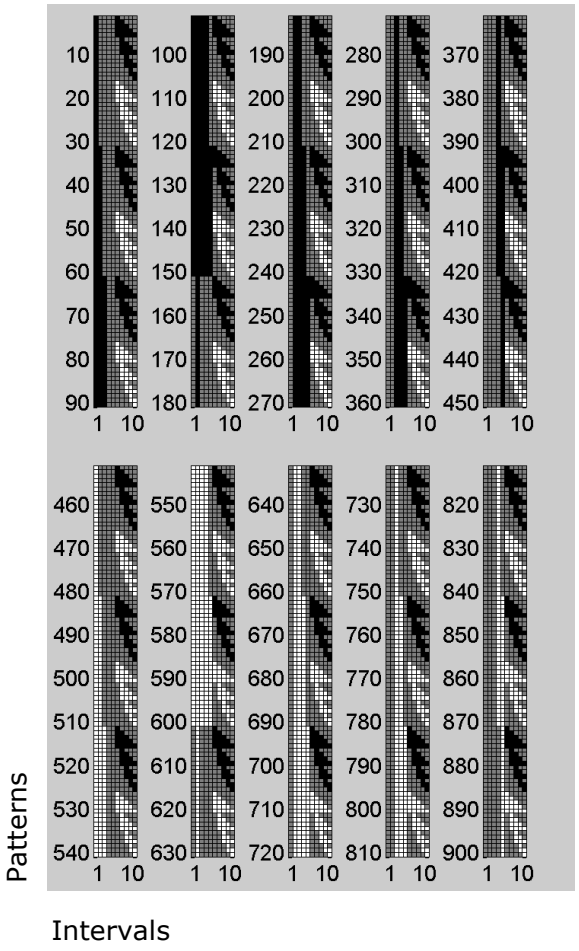


Figure 3.6 Patterns for two variables (*k*=*5*)

From Figure 3.3, Figure 3.4, Figure 3.5 and Figure 3.6, it can be interpreted as when the number of intervals is increased the number of patterns is also increased rationally. Because each pattern requires memory and costs run time, the number of intervals should be kept as less as possible.

After deciding on the number of intervals and finding the total patterns, dimension reduction equation (3.12) can be applied on them to reduce the number of nonessential patterns to save the memory and run time accordingly. Since, the patterns differ from each other with a few digits as being structured like neighbours; similar patterns are classified mostly in the same class at the next step.

At that point, Probabilistic Neural Networks comes into action. As illustrated in Figure 3.2 it is used to find the promising patterns between the untried patterns. The algorithm is coded in Matlab R2018b, and the Deep Learning Toolbox is used to apply Probabilistic Neural Networks. The function is called as *newpnn* and the usage is explained in Appendix.

Probabilistic Neural Networks uses Bayes Strategy instead of using sigmoidal activation function, which is widely used with an exponential function in back-propagation algorithm. This method can compute nonlinear decision boundaries, which can be

updated immediately with a new data, and can also be operated in parallel [14]. Because of its structure, it is faster than back-propagation especially for pattern recognition and classification.

Probabilistic Neural Networks (PNN) is applied here to classify the patterns depending on their acceptance probabilities. PNN was introduced by Specht and here summarized based on the flowing references; [14], [56], [59] and [62]:

While classifying patterns the decision rules are set to minimize the expected risks. These rules or strategies are called Bayes Strategies and can be applied to any number of categories, [14], [56] and [77]. Considering the categories A and B the state of natures are θ_A and θ_B, and the probability density functions are $f_A(x)$ and $f_B(x)$ respectively. Also, I_A and I_B are the loss functions related with the decisions $d(x) = \theta_A$ when $\theta = \theta_B$ and $d(x) = \theta_B$when $\theta = \theta_A$ (the losses are taken to be equal to zero when the decisions are correct). Further, h_A and h_B are the priori probability of occurrence of patterns from category A and B, and $h_B = 1 - h_A$ [56].

Then, for a state θ based on a set of measurements represented by a p-dimensional vector $x^t = [x_1 \dots x_j \dots x_p]$ the Bayes decision rule is written as in Eq. (3.2) [56]:

$$d(x) = \theta_A \; if \; h_A I_A f_A(x) > \; h_B I_B f_B(x)$$
$$d(x) = \theta_B \; if \; h_A I_A f_A(x) < h_B I_B f_B(x) \tag{3.2}$$

Also, the boundary between the region in which Bayes decision $d(x) = \theta_A$ and the region in which Bayes decision $d(x) = \theta_B$ is given as in Eq. (3.3) [56]:

$$f_A(x) = K f_B(x) \tag{3.3}$$

Where

$$K = \frac{h_B I_B}{h_A I_A} \tag{3.4}$$

The ratio of the loss functions, h_B/h_A, can be set to -1 if there is no reason for biasing the decision [56]. According to [57] a family of estimates of $f(x)$, at all points x the probability density function is continuous, is given with Eq. (3.5) [56]:

$$f_n(x) = \frac{1}{n\lambda} \sum_{i=1}^{n} W\left[\frac{(x - x_{Ai})}{\lambda}\right] \tag{3.5}$$

Eq. (3.6) is the weighting function $W(y)$ and states that weights are not bounded and cannot reach infinity:

$$sup_{-\infty<y<\infty}|W(y)| < \infty \tag{3.6}$$

Here, sup indicates the supremum.

$$\int_{-\infty}^{\infty}|W(y)|dy < \infty \tag{3.7}$$

$$\lim_{y\to\infty}|yW(y)| = 0 \tag{3.8}$$

$$\int_{-\infty}^{\infty}W(y)dy = 1 \tag{3.9}$$

In Eq. (3.5), let λ is chosen as a function of n then $\lambda = \lambda(n)$, and

$$\lim_{n\to\infty} n\lambda(n) = \infty \tag{3.10}$$

[57] proved that the expected error goes to zero with the number of training samples going to infinity:

$$E|f_n(x) - f(x)|^2 \ \to 0 \ as \ n \to \ \infty$$

The assumptions of the absolute continuity of the distribution F(x) are relaxed at [77] and [78], and then [79] extended Parzen's results for multivariate case [56]. Then the multivariate estimates are found by Eq. (3.11) as:

$$f_A(x) = \frac{1}{(2\pi)^{p/2}\sigma^p}\frac{1}{m} \times \sum_{i=1}^{m} exp\left[\frac{(x - x_{Ai})^T(x - x_{Ai})}{2\sigma^2}\right] \tag{3.11}$$

Where σ is the smoothing parameter and it has a very important influence on the approximations.

The Probabilistic Neural Networks like "Feed Forward Networks" has a parallel structure [14]. This type of Neural Networks is very flexible to accept new data and it can be applied easily with its one-step only learning technique [56]. It learns not from trials instead from experience, that others made for the Neural Network [56]. Therefore it depends on the functions used inside the neuron [14]. Because of these characteristics, the Probabilistic Neural Networks is faster than back-propagation [80] and they perform well with few training points [56].

In this study, Probabilistic Neural Networks is preferred to use for their advantages and success on the pattern recognition and classification, and also their tolerance to the usage of binary-bipolar numbering combination.

As an example, in Figure 3.8 the usage of Probabilistic Neural Networks in the Cavus algorithm is illustrated for two variables, X_1 and X_2, and the objective, Y. At first, the successful and unsuccessful patterns are distinguished according to their influence on the objective function. If the objective function is decreasing (for finding minimum) at a pattern then the pattern is defined as the *successful pattern* and shown here with a blue arrow. The head of the arrow shows the direction of the action. If it is away from the lower bound of the variable, it means that the variable value is increased at the successive point, and the pattern has the interval value of 1 for that variable. If it is in the opposite direction then the value for that interval becomes -1. If the pattern has an effect on the objective function to increase then the pattern is defined as *unsuccessful pattern* and shown with a red arrow. The directions of the arrows and the numbering for the successful patterns are also valid for these unsuccessful patterns.

Indeed, the successful patterns are selected based on two criteria.

The pattern:

- that minimizes the result;
- that has the equal resultant value (due to the probability of convergence);

After collecting the successful and unsuccessful patterns, Probabilistic Neural Networks is trained. The classes are specified based on the related objective function values. The number of classes for the successful patterns is $(n-1)$. Then the n^{th} class is allocated for the unsuccessful patterns.

The untried patterns are picked out from the combined set of total possible patterns which are formed with the variable and interval numbers. Untried patterns are applied on the trained neural network and the possible successful and unsuccessful patterns are distinguished. Figure 3.8 shows how the Probabilistic Neural Networks selects promising patterns from the successful and unsuccessful patterns.

As a result, as in the Figure 3.8b the promising pattern would be similar to one of the grey lines and headed according to the other successful patterns which may be neighbouring, parallel or both. From the performance values of the tried patterns, neighbouring patterns which have greater probability to minimize the fitness function are selected with the help of Probabilistic Neural Networks. From another aspect, it actually

matches the patterns which have one or two digit differences from the successful patterns, and their combinations. These untried promising patterns with successful patterns are sent to the design part, which calculates the fitness function; and the results are turned with their corresponding patterns to further deliberation.

At that point, each class has number of patterns. Not all but the most promising patterns should have the priority. For that reason, the 2-digit hamming distance function is applied to find out the center of the pattern clusters. With this intermediate stage, the patterns that have higher probabilities to be successful arise. At the end of this process throughout the design space there would not be any untraced space. Moreover, all of this information is gained just from the first set of training points.

For the next step, as in Figure 3.8c, a number of patterns is selected and the next set of training points is applied to these areas. At this step, a dimension reduction method can be applied while selecting the training points. Since the algorithm is capable to bring neighbouring patterns, at least one dimension can be eliminated. After one dimensional reduction the number of training points becomes:

$$pako^2 = 2^m - \left[2^{m-2}\left((m \; mod \; 2) + 2\right)\right] \tag{3.12}$$

Further case specific reductions can be done accordingly. Each reduction saves noticeable amount of memory and run time against some compromise on the result.

The following step is illustrated in Figure 3.8d. The process will continue with the successful and promising successful grids on the search area, which is also stated in the loop of the flowchart of the algorithm in Figure 3.2.

When the required criterion for terminating the program is reached then the process is ended with the minimum value (or maximum value for the maximization). If the fitness is not at the desired level then the selected successful patterns are sent to the design part for generating further training points for the related intervals as described in Figure 3.7. The loop is repeated until the desired optimum is reached.

As a result, in Figure 3.8, it can be interpreted that there should be relations between unsuccessful and successful patterns in inverse proportion and/or direction, and in direct proportion and/or direction between successful patterns and between unsuccessful patterns, respectively.

[2] *Addressed to my elder sister Pakize*

Step 1: Calculate objection functions for n number of design points

Step 2: Compare the design points and find the successful and unsuccessful patterns

Step 3: Extract the untried patterns from the complete patterns

Step 4: With using PNN classify the promising and non-promising patterns

Step 5: Apply the hamming distance function between the successful patterns

Step 6: Apply the hamming distance function between the successful and the promising patterns

Step 7: Sort all of the (successful and promising) patterns based on their class orders (probabilities), and then their hamming distance values

Step 8: Select a number of top patterns for the further analysis

Step 9: Send n number of training points to the corresponding intervals of the selected patterns; at this step the dimension reduction function ($pako$) and sparse distribution can be used

Step 10: If the termination criteria is not met, go to *Step 1*

Figure 3.7 Logical Description of the Cavus Algorithm

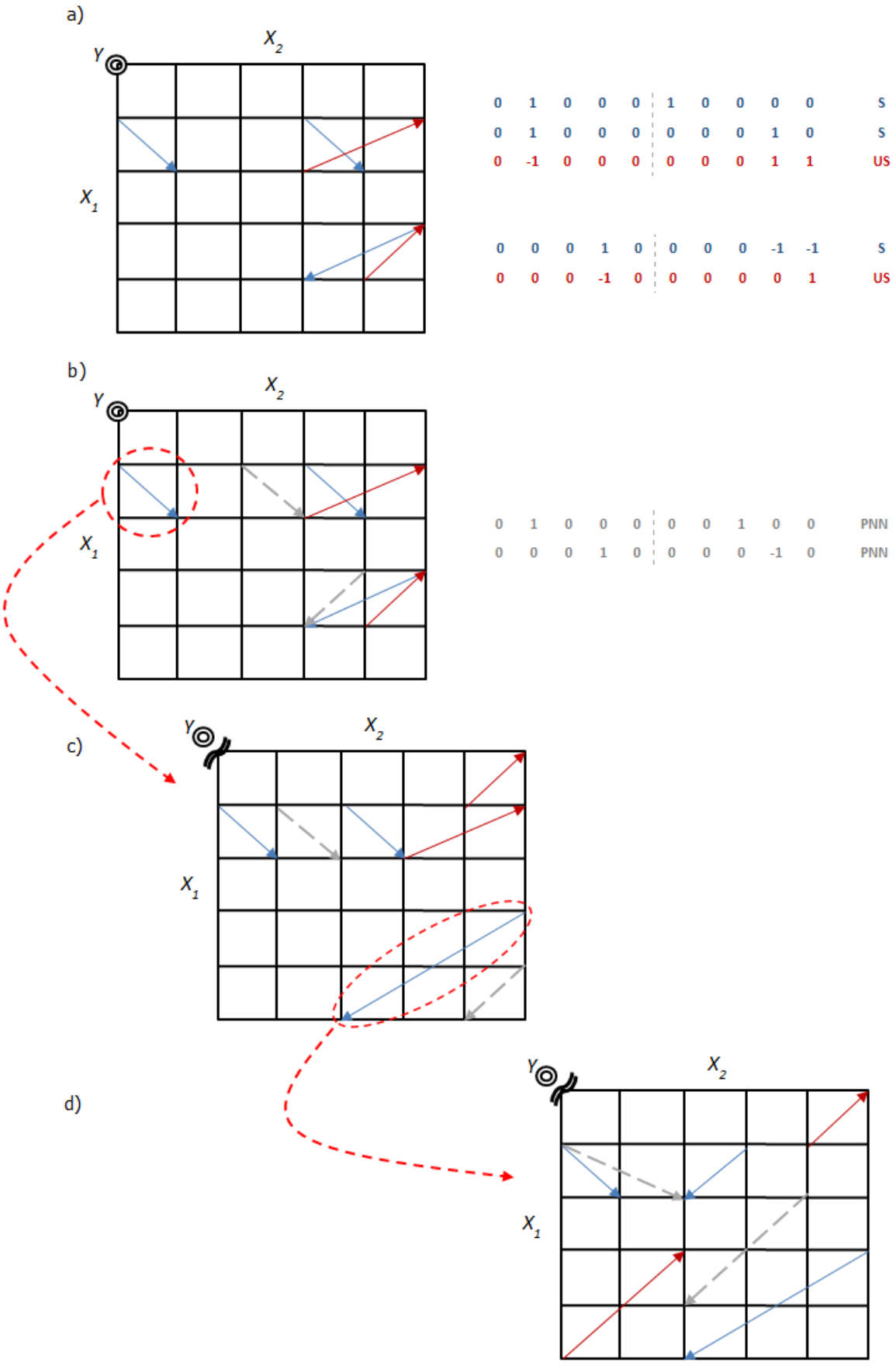


Figure 3.8 Selecting promising patterns with PNN

3.1.Trend Analysis for the number of Training Points

For making a decision on the required number of training points, a trend analysis was done. In the first case, 10 training points are given to the first run and also to the each following successful iterations. The interval number is kept small and selected as 2, which is independent from the increasing number of variables. Optimization results for a fixed number of training points, 10, are given in Table 3.1; and for the changing number of training points, which are proportional to the variable numbers, are given at Table 3.2. Each pattern value is calculated and checked, if it matches with the predicted value by the algorithm or not. Because each pattern with a decreasing effect on the objective function has its inverse pattern, the number of successful patterns is equal to the number of the unsuccessful patterns, as expected. For the first case, when the number of training points is kept as fixed, the algorithm loses its success with the increasing number of variables. If the Table 3.1 and the Table 3.2 are considered together, when the training points are more than the required value (like the case $n = 3$), the algorithm is highly trained and it does not let enough patterns to be included in the promising pattern set. Thus the success of the method decreases. Besides that, for the increasing number of variable numbers ($m > 3$) the fixed number of training points is less than the required level, then the success of the promising patterns of the optimization diminishes. This effect is seen in both of the tables, Table 3.1 and Table 3.2, for a relative number of intervals and variable numbers. It is also observed that with the increasing number of variables, the success of the algorithm increases as expected; this is because of eliminating the less promising patterns, which come from more digit changes.

$$Success\ of\ the\ method = \frac{Calculated\ successful\ patterns}{Total\ promissing\ patterns} \tag{3.13}$$

Table 3.1 Optimization results for the fixed number of training points (n=10)

Number of variables	7	6	5	4	3
Number of training points	10	10	10	10	10
Number of intervals	2	2	2	2	2
Total pattern number	279936	46656	7776	1296	216
Used patterns	90	90	76	84	58
Unused patterns	279846	46566	7700	1212	158
Successful patterns*	45	45	38	42	29
Unsuccessful patterns	45	45	38	42	29
Promising patterns**	147638	24109	4011	614	83
Total promising patterns* + **	147683	24154	4049	656	112
Calculated successful patterns***	125287	21861	3737	611	106
Success of the method	0.848	0.905	0.923	0.931	0.946

* Calculated
** Estimated
*** Calculated afterwards

Table 3.2 Optimization results for the changing number of training points ($n=k^m$)

Time (sec)	410	218	122	74	50
Number of variables	7	6	5	4	3
Number of training points	128	64	32	16	8
Number of intervals	2	2	2	2	2
Total pattern number	279936	46656	7776	1296	216
Used patterns	12586	3076	718	138	32
Unused patterns	267350	43580	7058	1158	184
Successful patterns*	6323	1543	363	70	16
Unsuccessful patterns	6323	1543	363	70	16
Promising patterns**	134851	22052	3587	583	92
Total promising patterns* + **	141174	23595	3950	653	108
Calculated successful patterns***	138166	22980	3836	629	103
Success of the method	0.979	0.974	0.971	0.963	0.954

* Calculated
** Estimated
*** Calculated afterwards

Further improvements are done by eliminating the points which come from the similar or partially related intervals. This saves the memory usage. On the other hand, it may decrease the efficiency. However, with the increasing number of variables the required memory is also increasing. Due to this fact, the reduced number of patterns is used in the test cases presented here.

Table 3.1 and Table 3.2 can be interpreted as that, for the design cases that require high memory and computational time the number of training points can be selected as less as possible without much compromise to catch the real successful patterns. It is found that, the ratio of calculated successful patterns in the total promising patterns are high enough and changing between 85% and 98% for the selected test case. Moreover, when we consider the ratio of used patterns to total pattern number; for example, in Table 3.2 for 7 variables it is even 4%, which means 98% of success is reached from the derived information of 4% of the whole design space. In fact, total patterns are used to explore the design space better and contribute to the surrogate model. Therefore, the precise calculations are done on really interesting patterns provided by the surrogate model and may have the optimum value.

3.2. Test Case: Rosenbrock Function

To demonstrate the functionality of the novel algorithm, Rosenbrock function is used as an initial test case. This function is mainly used to test the gradient-based optimisation algorithms. It is a challenging function with having many local minima especially around the global minimum. Also, it is flexible to be used for 2 or more variables. For m-dimensional domain, the "Rosenbrock" function is given as:

$$f(x) = \sum_{i=1}^{m-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \tag{3.14}$$

The global minimum is $f(x) = 0$ at $x = (1, \dots, 1)$. Thus, the global minimum has always the value, 0, and it is found at 1 for all the axes. Besides, the form can be illustrated for two variables easily. This case is actually examined step by step in detail in the next section with the related plots. Further, the following sections present the performance of the algorithm with the increasing number of variables.

3.2.1. Method description and the results for Rosenbrock Function for 2 variables

The "Rosenbrock" function is illustrated in the next 3-dimentional plot, Figure 3.9, for 2 variables. At the first sight Rosenbrock function seems very easy to find the optima, however the convergence to the global optimum through the tricky valley is highly difficult.

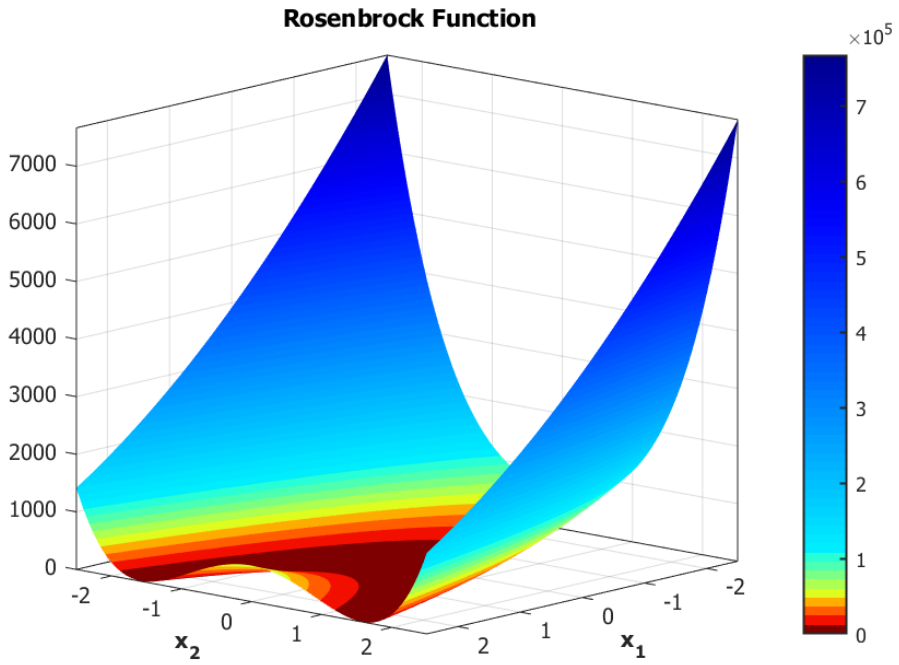


Figure 3.9 Rosenbrock Function (3D view)

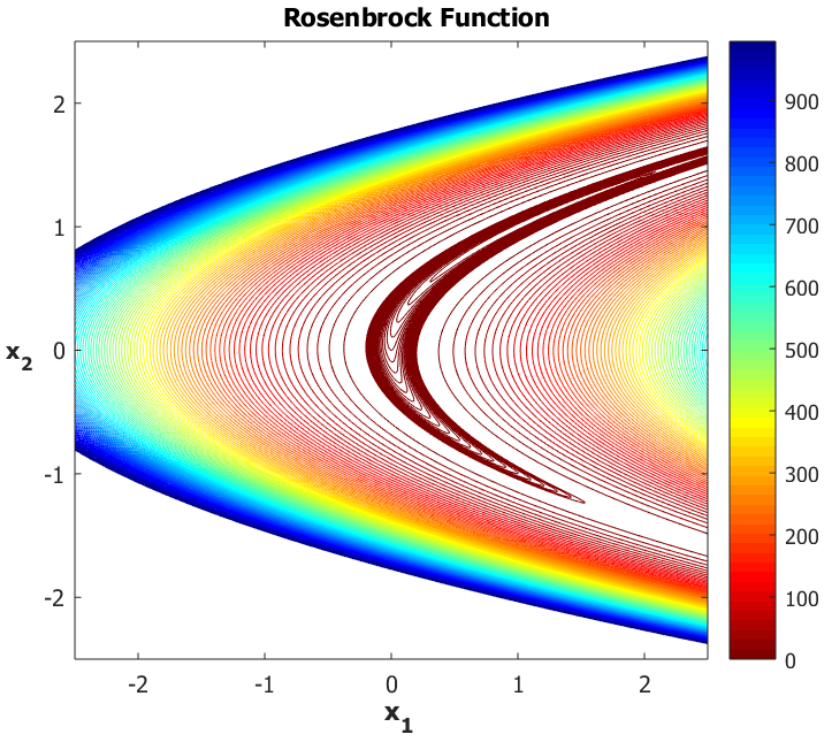


Figure 3.10 Rosenbrock Function (2D view)

Here, the technique is described for 2 variables $(m = 2)$ in order to ease the illustration of the optimization steps. This algorithm does not need any starting point but needs lower and upper bounds. For the two variables x_1 and x_2, uniformly distributed perturbed numbers are generated between lower bounds $lb = [-2.5, -2.5]$ and upper bounds $ub = [2.5, 2.5]$. The ranges between the lower and the upper bounds of the variables are divided into 2 intervals; the interval number is notated as $k = 2$. Thus, the corresponding two intervals are $[-2.5\ \ 0]$ and $[0\ \ 2.5]$ for each variable at the first iteration.

For the sake of the performance of the program, the number of the data/design points (n) to start searching should be selected such as $k\ x\ m \leq n \leq k^m$. Any number lower than $k\ x\ m$ reduces the accuracy of the resultant optimum point, and also any number more than k^m expends high memory of the computer and extends the run time. For the interval number, $k = 2$, and the number of variables, $m = 2$, the number of the data points that will train the algorithm is selected as 4.

These unsystematically distributed points selected by the computer are also illustrated on the next figure and tabulated with the resultant $f(x)$ values on the next table, Table 3.3.

Table 3.3 Design variables with their boundaries and values for the first training set

	Lower bounds	**Upper bounds**	**1. Data point**	**2. Data point**	**3. Data point**	**4. Data point**
x_1	-2.5	2.5	-0.73995	0.97463	1.67295	-1.28586
x_2	-2.5	2.5	1.76959	-0.94789	0.47668	-2.13496
$f(x)$			**152.37190**	**360.16227**	**539.66464**	**1440.42783**

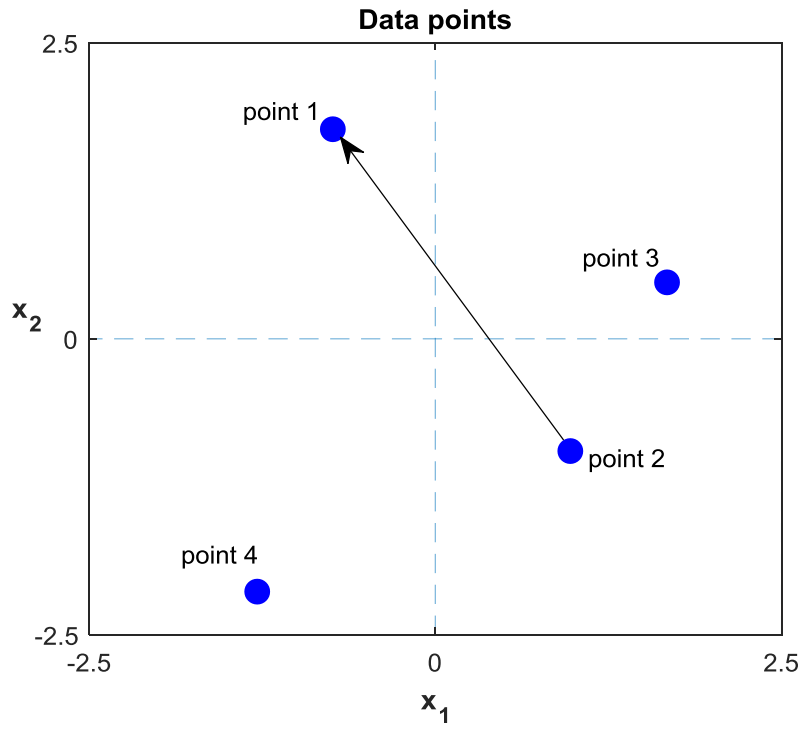


Figure 3.11 First set of data points

After gathering the training data points, the algorithm calculates the differences between the variables and the corresponding differences between the results. The evaluation criteria is that if the value of one variable decreases from one data point to other data point the value of the related pattern between these two variables is -1, otherwise if it tends to increase the value is 1. The same rule is applied between the results/the objective values also. For example for the first and second data points: while pointing out the direction from point 2 to point 1 (i.e. following the arrow on the figure) the value of the first variable is decreasing through its first interval, which is represented as $[-1\ 0]$; whereas the second variable is increasing through its second interval, which is represented as $[0\ 1]$. By the way, the result of $f(x)$ is also decreasing from approximately 360 to 152. This means the pattern between point 1 and point 2 is $[-1\ 0\ 0\ 1]$ and it is a $successful$ action because the result is reducing also. The comparisons are done for each point combinations and the patterns are gathered. The degree of success is decided as: when the result is reducing then it is defined as the *successful action*; on the other hand when the result is increasing (for a minimization problem) then it is defined as the *unsuccessful action*.

Finally, when the number of data points is n, then the combination numbers will be $n\,x\,(n-1)$ and the number of classes of these combinations is at most $(n-1)$. In case of recurring patterns, the number of classes is reduced. Also, one half of these combinations consist of *successful actions* whereas the other half is of exactly their reflected *unsuccessful actions*, as expected. The reason for that, while combining the points, the same pairs match two times which are actually just in different orders (i.e. the arrow just changes its direction).

Table 3.4 Successful patterns

point 1 - point 2	-1	0	0	1
point 1 - point 3	-1	0	0	1
point 1 - point 4	1	0	0	1
point 2 - point 3	0	-1	-1	0
point 2 - point 4	0	1	1	0
point 3 - point 4	0	1	0	1

Table 3.5 Unsuccessful patterns

point 2 - point 1	0	1	-1	0
point 3 - point 1	0	1	0	-1
point 3 - point 2	0	1	0	1
point 4 - point 1	-1	0	-1	0
point 4 - point 2	-1	0	-1	0
point 4 - point 3	-1	0	-1	0

However, for 2 variables and 2 intervals there exist $(2k)^m = (2\ x\ 2)^2 = 16$ possible patterns in total. These possible patterns can be listed as:

Table 3.6 Total possible patterns

-1	0	-1	0
-1	0	0	-1
-1	0	1	0
-1	0	0	1
0	-1	-1	0
0	-1	0	-1
0	-1	1	0
0	-1	0	1
1	0	-1	0
1	0	0	-1
1	0	1	0
1	0	0	1
0	1	-1	0
0	1	0	-1
0	1	1	0
0	1	0	1

After discarding the repeated patterns, out of 16 possible patterns only 8 of them are used. For the rest 8 patterns there is no information yet; whether they would be successful or not when they are used.

Since the aim of the novel algorithm is to reduce the function evaluations for the optimization process, instead of examining these patterns with the new data points,

Probabilistic Neural Networks (PNN) is used to decide whether they would be successful or not. Roughly, PNN finds out the neighbouring patterns of the same classes. The idea of the new algorithm is that, if a pattern is successful the neighbouring patterns should also have similar success. PNN helps to reach these patterns. The training patterns are the successful and unsuccessful patterns that are found out from the trial points of the previous iteration. Additionally, the classes could be assigned according to the reduction size on the $f(x)$ values. Out of this knowledge, (also with ignoring one of the repeated patterns) the first two successful patterns are grouped into the first class, and the following two are into the second, and then the last one is into the third class.

Table 3.7 Successful patterns with classes

1st class	-1	0	0	1
	1	0	0	1
2nd class	0	1	1	0
	0	-1	-1	0
3rd class	0	1	0	1

As a result, the test patterns should be the patterns that are not used before. At this stage, PNN is applied to distinguish the *possible successful patterns* and the *possible unsuccessful patterns from the test patterns*; the results can be tabulated as:

Table 3.8 Possible successful patterns

1st class	0	-1	0	1
	-1	0	1	0
	0	-1	1	0
	1	0	-1	0
	1	0	1	0
2nd class	0	-1	0	-1

Table 3.9 Possible unsuccessful patterns

1st class	-1	0	0	-1
	1	0	0	-1

The classification of PNN is illustrated in the Figure 3.12. The number of successful classes is selected as $(n-1) = 4-1 = 3$; as a result the n^{th} class becomes the class of unsuccessful patterns. As seen in the figure, one pattern has the probability of both to be successful and unsuccessful. This pattern is illustrated with the red-green diagonal plaid colouring.

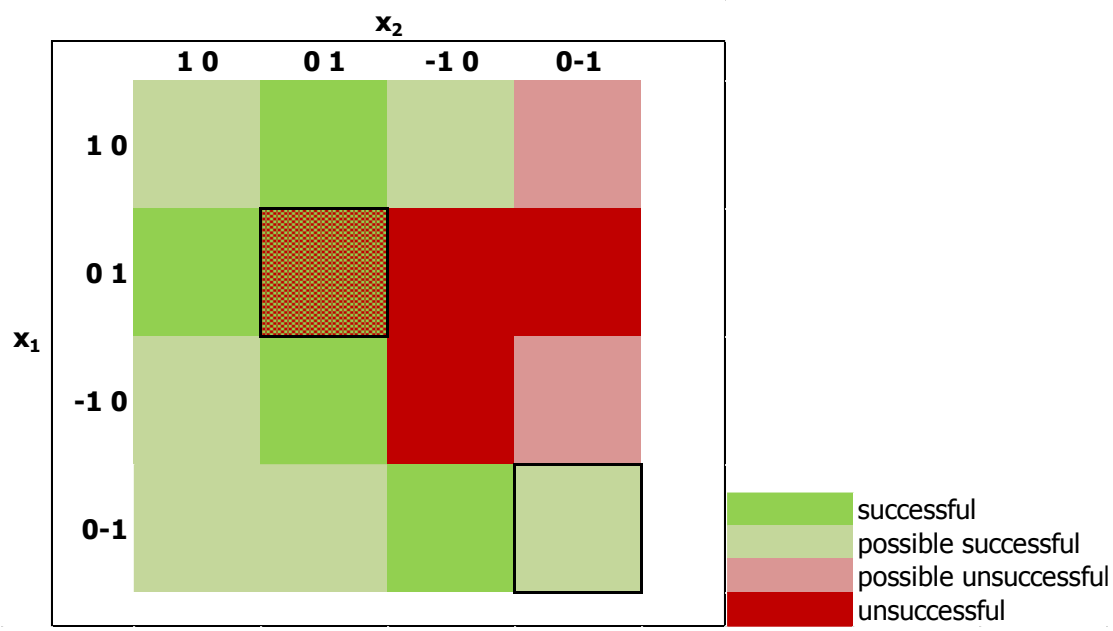


Figure 3.12 PNN classification

From the trend analyses, it appears that the effective number of classes for the *successful patterns* is between 2 and m; and for the *possible successful patterns* is between 2 and $m/2$. Hence, the first patterns of the first 2 classes are chosen from the *successful patterns*, and also from the *possible successful patterns* for the next run. Eventually, the promising patterns become:

Table 3.10 Promising patterns with their bounds

				Lower bound		Upper bound	
x_1		x_2		x_1	x_2	x_1	x_2
-1	0	0	1	-2.5	-2.5	2.5	2.5
0	1	1	0				
0	-1	0	1				
0	-1	0	-1				

The next run will continue within these intervals:

For the first promising pattern the bounds of the first variable, x_1, become $[-2.5\ 0]$ and for the second variable, x_2, they are $[0\ 2.5]$.

For the second promising pattern the bounds of the first variable, x_1, become $[0\ 2.5]$ and for the second variable, x_2, they are $[-2.5\ 0]$.

For the third promising pattern the bounds of the first variable, x_1, become $[0\ 2.5]$ and for the second variable, x_2, they are $[0\ 2.5]$.

For the fourth promising pattern the bounds of the first variable, x_1, become $[0\ 2.5]$ and for the second variable, x_2, they are $[0\ 2.5]$.

Actually, it is known that the global minimum is at the point $(1,1)$ which is between $[0\ 2.5]$ for the both variables; hence the correct patterns are highlighted with green in the Table 3.10.

Although, the first two promising patterns are not able to find the global minimum but the last two have the possibility. Even though, this is obvious for us, the program continues with all of the promising patterns. For each promising patterns n times the trial points are used. This means, the next run includes $n\,x\,4 = 16$ data points, where $n = 4$. In the next figure, Figure 3.13, the data points from the 2^{nd} run are shown.

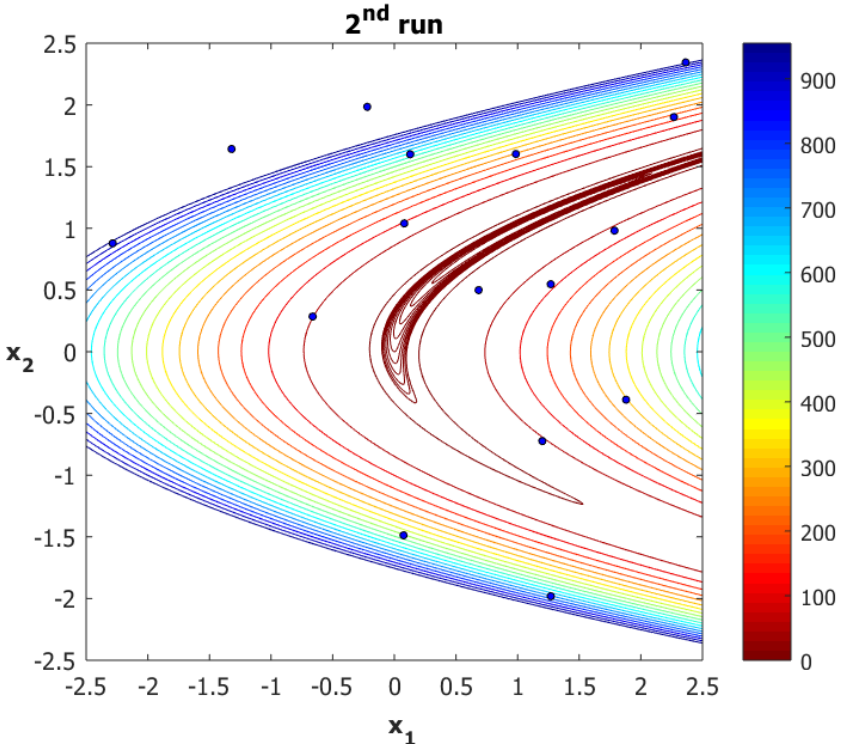


Figure 3.13 Second set of data points

The next figure, Figure 3.14, shows the same data points again, but they are distinguished according to the patterns they come from. Here, it is clearly seen that the 3^{rd} and the 4^{th} patterns have advantage over the other patterns, the 1^{st} and the 2^{nd} patterns, to reach the global optimum.

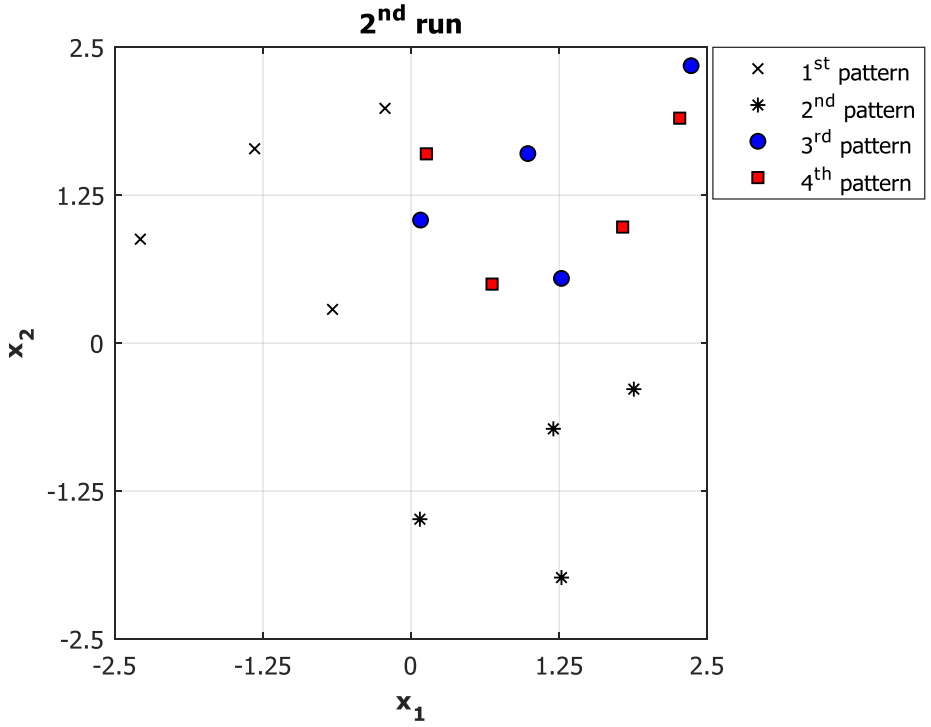


Figure 3.14 Second set of the data points grouped with the related pattern numbers

Table 3.11 The unsystematically distributed points for each pattern and the results

	x_1	x_2	f(x)	min	mean
1st pattern	-0.66276	0.28543	5.13100	5.13100	571.13983
	-1.32026	1.64305	6.38466		
	-0.22011	1.98408	376.15482		
	-2.28509	0.87873	1896.88885		
2nd pattern	0.07440	-1.48748	223.76631	223.76631	880.80255
	1.20081	-0.72394	469.14832		
	1.26950	-1.98078	1290.61397		
	1.88000	-0.38851	1539.68159		
3rd pattern	0.98563	1.60215	39.77662	39.77662	328.31428
	0.08000	1.04000	107.67905		
	1.26911	0.54730	113.13913		
	2.36348	2.34444	1052.66234		
4th pattern	0.68370	0.49922	0.20104	0.20104	447.74509
	0.12895	1.59974	251.38481		
	1.78590	0.98131	488.20277		
	2.26739	1.90132	1051.19175		

The selection of the patterns for the next run depends on the minimum and the mean values. For each pattern, the minimum and the mean values are normalized and summed. Afterwards, the sums are ranked in ascendant order, and then the pattern which has the lowest sum is selected for the next run; that is the 3^{rd} pattern for this example. It is also seen that the 4^{th} pattern which has also meaningful lower and upper bounds on the second rank and has the minimum $f(x)$ value.

Table 3.12 Normalized min and mean values

	Normalized min	Normalized mean	Sum
1st pattern	0.02	0.44	0.46
2nd pattern	1.00	1.00	2.00
3rd pattern	0.18	0.00	0.18
4th pattern	0.00	0.22	0.22

As in the previous run, each pattern brings some promising patterns after PNN, also. These are tabulated in the next table, Table 3.13.

Table 3.13 Actual patterns with the promising patterns for the next run

					Lower bound		Upper bound	
	Promising patterns for the next run				x_1	x_2	x_1	x_2
3rd pattern	-1	0	0	1	0	0	2.5	2.5
	-1	0	-1	0				
	0	-1	0	1				
	0	-1	1	0				
4th pattern	-1	0	-1	0	0	0	2.5	2.5
	-1	0	0	-1				
	1	0	0	-1				
	-1	0	1	0				
1st pattern	0	1	-1	0	-2.5	0	0	2.5
	1	0	0	1				
	0	-1	0	1				
	1	0	1	0				
2nd pattern	-1	0	-1	0	0	-2.5	2.5	0
	-1	0	0	-1				
	0	-1	1	0				
	0	-1	0	-1				

At the 3^{rd} run, which is illustrated in the next figure, Figure 3.15, the search continues on the ¼ of the search area which has the highest probability on others.

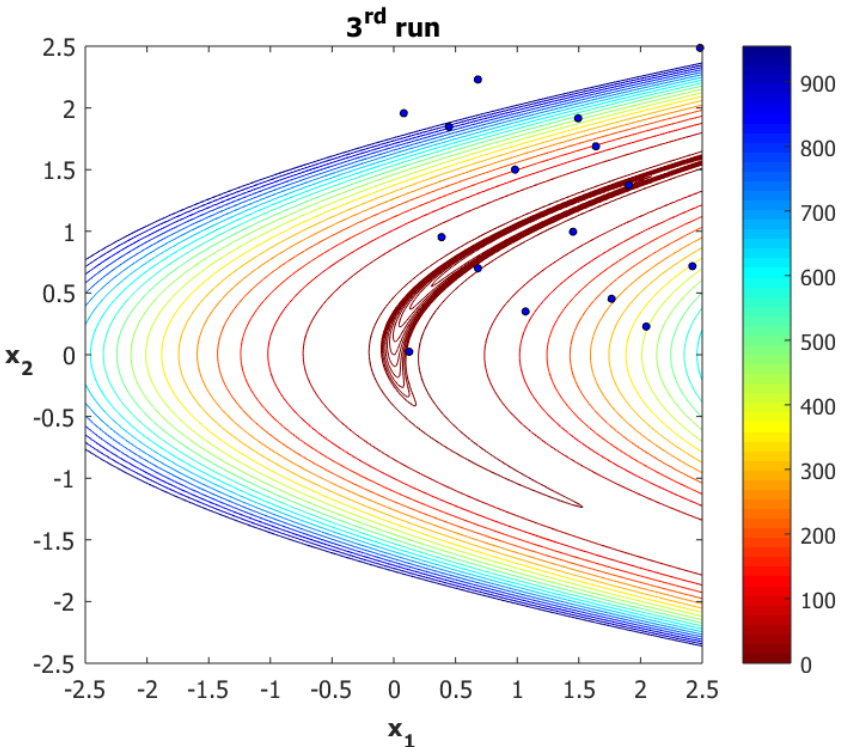


Figure 3.15 Third set of data points

At the 4^{th} run, the data points come closer to the valley, as it could be seen in the next figure, Figure 3.16.

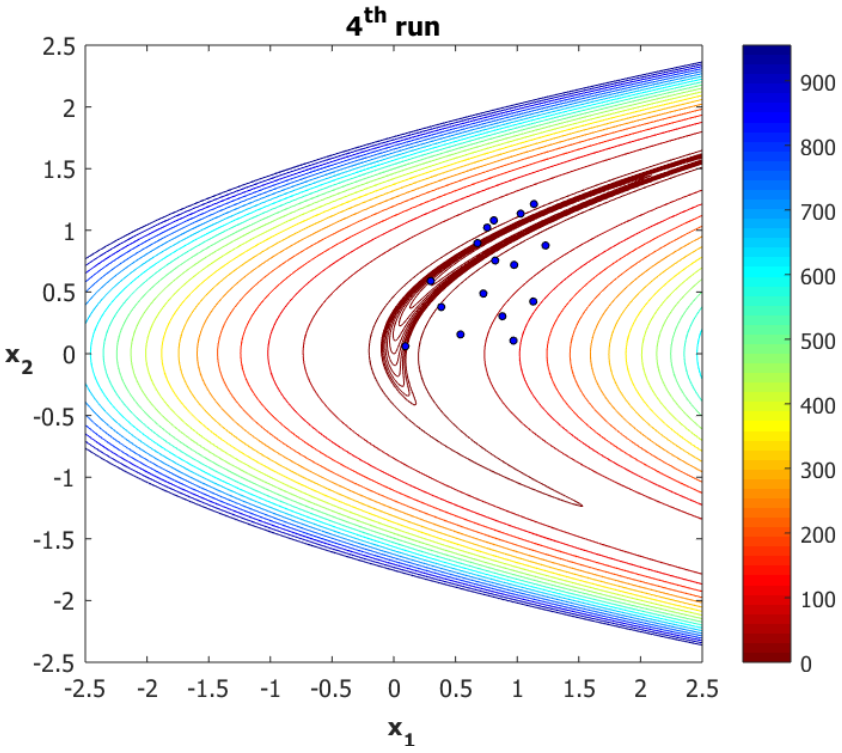


Figure 3.16 Fourth set of data points

The 5^{th} run includes closer data points to the global optimum, however it is still a tricky area to be able to converge to the right curvature.

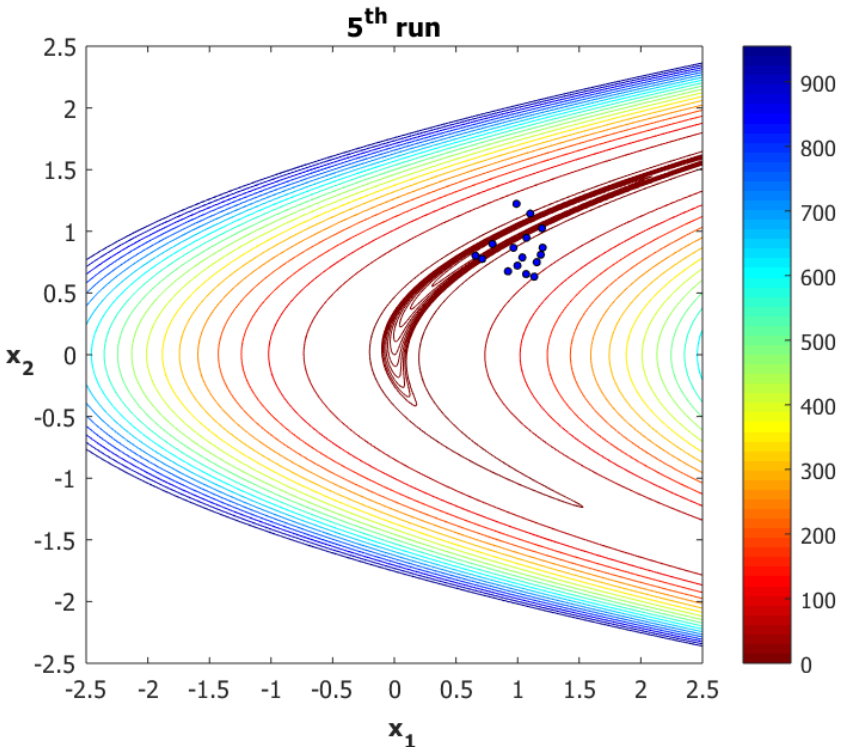


Figure 3.17 Fifth set of data points

At the 6^{th} run the data points are highly closer to the global minimum; the 7^{th}, the 8^{th} and the 9^{th} runs are also need, and they become successful while converging to the optimum point.

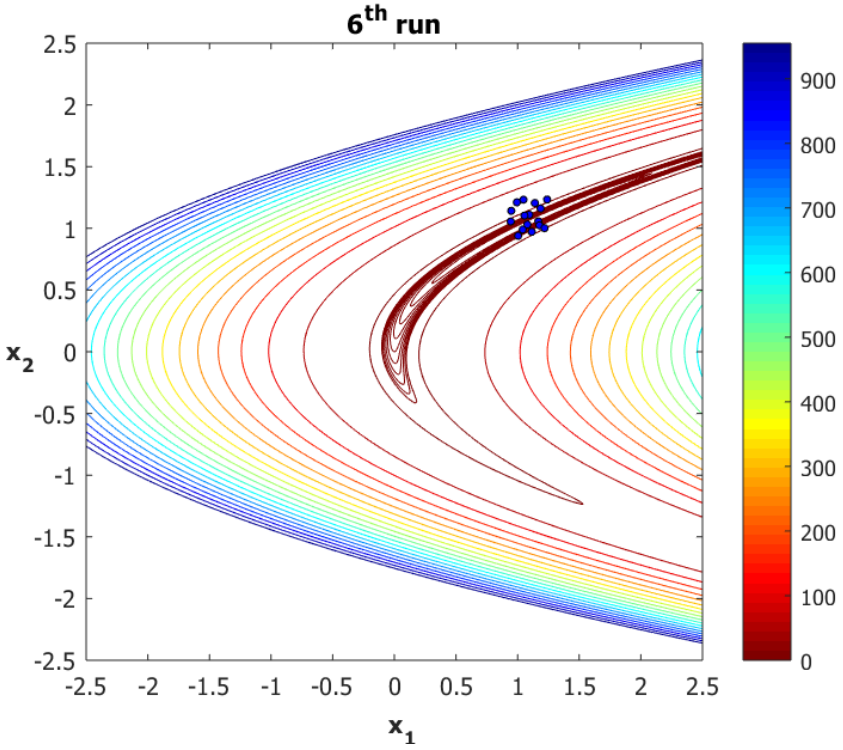


Figure 3.18 Sixth set of data points

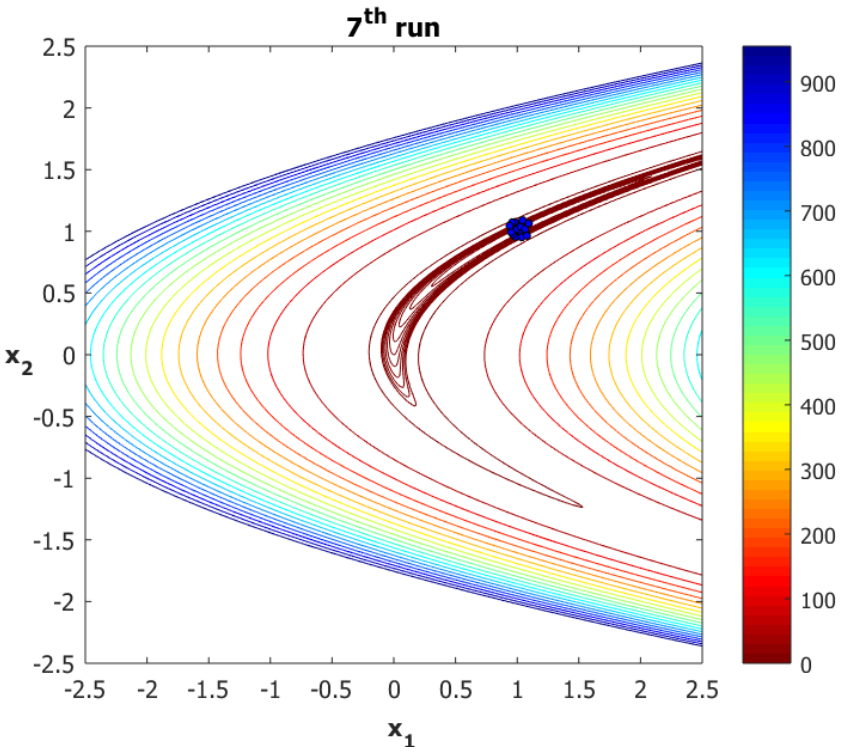


Figure 3.19 Seventh set of data points

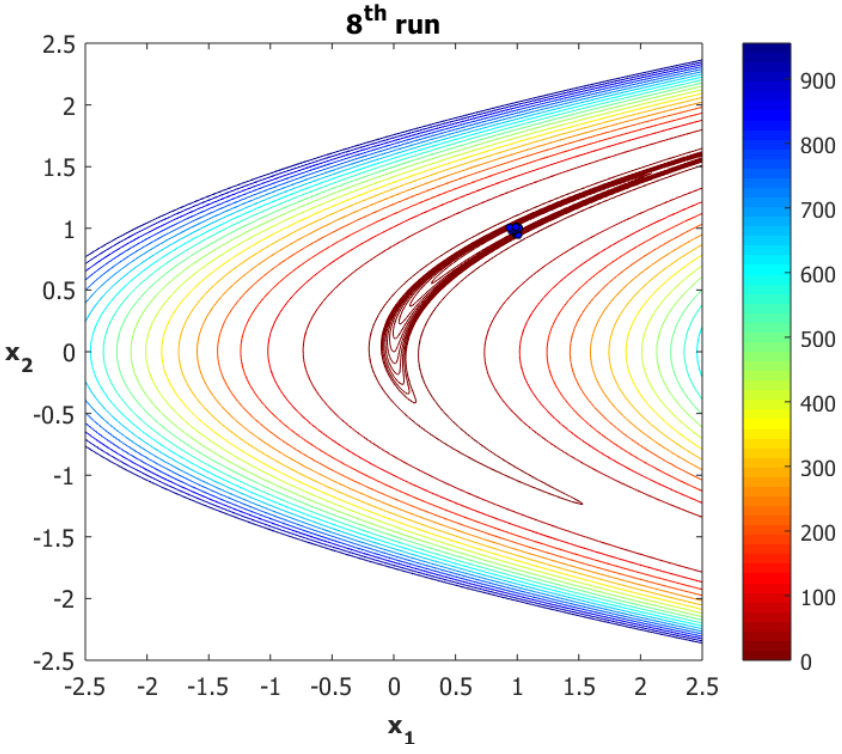


Figure 3.20 Eighth set of data points

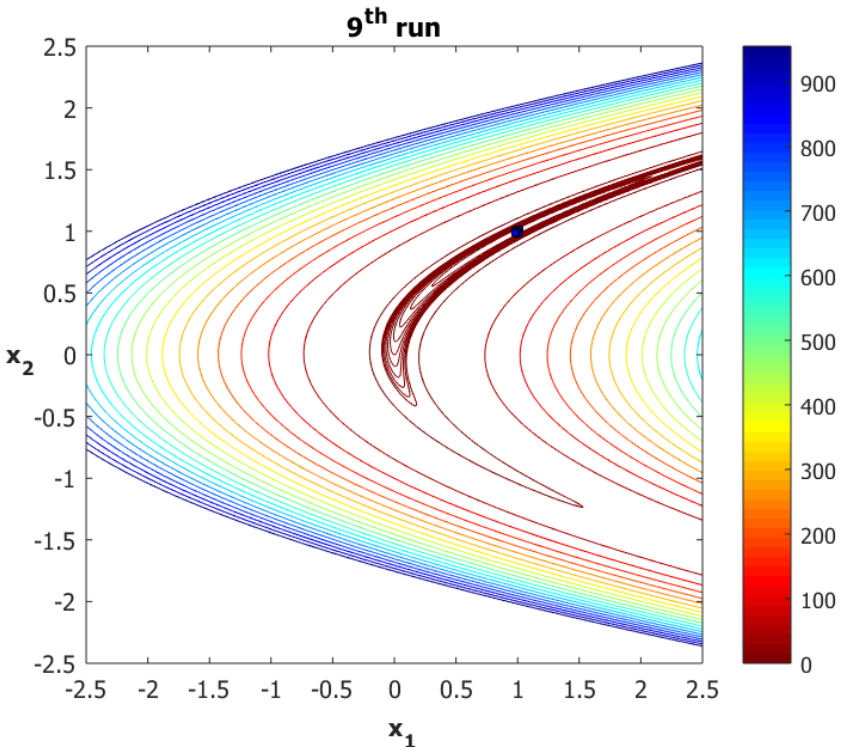


Figure 3.21 Ninth set of data points

The last run, the 9^{th} run, is shown again in Figure 3.22 in detail:

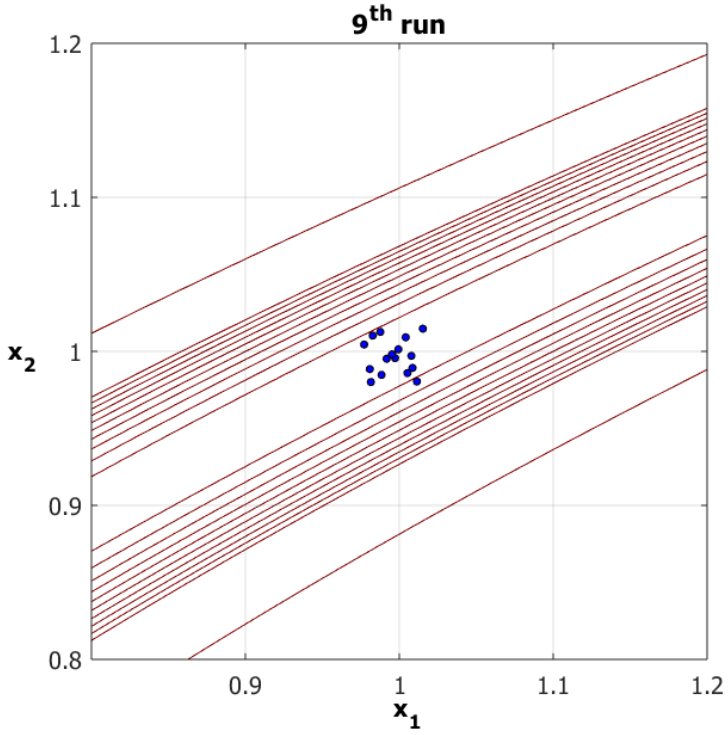


Figure 3.22 Ninth set of data points in detail view

The termination occurs after 132 function evaluations when there is no change on the 4^{th} significant figure; at the point:

$f(x) = 0.00007$ with the variable values $x_1 = 1.00417$ and $x_2 = 1.00912$

Whereas Genetic Algorithm reaches $f(x) = 0.00766$ with the variable values $x_1 = 0.91308$ and $x_2 = 0.83269$ after 8450 function evaluations and 168 generations between the same boundaries.

3.2.2. The results for Rosenbrock Function for 7 variables

To increase the complexity the number of variables is also increased. For this section the number of variables is selected as 7. The number of the training data points is selected with considering the efficient range; as dealt before from the range: $k\,x\,m \leq n \leq k^m$. It may be advantageous to choose a number which is the power of k in order to have uniformly distributed data points in intervals. Thus, according to this inequality $(2\,x\,7 \leq n \leq 2^7)$, it saves the run time and the memory to decide on the lowest value; as $n = 16$.

For 7 variables and 2 intervals there exist $(2k)^m = (2\,x\,2)^7 = 16384$ possible patterns in total.

Referring to the trend analyses again, the first patterns of the 6 classes are chosen from the *successful patterns*, and also the first patterns of the 4 classes are chosen from the *possible successful patterns* for the next run. The 16 data points of the first run are tabulated with the related resultant values in the Table 3.14:

Table 3.14 Design variables with the results for the first training set (m=7)

x_1	x_2	x_3	x_4	x_5	x_6	x_7	f(x)
0.6227	-0.1237	1.0679	0.2338	-0.1481	0.5483	0.2874	255
-0.1314	0.0901	0.0597	0.9998	-0.3585	-0.3670	-0.6343	375
-0.8414	-0.7113	-0.0923	-0.2456	1.5044	0.6850	1.9951	944
0.0794	0.8658	-1.3095	1.8634	0.6080	-0.7601	1.5199	1547
-1.6944	0.5094	-0.3551	-0.5707	-1.9227	1.9676	2.3661	1696
1.5517	1.6179	1.8152	0.6954	2.2401	1.7125	0.9075	2617
-1.2828	1.4843	-0.7182	2.1461	-1.1624	1.1220	-2.1063	5601
2.2155	-0.6165	-1.7444	-0.7580	-1.3331	-0.2233	-1.5972	6006
1.8556	1.1403	2.3475	-1.2974	1.1064	1.3844	-1.2423	6317
0.7023	2.2696	-1.0468	-1.7823	0.1300	-1.2395	-2.3716	7618
-2.3811	-1.3330	1.4753	-1.1354	1.7886	-1.7204	0.4675	9106
-1.1620	2.0591	0.6763	-2.1722	-1.6115	-2.0663	1.1608	9191
-2.0022	-2.0409	0.5160	1.5531	2.1456	-1.4854	-0.0897	9425
-0.3508	-1.5912	-2.4703	0.3586	0.9360	2.3738	-0.5058	10178
1.1185	-1.1639	-1.9998	2.2936	-2.2847	-2.3668	1.7658	14958
1.9065	-2.4749	2.0138	-2.4985	-0.8664	0.1704	-1.2517	15000

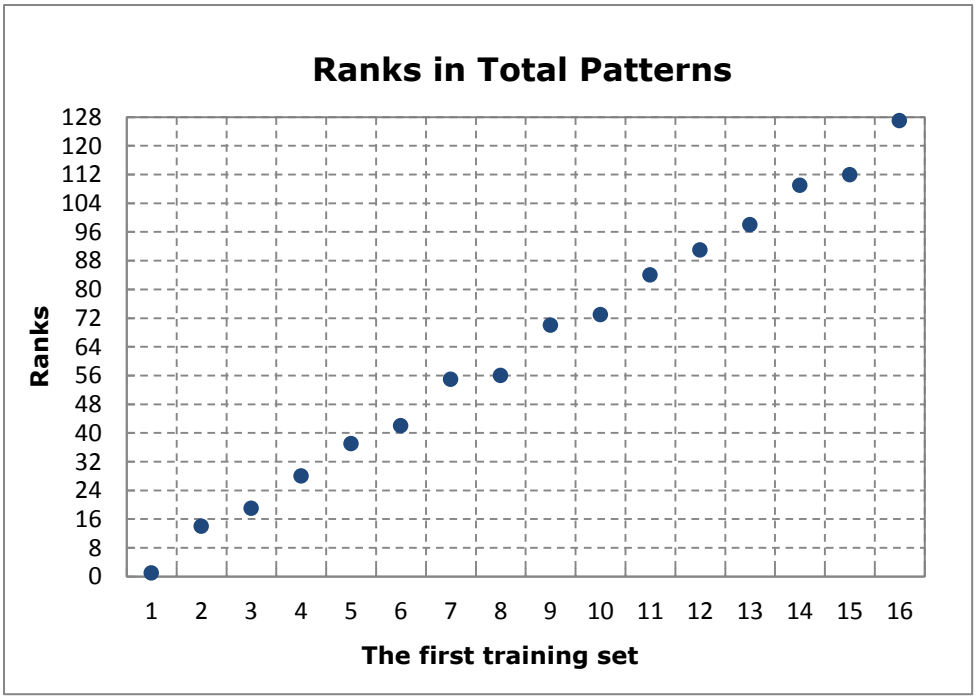


Figure 3.23 Ranks of the first training set

Table 3.14 and Figure 3.23 illustrate together how evenly the first training set is distributed on the search space. The 16 data points for the first run are selected accordingly that, each variable has values distributed to the 16 sublimits between the predetermined upper and lower limits. With the applied dimension reduction method, the variables are evenly matched as a result.

Out of 16384 patterns, 172 patterns are tried at the first run between the lower and upper bounds, -2.5 and 2.5 respectively. The 110 patterns out of the 172 patterns are found as successful and the rest 76 patterns are as unsuccessful. Actually, 14 patterns are common. Further, there are 16212 untried patterns which are distinguished by PNN into $n-1=15$ classes, which have probability to be successful. The 1^{st} patterns of the first 6 successful classes and the 1^{st} patterns of the possible successful patterns are gathered. Eventually, the total promising patterns become:

Table 3.15 The promising patterns for the next run

x_1		x_2		x_3		x_4		x_5		x_6		x_7	
0	-1	-1	0	0	1	0	1	-1	0	0	1	0	1
1	0	0	1	0	1	0	1	-1	0	-1	0	-1	0
-1	0	1	0	-1	0	1	0	0	1	0	1	0	1
0	-1	0	1	-1	0	0	1	0	1	-1	0	0	1
-1	0	0	1	-1	0	1	0	-1	0	0	1	0	1
0	1	0	1	0	1	0	1	0	1	0	1	0	1
-1	0	-1	0	0	1	0	1	-1	0	0	1	0	1
-1	0	0	1	0	1	0	1	1	0	0	-1	0	1
-1	0	0	1	1	0	0	1	0	1	0	1	0	1
0	1	0	-1	-1	0	0	1	0	1	0	1	0	1

Keeping in mind that, the global minimum of the "Rosenbrock" Function is at the point $f(x) = 0$, where $\forall\, x = 1$, the right patterns should be in the interval of 0 to 2.5 at this stage. The grey coloured cells on the table actually indicate the right selections. The 6^{th} pattern is expected to be the successful one; however this fact is also be detected automatically by the algorithm after summing up the normalized $min\,\&\,mean$ values of the data points at the next run. Table 3.16 lists the patterns, which are ranked according to the sum.

Table 3.16 The rank of the promising patterns

	min	**mean**	**sum**
6th pattern	0.363	0.000	0.363
8th pattern	0.062	0.367	0.429
10th pattern	0.190	0.350	0.540
7th pattern	0.000	0.666	0.666
9th pattern	0.277	0.411	0.688
4th pattern	0.795	0.590	1.385
1st pattern	1.000	0.635	1.635
5th pattern	0.762	1.000	1.762
2nd pattern	0.855	0.985	1.840
3rd pattern	0.960	0.987	1.947

Thus, from the first set of data listed in Table 3.14, the promising patterns for the next run are selected and shown in Table 3.15. Out of these 10 promising patterns, 6^{th} pattern is selected based on its sum value as presented in the Table 3.16.

The next run continues within the intervals that the 6^{th} pattern has. The process continues until reaching the termination criteria. The result is found after 1776 function evaluations at the point $f(x) = 0.0004$.

A comparison is done with Globex algorithm which was presented by Jacob [52]. The developed algorithm at [52] is also in the group of direct search methods and has similar characteristics like using directions based on the previously produced data points, besides not requiring the derivatives of the function. Starting from an initial point, and if it is needed also with constraints, it finds out the global minimum of a function or functional. It has wide field of application (application not limited to linear, quadratic or convex/concave functions). Moreover, it requires low memory and it is highly fast compared to numerous existing methods [52]. Thus, it has high priority and is highly reasonable to take as a reference.

Introducing the starting point as:

Table 3.17 The starting point for Globex Algorithm

x_1	x_2	x_3	x_4	x_5	x_6	x_7
-1.2	1.0	-1.2	1.0	-1.2	1.0	-1.2

Also the starting step sizes for the variables as:

Table 3.18 The starting step sizes for Globex Algorithm

dx_1	dx_2	dx_3	dx_4	dx_5	dx_6	dx_7
0.1	0.1	0.1	0.1	0.1	0.1	0.1

And, the constraints as:

$$x_2 < 0 \; and \; (x_1 + \; x_2) > 1$$

Then, the Globex algorithm [52] converges to the point $f(x) = 4.16920$ after 12625 function evaluations.

Genetic Algorithm is also used because of its popularity in optimization. It is based on the principles of natural genetics and selection. It does not use the derivatives in the search procedure but uses only the values of the objective function. Instead of a single starting point, a population of points (trial design vectors) is used for initiating the procedure. Since a high number of distributed candidate points is used it is less likely to stack in a local minimum [43].

For Rosenbrock function with 7 variables, while Genetic Algorithm has the same lower and upper bounds as the Cavus algorithm, it converges to the point $f(x) = 2.8815$ after 15400 function evaluations and 76 generations.

The results for those three algorithms are gathered in Table 3.19 for a clear comparison:

Table 3.19 The results for the "Rosenbrock" Function with 7 variables

Algorithm	x_1	x_2	x_3	x_4	x_5	x_6	x_7	**f(x)**	**Function evaluations**
Cavus	1.0004	0.9995	0.9992	0.9993	0.9997	0.9996	0.9993	0.0004	1776
Globex	0.6175	0.3825	0.1580	0.0353	0.0114	0.0100	0.0001	4.1692	12625
Genetic	0.8341	0.6922	0.4779	0.2270	0.0605	0.0189	0.0109	2.8815	15400
Exact values	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**0.0000**	

As a result, for the Rosenbrock function with the 7-variable case, the Cavus algorithm has the lowest function evaluations and the best convergence to the optima when compared with the Globex and the Genetic Algorithm. When compared with the Genetic Algorithm, this saves 89% of the analyses and thus the required computational time.

3.2.3. The results for the "Rosenbrock" Function for 14 variables

For this part the number of variables is increased as 14. The number of the training data points is selected depending on the inequality: $k\,x\,m \leq n \leq k^m$. As mentioned before it is also advantageous to choose a number which is the power of k and also one of the multipliers of m, in order to have a better multidimensional distribution like Latin Hypercube Sampling [81] [82] [83]. Thus, n is decided to be $n = 2^7 = 128$.

For 14 variables and 2 intervals there exist $(2k)^m = (2\,x\,2)^{14}$ possible patterns in total. Here, to decrease the memory usage, zeros are dropped, and then the total patterns are reduced to $2^m = 2^7 = 16384$. This intermediate step can be used for the problems that have high number of variables to gain the computational performance against the compromise on the result.

Referring to the trend analyses again, the first patterns of the first 6 classes are chosen from the *successful patterns*, and also the first patterns of the first 4 promising classes are chosen from the *possible successful patterns* for the next run. The lower and upper bounds are -2.5 and 2.5 respectively as before. Because of the high number of data the intermediate steps are not tabulated here. After the first run, the 1^{st} patterns of the first 6 successful classes and the 1^{st} patterns of the first 4 promising classes are gathered in Table 3.20. Eventually, the total promising patterns become:

Table 3.20 The promising patterns for the next run $m = 14$

x_1		x_2		x_3		x_4		x_5		x_6		x_7		x_8		x_9		x_{10}		x_{11}		x_{12}		x_{13}		x_{14}	
0	1	0	1	0	1	0	1	0	1	-1	0	0	1	0	1	0	1	0	1	0	1	0	1	-1	0	0	1
0	1	0	1	0	1	0	1	-1	0	-1	0	0	1	0	1	0	1	0	1	0	1	-1	0	-1	0	0	1
0	1	-1	0	-1	0	0	1	0	1	-1	0	-1	0	0	1	-1	0	-1	0	0	1	0	1	-1	0	-1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	1	0	1	-1	0	-1	0	-1	0	0	1	0	1	0	1	0	1	-1	0	-1	0	-1	0	0	1
-1	0	-1	0	0	1	0	1	-1	0	0	1	-1	0	-1	0	0	1	-1	0	-1	0	0	1	-1	0	0	1
0	1	0	1	0	1	0	1	-1	0	0	1	0	1	0	1	-1	0	-1	0	0	1	0	1	0	1	0	1
0	1	0	1	0	1	0	1	-1	0	0	1	0	1	0	1	-1	0	0	1	0	1	-1	0	-1	0	0	1
0	1	-1	0	0	1	0	1	0	1	0	1	-1	0	0	1	-1	0	-1	0	0	1	0	1	-1	0	-1	0
0	1	0	1	0	1	0	1	-1	0	0	1	0	1	0	1	-1	0	0	1	0	1	0	1	0	1	0	1

The global minimum of the "Rosenbrock" Function is at the point $f(x) = 0$, where $\forall x = 1$. Then, the right boundaries of the intervals should be in the range 0 and 2.5. The grey coloured cells on the Table 3.20 actually indicate the right selections. The 4^{th} pattern is expected to be the successful one, it is also be selected automatically by the algorithm after summing up the normalized $min\,\&\,mean$ values of the data points at the next run.

Since in Table 3.21 the 4^{th} pattern has the lowest sum, the next run continues within the intervals that the 4^{th} pattern has. The process continues until reaching the termination criterion. The result is found after 10368 function evaluations at the point $f(x) = 0.0627$.

Table 3.21 The rank of the promising patterns $m = 14$

	min	mean	sum
4th pattern	0.108	0.000	0.108
10th pattern	0.000	0.242	0.242
7th pattern	0.121	0.374	0.495
2nd pattern	0.267	0.496	0.763
1st pattern	0.589	0.251	0.840
8th pattern	0.476	0.498	0.974
9th pattern	0.672	0.745	1.417
5th pattern	0.803	0.741	1.545
6th pattern	1.000	0.864	1.864
3rd pattern	0.866	1.000	1.866

Comparisons are done with Globex algorithm and Genetic Algorithm. The starting point for the Globex algorithm is given at the following table.

Table 3.22 The starting point for Globex Algorithm

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}
-1.2	1.0	-1.2	1.0	-1.2	1.0	-1.2	-1.2	1.0	-1.2	1.0	-1.2	1.0	-1.2

Also, the starting step sizes for the variables are:

Table 3.23 The starting step sizes for Globex Algorithm

dx_1	dx_2	dx_3	dx_4	dx_5	dx_6	dx_7	dx_1	dx_2	dx_3	dx_4	dx_5	dx_6	dx_7
0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

And, the constraints are given as:

$$x_2 < 0 \; and \; (x_1 + \; x_2) > 1$$

Then, the Globex algorithm [52] converges to the point $f(x) = 11.097$ after 25014 function evaluations.

Genetic Algorithm is also used with a population of points for initiating the procedure. For Rosenbrock function with 14 variables, while Genetic Algorithm has the same lower and upper bounds as the Cavus algorithm, it converges to the point $f(x) = 11.971$ after 19000 function evaluations and 94 generations.

The results for those three algorithms are gathered in Table 3.24 and Table 3.25 for comparison:

Table 3.24 The resultant variable values for Rosenbrock Function with 14 variables

Algorithm	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	f(x)
Cavus	1.0099	1.0050	0.9969	1.0049	0.9984	1.0038	0.9947	1.0091	0.9968	1.0044	0.9989	0.9995	0.9990	0.9932	0.063
Globex	0.6166	0.3834	0.1587	0.0355	0.0114	0.0102	0.0102	0.0102	0.0102	0.0102	0.0102	0.0102	0.0100	0.0001	11.097
Genetic	0.4225	0.1753	0.0277	0.0156	0.0229	0.0202	0.0301	0.0084	0.0309	0.0124	0.0092	0.0106	0.0059	0.0052	11.971
Exact values	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**0.000**

Table 3.25 The results for Rosenbrock Function with 14 variables

Algorithm	f(x)	Function evaluations
Cavus	0.063	10368
Globex	11.097	25014
Genetic	11.971	19000

As a result, the Cavus algorithm has the lowest function evaluations and the best convergence to the optima for the Rosenbrock function with 14 variables.

These three algorithms with their $f(x)$ values and the number of function evaluations are tabulated with changing number of variables in the Table 3.26. To have a better understanding they are also presented with the graphs in Figure 3.24 and Figure 3.25.

Indeed, it is shown that, the two targets (to have a better accuracy and less number of function evaluations) are reached by the Cavus algorithm. The accuracy of the Cavus algorithm is pretty good than other two algorithms for the increasing variable numbers for this test case. Even though, the inclination of the number of function evaluations has a tendency to increase progressively with the increasing number of variables, the Cavus algorithm has still better results than the other two algorithms. When compared with the Genetic Algorithm, this still saves 45% of the analyses and thus the required computational time. Correspondingly, the improvements may be done in future works.

Table 3.26 Comparison of the algorithms with the changing variable numbers

	Algorithm	f(x)	Function evaluations
m = 14	Cavus	0.063	10368
	Globex	11.097	25014
	Genetic	11.971	19000
m = 10	Cavus	0.015	2912
	Globex	7.138	17662
	Genetic	7.800	18000
m = 7	Cavus	0.000	1776
	Globex	4.169	12625
	Genetic	2.882	15400
m = 2	Cavus	0.000	132
	Globex	0.146	4078
	Genetic	0.008	8450

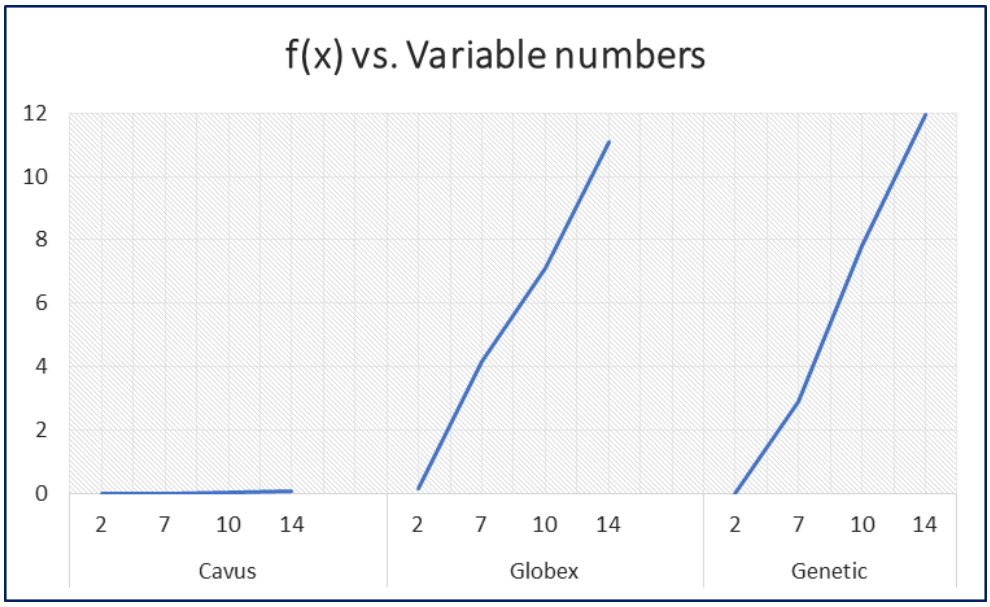


Figure 3.24 Comparison of the algorithms for Rosenbrock function (Results)

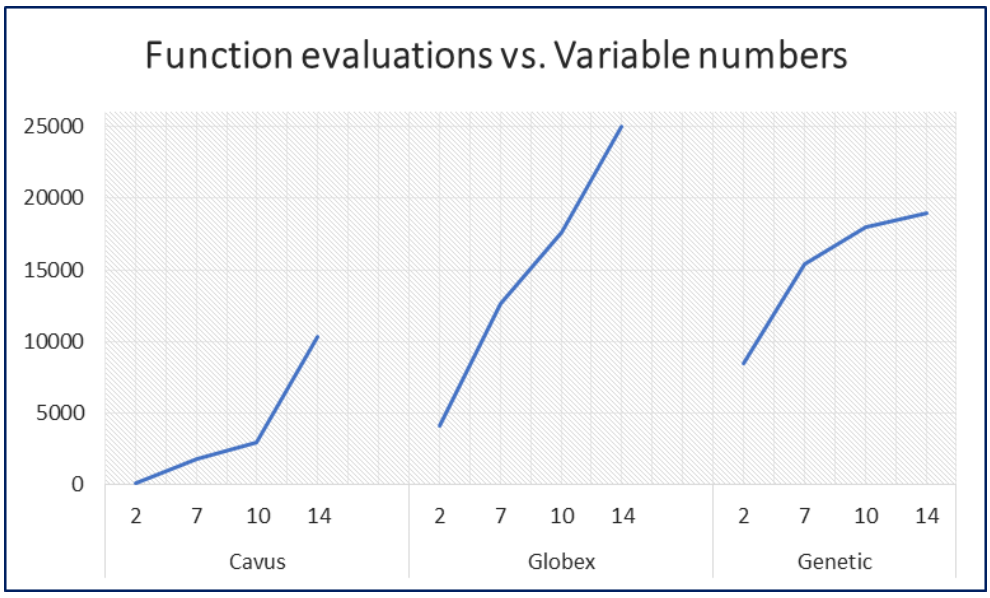


Figure 3.25 Comparison of the algorithms for Rosenbrock function (Function evaluations)

4. Design Cases

4.1. Aircraft Design

Optimization problems are defined as gathering many challenging variables and obtaining the best solution space. Although one may not have any time constraint the selected search method directly affects the quality of the results. Meanwhile one technique may be successful for one type of problem whereas it may fail in another type of problem. Because of that, researchers try to develop a common tool that can adapt to different situations. Especially for the complex projects executed between different scientist, or departments, or even different companies it becomes a must a unique tool controllable by everybody to catch the better solutions.

Aircraft design is one of the challenging but indeed more pleasurable subject area for discovering the deep math and physics of life. Naturally profiting from the broad area of science, it is very convenient to be taken as an application model for multidisciplinary and multiobjective design optimization.

Design optimization of air vehicles is a complex process with depending on many design variables and the highly non-linear physics models. Besides that, the optimization process of an air vehicle has relatively fewer local minimums with higher variable numbers while considering other kind of optimization problems which have many local gradient changes. An artificial intelligence technique is applied here to reduce divergence relative to well-known algorithms like Genetic Algorithm and Simulated Annealing, with the required number of runs for convergence as the objective for changing number of design variables.

The motivation of this study has many bases. At first, an aircraft mission is a perfect closed loop process with the law of conservation of energy. Clearly, it has standard segments and each segment and its requirements – inputs and outputs – are well-known. In other words, this closed loop system is more conservative than many statistical problems; and it is worth to use directed search methods for aircraft design. This means that, aircraft design should be handled in its own nature while using optimization algorithms. This extended area deserves to find out an optimization method that is special to its own characteristics. Mostly, researchers in aerospace use search techniques that need huge number of function evaluations; indeed there must be better ways to have an improvement.

In addition to having a closed loop process feature, at the beginning of the design process researches may have huge number of data from wind tunnel tests as well as from flight tests at least for the conventional aircraft. All of these data with engineering sense of experienced engineers, which is inseparable throughout a design process, have

a great impact on all of the design phases. Learning from data, which is called as data mining, with the combination of engineering sense has very precious fine tuning effect on a design with knowledge based methods. Some of the complex interactions can also be estimated with relative magnitudes just from the data at hand at the beginning of the design process, which accelerates the calculations and force the results to converge to better values. With more philosophy, these complex but systematic machines, aircraft, with dependent and independent variables, with known interactions to each other in an optimization process can be improved with the algorithms which mimic pattern classification property of human intelligence, i.e. Artificial Intelligence (AI), for the early as well as ensuing stages of design.

In this study, Probabilistic Neural Networks, which was introduced by [14], is employed in the observation part of the optimization algorithm for its success on classification and pattern recognition.

For the completeness of the entire picture in the following sections aircraft design models are introduced for the use case demonstration of the Cavus algorithm, which have been developed in a previous study, [68].

4.1.1. Conceptual design of an unmanned supersonic aircraft

As a first design case, a multi-role unmanned supersonic aerial vehicle (UCAV) conceptual design optimization problem is addressed, which is also dealt with in detail in [68]. This aircraft type is selected because of its

- challenging specifications,
- property being good at dynamics and
- capacity for multi role and varying missions.

The aircraft shall be capable to act according to its role, maneuver, resist high maneuver loads, carry payload, drop payload, cruise at high altitudes and have long endurance without refuelling. It will autonomously locate its targets, navigate autonomously, but also be able to be flown by a remotely controlled pilot with a ground station system.

It is planned to have a simple and fixed shape during the flight. In other words, except the control surfaces (flaps, ailerons, elevator and rudder) it shall not have any moving components (i.e., fixed swept wing and fixed tail configuration).

The mission is composed of 14 segments: engine start and warm-up, taxi, take-off, climb, cruise-out, loiter, descent, dash-out, action, dash-in, climb, cruise-in, descent, landing-taxi and shutdown. The segment characteristics are described in details in the next section.

It should be able to carry payload until the end of the flight if the mission is aborted. This payload will be carried externally under the wings. The detailed placement is not addressed in this work. It is assumed that they will be installed properly without affecting the static and dynamic stability of the unmanned supersonic aircraft.

The positions of the wings, horizontal and vertical tails, and landing gears with respect to the fuselage are calculated in the Aircraft Design Code prepared. The fuel will be carried internally in the wings; no external fuel tanks are planned.

The engine sizing is automatically done in the code which gives its dimensions and its thrust. The calculations are based on a rubber engine with turbofan characteristics which has constant bypass ratio and specific fuel consumption.

Landing gears are planned to be tricycle and retractable, and designed to find the dimensions according to the changing aircraft configurations. The placements are changed for different aircraft configurations with centre of gravity.

Cost is another important parameter in aircraft design and it is taken as the objective function in this study. A subroutine in the code calculates the total acquisition cost.

Proper mathematical models are selected and coded in separate subroutines as summarized in the next sections. The main program picks up values of some parameters from an input file and then does the calculations; afterwards gives results to an output file. While calculating, subroutines are called by the main program in an order assigned by the programmer, before. Throughout the program some parameters need to be updated, this is coped with calling the required subroutine in the concerning subroutine again.

How the subroutines of the aircraft design part communicate with each other are shown by a flow chart in Figure 4.1.

Propulsion System and Aerodynamics subroutines are called more frequently because of frequently updated variables like Mach number, thrust and induced drag.

The detailed information about the conceptual design phases with equations, requirements, constraints and constants are given in the following sections.

While calling the subroutines, design variables are carried thought the process. All the variables are tabulated in detail in the Table 4.1.

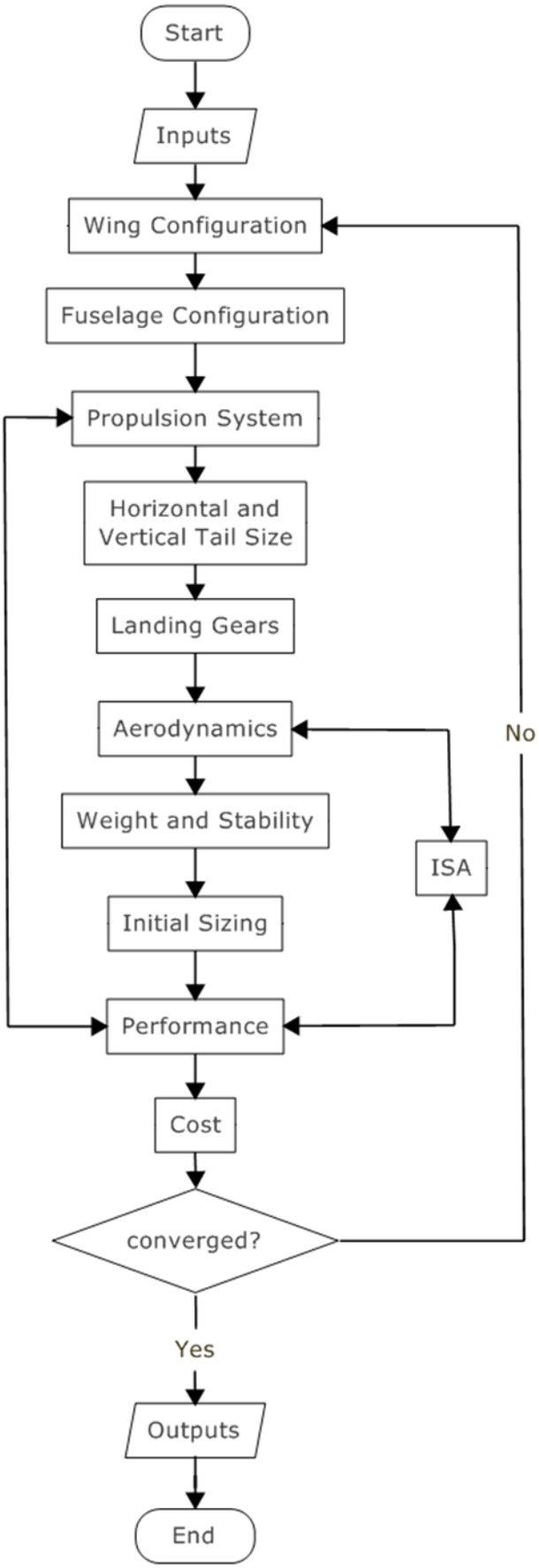


Figure 4.1 Flowchart of the Aircraft Design Algorithm

Table 4.1 Aircraft Design part inputs and outputs

SUBROUTINES	Inputs	Outputs
ISA (Atmospheric Properties)	H, ρ_0, g_0, a_0	ρ, a
WING CONFIGURATION	$D_{f_{max}}, \Lambda_w, b, AR, \Gamma_w, (t/c)_w$	$\Lambda_{LE}, c_{r_{fw}}, \lambda_{fw}, t_{c_{fw}}, \tau_{fw}, AR_{wet}, \lambda, w_{f_{max}}, (t/c)_{t_w}, (t/c)_{r_w}, S, c_r, c_t, \bar{y}, \bar{C}, \bar{X}, S_{net_w}, S_{wet_w}, (L/D)_{max}, V_f, w_{f_{max_w}}, S_{cs_w}$
FUSELAGE CONFIGURATION	$D_{f_{max}}, W_0, w_{f_{max}}$	$h_{f_{max}}, L_f, l_N, l_A, l_M, A_{max}, FinenessRatio, Volume, S_{wet_f}$
PROPULSION SYSTEM	M, W_0, BPR, SFC	$T, T_{av}, L_{eng}, D_{eng}, SFC_c, T_{cruise}$
HORIZONTAL and VERTICAL TAIL	$D_{f_{max}}, L_f, b, S, w_{f_{max}}, \bar{C}, l_A, \Lambda_w, (t/c)_{VT}, (t/c)_{HT}, \Lambda_{VT}, AR_{VT}, AR_{HT}, \lambda_{VT}, \lambda_{HT}, \Gamma_{VT}, \Gamma_{HT}, V_{VT}, V_{HT}, l_{VT_{co}}, l_{HT_{co}}$	$l_{VT}, l_{HT}, \Lambda_{HT}, S_{VT}, S_{HT}, h_{VT}, c_{r_{VT}}, c_{t_{VT}}, \bar{C}_{VT}, \bar{z}_{VT}, c_{r_{fVT}}, \lambda_{fVT}, (t/c)_{r_{VT}}, (t/c)_{t_{VT}}, (t/c)_{fVT}, \tau_{fVT}, S_{net_{VT}}, S_{wet_{VT}}, b_{HT}, c_{r_{HT}}, c_{t_{HT}}, \bar{C}_{HT}, \bar{y}_{HT}, w_{f_{max_{HT}}}, c_{r_{fHT}}, \lambda_{fHT}, (t/c)_{r_{HT}}, (t/c)_{t_{HT}}, (t/c)_{fHT}, \tau_{fHT}, S_{net_{HT}}, S_{wet_{HT}}$
LANDING GEARS	W_0	$D_{nosewheel}, D_{mainwheel}, F_M, F_N, w_{nosewheel}, w_{mainwheel}$
AERODYNAMICS	$H, \rho, a, V_{cruise}, W_{payload}, \Lambda_{LE}, \Lambda_w, W_0, L_f, D_{f_{max}}, A_{max}, S_{wet_f}, b, S, AR, \bar{C}, S_{wet_w}, V_{VT}, V_{HT}, l_{VT}, l_{HT}, S_{VT}, S_{HT}, S_{wet_{VT}}, S_{wet_{HT}}, \bar{C}_{VT}, \bar{C}_{HT}$	$\rho_{cruise}, a_{cruise}, \#_{bom\_big}, \#_{bomb\_aim9}, M_{cruise}, q_{cruise}, \beta, Re_{fus}, Re_{cutoff.fus}, C_{f.fus}, FF_{fus}, Q_{fus}, C_{D0_{fus}}, Re_w, Re_{cutoff.w}, C_{f.w}, FF_w, Q_w, C_{D0_w}, Re_{HT}, Re_{cutoff.HT}, C_{f.HT}, FF_{HT}, Q_{HT}, C_{D0_{HT}}, Re_{VT}, Re_{cutoff.VT}, C_{f.VT}, FF_{VT}, Q_{VT}, C_{D0_{VT}}, (D/q)_{misc}, C_{D0_{misc}}, (D/q)_{wavesh}, (D/q)_{wave}, C_{D0_{wave}}, C_{D0_{parasitic}}, C_{D0_{Leakage}}, C_{D0_{total_{clean}}}, C_{D0_{total}}, e, K, K_{clean}, Ground\_effect, C_{Lmax_{clean}}, C_{Lac_{max_{TO}}}, C_{Lmax_{landing}}, C_{D0_c}$

Table 4.1 Aircraft Design part inputs and outputs (continued)

SUBROUTINES	Inputs	Outputs
WEIGHT and STABILITY	n_{max} , S , AR , $(t/c)_{r_w}$, λ , V_{HT} , Λ_{LE} , Λ_w , AR_{VT} , λ_{VT} , Λ_{HT} , Λ_{VT} , M , W_0 , T, T_{av} , BPR , S_{cs_w} , $w_{f_{max_{HT}}}$, $D_{f_{max}}$, $h_{f_{max}}$, b_{HT} , S_{VT} , S_{HT} , L_f , l_{VT} , l_{HT} , $D_{nosewheel}$, $D_{mainwheel}$, $\overline{C}_{HT}$, $\overline{C}_{VT}$, e , $W_{payload}$, $w_{f_{max_w}}$, L_{eng}	N_z , W_{wing} , $W_{horizontaltail}$, $W_{verticaltail}$, $W_{fuselage}$, $W_{mainlandinggear}$, $W_{noselandinggear}$, W_{engine} , $W_{instruments}$, $W_{avionics}$, W_{else} , $x_{cg.engine}$, $x_{cg.fus}$, $x_{cg.else}$, $x_{cg.HT}$, $x_{cg.VT}$, $x_{cg.wing}$, $x_{cg.fuel}$, $x_{cg.payload}$, M_{cg} , W_{cg} , x_{cg} , $x_{cg.mgear}$, $x_{cg.ngear}$, x_{ac_w} , $Static\_margin, W_e$, x_n
INITIAL SIZING	SFC, SFC_c , $(L/D)_{max}$, $w_{f_{max_w}}$, W_e , $W_{payload}$, R_{cr} , V_{cruise} , E_{ltr} , V_{max}	W_0 , W_f , W_1/W_0 , W_2/W_1 , W_3/W_2 , W_4/W_3 , W_5/W_4 , $(L/D)_{cr}$, $(L/D)_{ltr}$, W_6/W_5 , W_7/W_6 , R_d , $(L/D)_d$, W_8/W_7 , W_{fuel_1} , W_9/W_8 , W_{f_c} , W_9 , W_{10}/W_9 , W_{11}/W_{10} , W_{12}/W_{11} , W_{13}/W_{12} , W_{14}/W_{13} , W_{15}/W_{14} , W_{15}/W_0 , W_f/W_0 , $W_{0_{initialsizing}}$, W_3
PERFORMANCE	W_0 , $S, AR, C_{L_{max_{clean}}}$, $C_{L_{ac_{max_{TO}}}}$, $C_{L_{max_{landing}}}$, $C_{D_{0_c}}$, $C_{D_{0_{total}}}$, K , SFC , SFC_c , $(L/D)_{max}$, W_{fuel_1} , W_f , $W_{payload}$, W_e , W_9 , ρ_0 , ρ , e , M , W_{15}/W_{14} , W_{15}/W_0 , W_3 , H , W_{10}/W_9 , W_{f_c}	$(W/S)_{TO}$, T/W , T , s_{TO} , $(W/S)_{Landing}$, s_L , $(L/D)_{max_s}$, ROC_{max} , θ_{max} , $Maximum\_ceiling$, T_{ROC} , ROC , $(\sqrt{C_L}/C_D)_{max}$, $RANGE$, V_{max} , M , M_{max} , E_{max}, V_{corner} , n_{max} , R_{min} , ω_{max} , $R_{min_{pull-up}}$, $\omega_{max_{pull-up}}$, $R_{min_{pull-down}}$, $\omega_{max_{pull-down}}$, q_c , $(L/D)_c$, T_{rc} , d , xx
STRUCTURAL LOAD	H , $C_{D_{0_c}}$, T , K , $C_{L_{max_{clean}}}$, W_9 , S , ρ	$V_{manuever}$, K_p , V_{stall_c} , $C_{L_{max_c}}$, n_{max} , q_c
COSTS	W_e , $W_{avionics}$, M , V_{max} , T , $Quantity, FTA$, T_i , R_e , R_t , R_q , R_m	H_e , H_t , H_q , H_m , $Cost_D$, $Cost_F$, $Cost_M$, $Cost_E, Cost_A, COST$

4.1.2. Mission Profile

Figure 4.2 illustrates the planned mission profile for the unmanned supersonic aircraft.

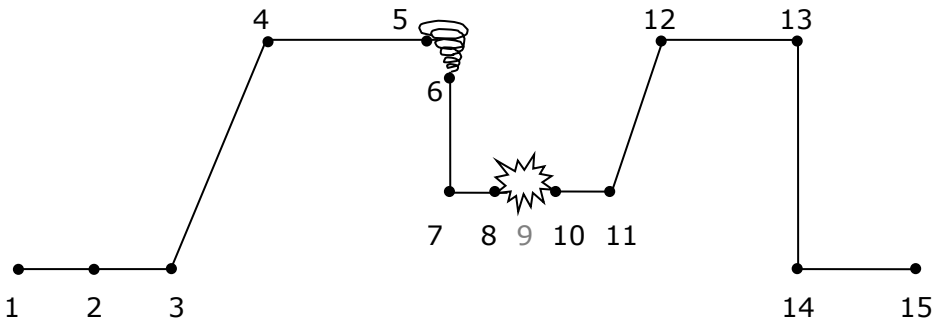


Figure 4.2 Mission segments

Each segment can be described as:

0-1 : Engine Start and Warm-up
1-2 : Taxi
2-3 : Take-off
3-4 : Climb
4-5 : Cruise-out
5-6 : Loiter
6-7 : Descent
7-8 : Dash-out
8-10 : Action
10-11 : Dash-in
11-12 : Climb
12-13 : Cruise-in
13-14 : Descent
14-15 : Landing, Taxi and Shutdown

This is a typical mission profile for UCAVs including all the required segments [2], [84].

In action, the unmanned supersonic aircraft is designed for air-to-air and air-to-ground tasks which include kinds of maneuvers that result in high g forces.

The details of the mission segments may be understood better in the following sections.

4.1.3. Initial Sizing

Airplanes must normally meet very stringent range, endurance, speed and cruise speed objectives while carrying a given payload. It is important, to be able to predict the minimum airplane and fuel weights needed to accomplish a given mission [84].

Besides, a typical unmanned supersonic aircraft mission includes an action segment consisting of either certain number of turns or a certain number of minutes at maximum power, a payload drop, a cruise back and a loiter. The payload drop refers to the firing of the required equipment also [2].

In this design case, aerial refuelling and external fuel tanks are not considered and it is assumed that the fuel consumed is only which the wings can hold.

While estimating the mission fuel fractions, reserved and trapped fuel as required by civil or military design specifications are taken into consideration by 6% percentage of the used fuel at the end of the mission.

Under these assumptions, the fuel fractions are found for each segment with the help of the given tables and the equations in referenced design books [2], [84].

Under the light of [2], lift to drag ratios for loiter and cruise were decided for maximum performance. To maximize loiter efficiency it is assumed that the aircraft will be able to fly approximately with the velocity that gives maximum lift to drag ratio, L/D. Similarly, it is able to fly with the velocity that requires a L/D, which is 86.6% of the maximum L/D for the most efficient cruise [2]:

$$\left(\frac{L}{D}\right)_{cr} = 0.866 \left(\frac{L}{D}\right)_{max} \tag{4.1}$$

and

$$\left(\frac{L}{D}\right)_{ltr} = \left(\frac{L}{D}\right)_{max} \tag{4.2}$$

The total fuel fraction excepting action segment is calculated with:

$$\frac{W_f}{W_0} = 1.06 \left(1 - \frac{W_{15}}{W_0}\right) \tag{4.3}$$

For the action segment the available fuel is found by considering the maximum fuel capacity of the wings and the required fuel for other segments. And also, 6% more fuel for reserved and trapped fuel is also taken into account.

Then, the fuel burned during the action segment becomes:

$$W_{fc} = \frac{W_{f\max w} - W_f}{1.06} \tag{4.4}$$

The payload is carried externally under the wings. Two kinds of weapons were selected for the mission: a weapon below 2000 lb and Aim9 (Sidewinder) (200 lb). However, it was planned that there is a time at which all the payload would have been dropped, point 9. One other assumption was made for the segment 8-9 as at payload drop the fuel consumed is little compared to other segments [84].

Then, W_0 is found by iterating the following equation:

$$W_0 = \frac{W_{payload}}{\left(1 - \frac{W_f}{W_0} - \frac{W_e}{W_0}\right)} \tag{4.5}$$

4.1.4. Wing Configuration

While configuring the wing, the first decision is the selection of the airfoil. The airfoil selection should be made carefully because the airfoil affects the cruise speed, takeoff and landing distances, stall speed, and overall aerodynamic efficiency during all phases of flight. In supersonic flow, the aircraft encounter bow shocks, which results in extra drag, also. To prevent this, an airfoil which has a sharp or nearly sharp leading edge should be used and/or wing sweep can be given [2].

The competitor aircraft like F-16 and F111 use NACA six-digit series airfoils [85]. These airfoils have lower drag at higher speeds compared to four or five digits series. Among these airfoils $NACA\ 64A210$ was chosen for the unmanned supersonic aircraft, whose data was taken from [84], and was illustrated in Figure 4.3.

Figure 4.3 NACA 64A210

It is proposed that the airfoil can be chosen from an airfoil database which can be inserted to program and may be left as a future work. In this work, it is assumed that there is an airfoil which can meet the calculated performance parameters. For that reason, in the program some checks are made by the programmer in order to stay in the feasible region. So, $NACA\ 64A210$ may also be considered as a sample for this purpose.

Another concern was the leading edge sweep for this supersonic aircraft in order to reduce the drag. The leading edge sweep is calculated for straight trailing edge as in Figure 4.4 from [2].

It is assumed that the length of fuselage-wing intersection on the fuselage width is:

$$w_{f\ max} = 0.5 D_{f\ max}$$

(4.6)

So that, in order to be able to get the desired intersection length, the wing position on the fuselage was determined as above the centre line of the fuselage, which means a high wing configuration.

And the control surface area is calculated according to the competitor aircraft as:

$$S_{csw} = 0.1 S$$

(4.7)

Fuel capacity of the wing is found from Figure 4.4 [84] [86]:

$$V_{Fuel\ tank\ w} = 0.54 \frac{S^2}{b} \left(\frac{t}{c}\right)_r \frac{1 + \lambda\sqrt{\tau} + \lambda^2 \tau}{(1+\lambda)^2}$$

(4.8)

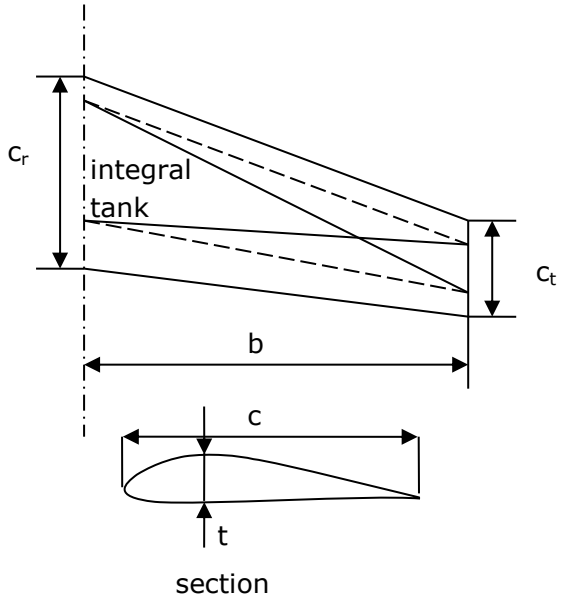


Figure 4.4 Approximation for integral fuel tank volume, available in a linear lofted wing [84] [86]

From [2] fuel density is taken as $0.78\ kg/lt\ (= 780\ kg/m^3)$. Then, the total fuel mass capacity of the wing is found with:

$$W_{f\ \max w} = 780\ V_{Fuel\ tank\ w}$$

(4.9)

4.1.5. Fuselage Configuration

The fuselage is designed to have approximately a circular cross section, to include the engine and also all the instruments that are needed.

The fuselage is composed of a nose, mid-section and aft section as illustrated in Figure 4.5.

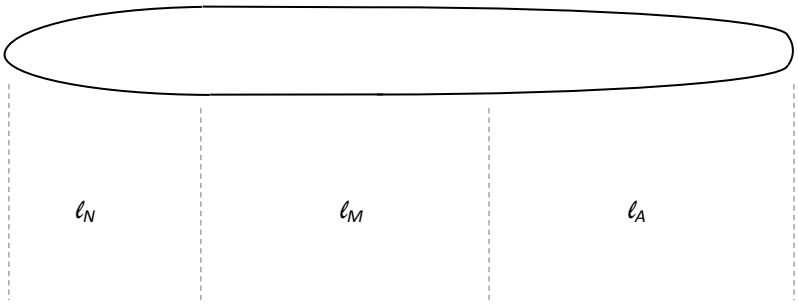


Figure 4.5 Fuselage sections [84]

Where,

$$\lambda_N = 1.75 D_{f\ max} \tag{4.10}$$

$$\lambda_M = L_f - \lambda_N - \lambda_A \tag{4.11}$$

$$\lambda_A = 2.75 D_{f\ max} \tag{4.12}$$

For a fuselage without a circular mid-section ($\text{finenessratio} = \mathrm{L_f}/\mathrm{D_{f_{max}}} \leq 4.5$) :

$$Volume = \frac{\pi}{4} {D_{f\ max}}^2 L_f \left(0.50 + 0.135 \frac{\lambda_N}{L_f}\right) \tag{4.13}$$

The fuselage is configured more in detail while adding other components of the aircraft. They are mentioned in the next sections.

4.1.6. Propulsion System

The engine inlet was planned to be on the nose of the fuselage as illustrated in Figure 4.6 [2]. It is designed as, the optimized unmanned supersonic aircraft only one engine mounted in the fuselage to use. So as, all the related equipment will be covered by the fuselage, also. And it is expected that, the effect of engine weight to the stability of the unmanned supersonic aircraft while maneuvering is minimized with this basic configuration. Further, it is possible to integrate S-shape inlet duct for stealth.

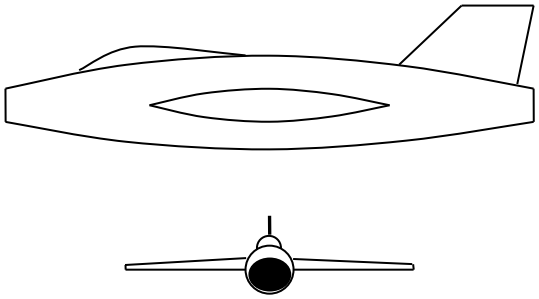


Figure 4.6 Inlet locations - buried engines (Nose) [2]

The thrust to weight ratio directly affects the performance of the aircraft. Since, throughout the mission the fuel is consumed so that weight changes, and thrust to weight ratio is also changing.

Specific fuel consumption is another point that should be concerned. While flying with maximum thrust the specific fuel consumption is calculated as [2] with 20% reduction for next-generation engines:

$$SFC_{\max T} = 0.8[60e^{-0.12BPR}] = SFC_c \tag{4.14}$$

For other mission segments, SFC is taken as constant and equal $0.64\, mg/Ns$ as competitors' engines have, approximately.

For the length and the diameter of the engine the equations from [2] is used with including 20% reduction for next-generation engines.

4.1.7. Horizontal and Vertical Tail Configuration

Tails provide for trim, stability and control. Especially, vertical tail plays a key role in spin recovery. And, these efficiencies can be optimized with the tail configuration. The conventional tail was selected for the unmanned supersonic aircraft because of its simplicity, light weight and adequate stability and control [2].

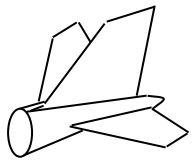


Figure 4.7 Conventional tail [2]

The airfoil of the unmanned supersonic aircraft's tail should be capable of maneuvering in supersonic conditions. For that reason, the airfoil proposed for the wings, $NACA\ 64A210$, is selected for the vertical and horizontal tail of the unmanned supersonic aircraft.

Vertical tail sweep angle, Λ_{VT} , was selected as 45^o for the initial value and is taken as a design variable changing between 35^o to 55^o.

Horizontal tail sweep angle, Λ_{HT} , is thought as 5^o more than the wing sweep, as usual for other aircraft. It was planned that this selection makes the tail stall after the wing, and also provides the tail with a higher Critical Mach Number than the wing, which avoids loss of elevator effectiveness due to shock formation [2].

Like the wing dihedral, the vertical and the horizontal tail dihedrals are taken as $zero$, initially.

While optimizing the unmanned supersonic aircraft, one of the important variables is changing the placements of the vertical and the horizontal tails.

Stability of the aircraft is also affected by the tail because of the lift it produces, its weight and the tail moment arm measured from the center of gravity. However, running the design code with different moment arm values showed that these moment arms really affect the results. Hence, these proportions are also selected as the design variables. As a result tail moment arms are calculated as:

$$l_{VT} = l_{VTco} L_f \tag{4.15}$$

$$l_{HT} = l_{HTco} L_f \tag{4.16}$$

Other variables of tails are calculated depending on the equations at [2], [84], [86], [87].

4.1.8. Landing Gears

For the supersonic unmanned aerial vehicle, retractable tricycle landing gear configuration was selected. Assuming that, nose landing gear carries 10% and main wheels carry 90% of static load of the unmanned supersonic aircraft:

$$F_N = 0.10W_0 \tag{4.17}$$

$$F_M = 0.90\,W_0 \tag{4.18}$$

The placement of the wheels is described in section 4.1.10.

4.1.9. Aerodynamics

The airfoil of the aircraft is planned to be selected from a database as stated. It is assumed that the required lift to drag ratio during the mission can be met. However, in order to be stay in the feasible regions some properties (like lift coefficient) are also calculated for supersonic conditions. Though, in the program some checks are able to be done, and these checks serve the programmer to be able to decide on the limiting values of the constraints at the optimization part. The flexibility of the airfoil also eases the results to spread on a wide region. The used basic aerodynamic equations are taken from [2], [84], [87] and are not needed to be given here in detail.

One of the challenging parameters for the UCAV is maximum velocity, V_{max}, and for a given thrust-to-weight ratio it is directly proportional to $\sqrt{W/S}$. With increasing wing loading, the maximum velocity also increases, accordingly the stalling speed increases, V_{stall}, which is undesirable. The solution to this problem is increasing $C_{L_{max}}$ sufficiently that; in spite of the large W/S, V_{stall} will be acceptable. Thus, to obtain the sufficient increase in $C_{L_{max}}$ high-lift devices are used which make efficient high-speed flight possible [87]. For that reason, Fowler-type triple slotted flap was chosen in contrast to its complexity and high cost. Triple slotted flap is illustrated in Figure 4.8 [2].

Figure 4.8 Triple slotted flap [2]

Ground effect is calculated for one meter above the ground [87]:

$$G = \frac{(16h/b)^2}{1 + (16h/b)^2} \tag{4.19}$$

The control surface limits and the rate limits are listed as for F-16 aircraft [88] in Table 4.2:

Table 4.2 F-16 Control Surface Actuator Models [88]

	Deflection Limit	Rate Limit	Time const.
Elevator	±25.0°	60°/s	0.0495 s lag
Ailerons	±21.5°	80°/s	0.0495 s lag
Rudder	±30.0°	120°/s	0.0495 s lag

4.1.10. Weight and Stability

In conceptual design, it is common to use statistical data from existing aircraft with curve fitting to form empirical weight equations as used in [39]. Because the data of existing UCAV configurations are not open, empirical equations and parameters were used from the open resources. Accordingly, the weight and stability equations were taken from references [2], [84], [87].

Assuming the main wheels carry 90% and the nose wheel carries 10% of static load of the unmanned supersonic aircraft and matching the centre of gravities of the wing and the main wheels:

$$x_{cg.mgears} = x_{cg.wing} \tag{4.20}$$

Then, locating the nose wheel accordingly:

$$x_{cg.ngears} = x_{cg1} - \left[9\left(x_{cg.mgears} - x_{cg1}\right)\right] \tag{4.21}$$

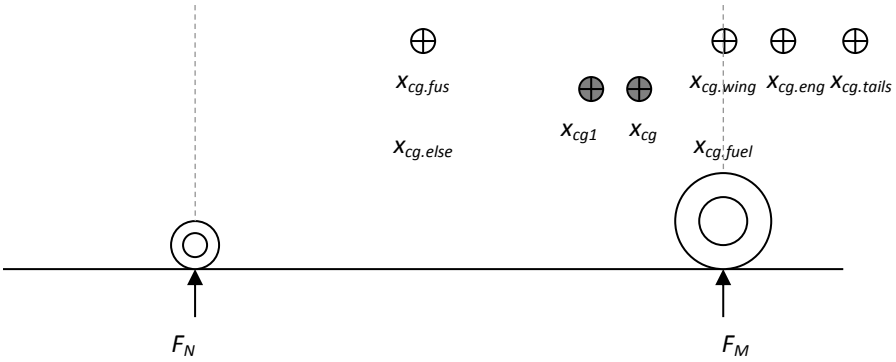


Figure 4.9 Centre of gravity-exaggerated views [87]

The resultant centre of gravity location of the unmanned supersonic aircraft is updated after adding landing gear values.

Then the empty weight of the unmanned supersonic aircraft:

$$W_e = W_{cg} - W_{payload} - W_{f\max w} \tag{4.22}$$

Static Margin

Static margin of a supersonic aircraft is usually changing between -5% and 5%; besides, new multi role supersonic aircraft have even lower static margins. Since, the static margin is one of the important parameter for stability and control it was inserted as a constraint to the code. This is calculated as:

$$Static\ margin = \frac{x_n - \bar{x}}{\bar{c}} \tag{4.23}$$

4.1.11. Performance

The performance calculations were made from references [2], [84], [87].

Takeoff wing loading:

$$\left(\frac{W}{S}\right)_{TO} = \frac{W_0}{S} \tag{4.24}$$

Takeoff distance

$$S_{TO} = \frac{37.5\left(\frac{W}{S}\right)_{TO}}{\sigma\, C_{L_{max}{}_{TO}}\left(\frac{T}{W}\right)_{TO}}$$

(4.25)

And, takeoff distance was selected as one of the constraints in the optimization part.

Wing loading at landing:

$$\left(\frac{W}{S}\right)_{Landing} = \left(\frac{W}{S}\right)_{TO}\frac{W_{15}}{W_0}\frac{1}{W_{15}/W_{14}}$$

(4.26)

Landing distance

$$s_L = 5\left(\frac{W}{S}\right)_{Landing}\left(\frac{1}{\sigma\, C_{L_{max}Landing}}\right) + S_a$$

(4.27)

Landing distance was also selected as one of the constraints in the optimization part.

Where,

$S_a = 137\,m$ was given by [2] for seven degree glideslope, which are related with obstacle clearance and the rate of descent, therefore approximated semi-empirically.

The maximum rate of climb, ROC_{max}

$$ROC_{max} = \left[\frac{(W/S)Z}{3\rho_\infty C_{D_0}}\right]^{1/2}\left(\frac{T}{W}\right)^{3/2}\left[1 - \frac{Z}{6} - \frac{3}{2(T/W)^2(L/D)^2_{max}Z}\right]$$

(4.28)

Maximum ceiling

Maximum ceiling is found with a control loop and caught when ROC equals $zero$. Correspondingly, maximum ceiling was also inserted in the constraints.

Range

$$RANGE = \frac{2}{SFC}\sqrt{\frac{2}{\rho_\infty S}}\frac{\sqrt{C_L}}{C_D}\left(\sqrt{W_i} - \sqrt{W_{i+1}}\right)$$

(4.29)

Maximum speed is calculated at 40000 ft

$$V_{max} = \left\{ \frac{[(T_A)_{max}/W](W/S) + (W/S)\sqrt{[(T_A)_{max}/W]^2 - 4C_{D_0}K}}{\rho_\infty C_{D_0}} \right\}^{1/2} \tag{4.30}$$

Endurance

$$E = \frac{1}{SFC}\frac{L}{D} ln \frac{W_i}{W_{i+1}} \tag{4.31}$$

Corner Velocity

In order to get the minimum instantaneous turn radius and the maximum instantaneous turn rate the unmanned supersonic aircraft should fly with the corner velocity, which is used while calculating maneuver radiuses and rates below:

$$V_{corner} = \sqrt{\frac{2n_{max}}{\rho_\infty C_{L_{max}}}\frac{W}{S}} \tag{4.32}$$

Minimum turn radius (at sustained level turn)

$$R_{min} = \frac{4K(W/S)}{g\rho_\infty (T/W)\sqrt{1 - 4KC_{D_0}/(T/W)^2}} \tag{4.33}$$

Maximum turn rate (at sustained level turn)

$$\omega_{max} = q\sqrt{\frac{\rho_\infty}{W/S}\left[\frac{T/W}{2K} - \left(\frac{C_{D_0}}{K}\right)^{1/2}\right]} \tag{4.34}$$

Instantaneous turn radius (at *pull up* maneuver)

$$R_{min\,pull-up} = \frac{V_\infty^2}{g(n-1)} \tag{4.35}$$

Instantaneous turn rate (at *pull up* maneuver)

$$\omega_{max\ pull-up} = \frac{g(n-1)}{V_\infty} \tag{4.36}$$

Instantaneous turn radius (at *pull down* maneuver)

$$R_{min\,pull-down} = \frac{V_\infty^2}{g(n+1)} \tag{4.37}$$

Instantaneous turn rate (at *pull down* maneuver)

$$\omega_{max\ pull-down} = \frac{g(n+1)}{V_\infty} \tag{4.38}$$

Then, the ***action time*** for known fuel weight

$$d = \frac{W_{fc}}{SFC\ T} \tag{4.39}$$

And, the ***number of complete turns***

$$xx = \frac{d\omega}{2\pi} \tag{4.40}$$

In the calculations it was assumed that cruise occurs at $40000\,ft$ and the action segment at $15000\,ft$. The atmospheric properties at these altitudes is obtained by calling the International Standard Atmosphere (ISA) subroutine, prepared with well-known equations which were not needed to deal in this study.

4.1.12. Structural Load

It is difficult to provide a complete structural analysis at the conceptual design stage. In spite of this, some structural load parameters are calculated in order to be within feasible structural limits.

The calculations are made within light of references [2], [87].

One of the parameters is maneuver speed defined as the maximum speed at which the control items can fully be deflected without damaging either the airframe or the controls themselves [2].

$$V_{maneuver} = V_{stall} + K_p(V_L - V_{stall}) \tag{4.41}$$

Where,

$$V_{stall} = \sqrt{\frac{2}{\rho}\frac{W}{S}\frac{1}{C_{L_{max}}}} \tag{4.42}$$

$$K_p = 0.15 + \frac{5400}{W + 3300} \tag{4.43}$$

And the factor in equation (4.41), K_p, comes from an empirical relationship and should be between 0.5 and 1.0 [2]. This is controlled as deciding a constraint in the code, also.

V_L is maximum level cruise speed, which was introduced to code as a constant.

Eventually, the maximum available sustained load factor is:

$$n_{max} = \left\{ \frac{\frac{1}{2}\rho_\infty V_\infty^2}{K(W/S)} \left[\left(\frac{T}{W}\right)_{max} - \frac{1}{2}\rho_\infty V_\infty^2 \frac{C_{D_0}}{W/S} \right] \right\}^{1/2} \tag{4.44}$$

An airplane should be designed for a limit load that includes factor of safety, which is usually taken as 1.5. So as introduced in the Weight and Stability part the ultimate load factor of the unmanned supersonic aircraft is:

$$N_Z = 1.5 n_{max} \tag{4.45}$$

4.1.13. Cost Model

Cost are calculated from [89] with DAPCA, the Development and Procurement Cost of Aircraft model.

DAPCA estimates the hours required for research, development, test and evaluation and production by the engineering, tooling, manufacturing, and quality control groups. These are multiplied by the appropriate hourly rates to yield costs. Development support, flight test and manufacturing material costs are directly estimated by DAPCA [89].

For the unmanned supersonic aircraft it was suggested to use aluminium as a material and camouflage paint. The cost is estimated according to this material.

While calculating, the number of flight test aircraft was thought as a constant and equated to 2.

The number of the optimized unmanned supersonic aircraft, $Quantity$, was selected as a constant and equated to 500.

Average hourly rates were taken from [2] for the year 2012. These were adjusted to the year 2019 based on the Consumer Price Indexes calculated from CPI inflation calculator in [90] [91].

The CPI inflation calculator uses the average Consumer Price Index for a given calendar year. This data represents changes in prices of all goods and services purchased for consumption by urban households. For the current year (2019), the latest monthly index value is used [90] [91].

Engineering hourly rates in 2012 [2](and the adjusted values for 2019):

$$R_E = \$115\ (\$130)$$

Tooling hourly rates in 2012 [2](and the adjusted values for 2019):

$$R_T = \$118\ (\$133)$$

Quality control hourly rates in 2012 [2](and the adjusted values for 2019):

$$R_Q = \$108\ (\$122)$$

Manufacturing hourly rates in 2012 [2](and the adjusted values for 2019):

$$R_M = \$98\ (\$111)$$

Acquisition cost of each unmanned supersonic aircraft in $:

$$COST = {H_E R_E + H_T R_T + H_M R_M + H_Q R_Q + Cost_D + Cost_F + Cost_M + Cost_E N_E + Cost_A}\Big/{Quantity} \quad (4.46)$$

4.1.14. Verification of the Aircraft Design Part

The accuracy of the aircraft design part was proved by a supersonic aircraft with similar missions. Therefore, the F-16 was examined because of its known dimensions and performance characteristics, which were written from references [2], [68], [84], [89], [92], [93] and [94].

Since, F-16 is not an unmanned aircraft some small adjustments were made to approximate the results.

These adjustments:

- The unmanned configuration factor of 0.7 was not used for fuselage length.
- Because F-16 is mainly composed of aluminium structure the fudge factors for composite aircraft were not used.

In the main program of the optimization, the values of the design variables were appointed through the algorithm, and constants were taken from an input file. For the verification study, without linking the optimization part to the aircraft design part, all inputs were introduced from an input file according to F-16 as shown in the Table 4.3.

Table 4.3 Design Inputs for the F-16 aircraft

Inputs	**F-16**	**UCAV**
SFC [1/h]	0.64	0.64
SFC_c [1/h]	2.06	2.06
$W_{payload}$ [kg]	1964	1964
Quantity	>3000	3000
b [m]	9.144	9.144
AR	3.0	3.0
$\Lambda_{c/4}$ [deg]	32	32
Λ_{HT} [deg]	40	40
Λ_{VT} [deg]	47.5	47.5
Γ_w [deg]	0	0
Γ_{HT} [deg]	-10	-10
Γ_{VT} [deg]	0	0
BPR	0.87	0.87
λ_{VT}	0.437	0.437
λ_{HT}	0.390	0.390
AR_{VT}	1.294	1.294
AR_{HT}	2.114	2.114

Here, the production quantity for F-16 was found from Table 4.4, [84].

In the optimization code, this table was also used while deciding on the production $Quantity$; and a reasonable number, 500, is used there.

Table 4.4 Examples of Airplane Program Production Runs [84]

Fighters	
Type	**Number produced**
General Dynamics F-111	563
General Dynamics F-16	>3000
Gloster Meteor	3545
Gloster Javelin	435
Grumman F9F2-5	1325
Grumman F9F6-8	1985
Grumman F11F	201
Grumman F14	>900
Lockheed F-94	387
Lockheed F-80	1732
Lockheed T-33	5691
Lockheed F-104	2578
McDonnell F-4	>5000
McDD F-15	>2000
McDD F-18	>1500
SAAB JA37	329
SAAB J35A	604

Although, some necessary inputs are not known exactly for F-16, the assumed input values of UCAV are listed in Table 4.5. Better approximations may be done in future works also.

Table 4.5 Assumed Inputs

Assumed Inputs	**F-16**	**UCAV**
l_{Vtco}	unknown	0.30
l_{Htco}	unknown	0.30
V_{VT}	unknown	0.07
V_{HT}	unknown	0.40
D_{fmax} [m]	varying	1.5
V_{cruise} [km/h]	varying	1460
R_{cr} [km]	varying	750
E_{loiter} [h]	varying	0.5
Airfoil (wing)	NACA 64A204	NACA 64A210
Airfoil (tails)	Biconvex	NACA 64A210

The outputs obtained from the conceptual design program are tabulated in Table 4.6, together with F-16 values.

Table 4.6 UCAV Comparison Table with F-16

Outputs	**F-16**	**UCAV**
S [m^2]	27.87	27.87
Λ_{LE} [deg]	40	40
λ	0.227	0.202
L_f [m]	15.0	14.9
T [kN]	111.2	112.0
L_{eng} [m]	4.67	4.18
D_{eng} [m]	0.91	0.96
$D_{nosewheel}$ [m]	0.46	0.43
$w_{nosewheel}$ [m]	0.14	0.10
$D_{mainwheel}$ [m]	0.65	0.67
$w_{mainwheel}$ [m]	0.20	0.20
W_e [kg]	8910	6328*
W_f** [kg]	3162	3078
W_{TO} [kg]	14036	11370*
n_{max} [g]	9	9
s_{TO} [m]	457	415*
s_L [m]	914	640*
$W/S)_{TO}$ [N/m^2]	4550	4002*
ROC_{max} [m/s]	254	242
Maximum ceiling [m]	15240	15724
$Range_{combat\ mission}$ [km]	1759	2148*
Maximum endurance [h]	2.42	2.35
V_{max} [km/h], (M_{max})	2175, (2.05)	2183, (2.05)
ω_{max} [deg/s]	13	13
Engine Cost [$]	~ 4.0	4.3
COST [$]	14-18 million	14.6 million

*Due to unmanned structure
**Only the internal fuel capacity

Consequently, the resultant top view of the UCAV was sketched and could be checked against the top view of F-16 illustrated in the Figure 4.10 and Figure 4.11.

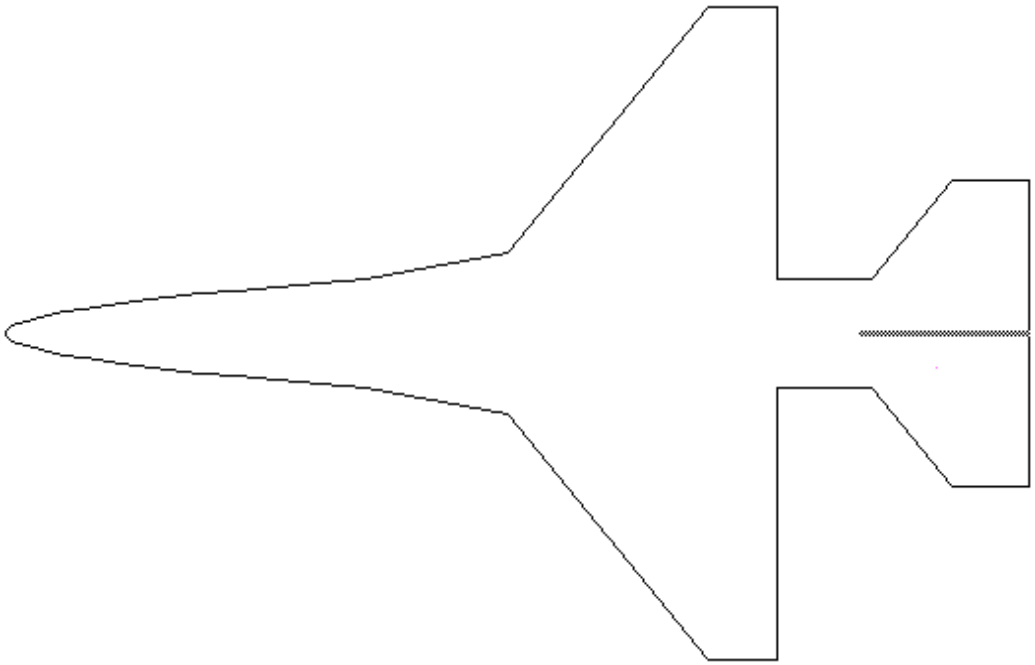


Figure 4.10 Top View of the resulting UCAV design

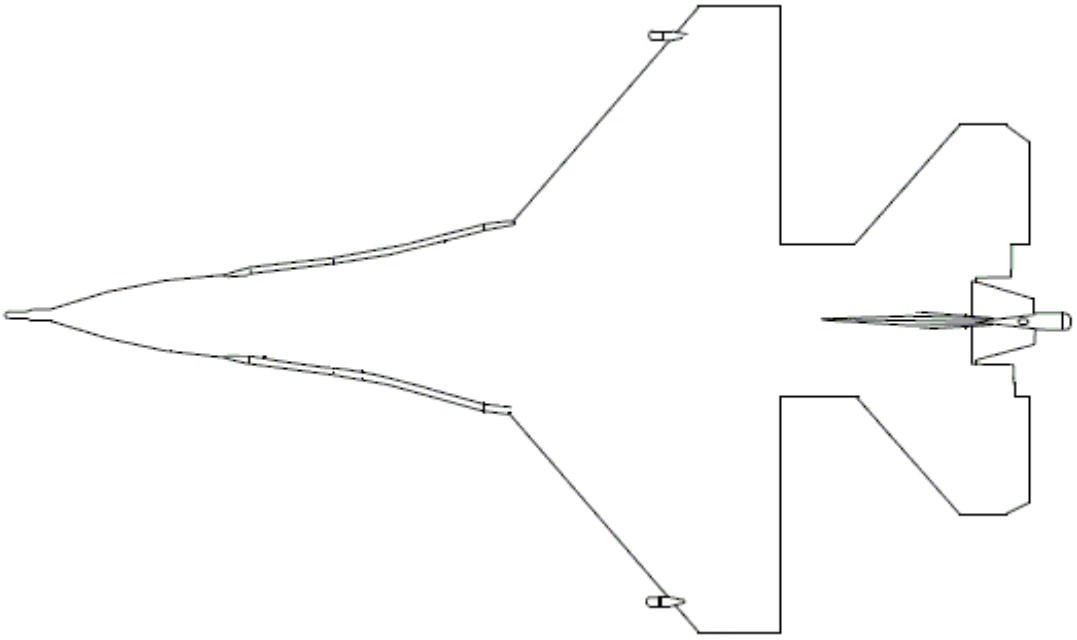


Figure 4.11 Top View of reference F-16 design

Before commenting on the results, it should be remembered that the planned mission parameters for a UCAV will normally be different. For example, the cruise segment range and the loiter time could not be found in the literature as well.

Another difference was on airfoils. Since the data for the airfoils of F-16 ($NACA\ 64A204$ for the wings and biconvex shape for the tails) were not available, thus $NACA\ 64A210$ was used for the UCAV for both the wing and the tails. It was assumed that, this airfoil would approximate the real values.

Table 4.6 shows that the design code calculated results, which are very close to the real values. The main difference is with the empty weight only. Since the systems related to the pilot are not included in the UCAV equations the empty weight was calculated less than that of F-16. Similarly, take off gross weight was found less by the same amount. On the other hand, the performance parameters affected directly by the gross weight were improved accordingly [68].

The resultant shape of the UCAV approximates the real aircraft quite closely resulting into a similar external shape as F-16. As listed in Table 4.6, there are relatively small deviations in dimensions and performance characteristics. Moreover, the resultant cost is also fall into the range. In addition, it should be stated that the UCAV was found statically stable.

In summary, this work shows the design part of the code works well and is ready to be integrated into the optimization part as shown in the section 5.1.

4.2. Trajectory Optimization

For this design case the trajectory optimization for a passenger aircraft is chosen. Trajectory Calculation Module (TCM), a tool for trajectory calculation written in German Aerospace Centre (DLR) [95] is used to optimize the objective function. The inputs of the module are:

- Latitude, longitude and height of the departure airport
- Latitude, longitude and height of the destination airport
- Number of control points
- Way points (used as design variables)
- Lateral displacement of the control points in meters: $1000e3$
- Pressure altitude in meters: 10972.8
- Maximum distance between two waypoints in meters: $100e3$

Basically, the flight route between two waypoints is modelled as an *orthodrome* (shortest path along the surface of the earth's surface) [95].

The output of the module is:

- Flight Time (the objective)

The detailed mathematical model for the trajectory calculation is not given here. Indeed, the aim of this section is to compare the results of the hybrid algorithm with those from the literature (Fmincon, Genetic and Globex [16]) also with an example from one of the fundamental aviation domains, *Air Transportation*. For Fmincon and Genetic Algorithm, the related Matlab functions were used. Globex is applied from [16] as illustrated in Figure 2.10.

The departure and destination cities (end points) were decided with the help of the studies [96] and [97], which forecast the potential urban air mobility markets by the year 2042: Istanbul and Washington.

The atmospheric condition is selected as having the steady state winds throughout the range. The objective is to reduce the flight time against wind. Whereas the end points are fixed, the way points are flexible. Due to that reason, the way points were decided to be the independent variables of the optimization. The range of the way points are in the magnitude of 2 and can differ between -1 and 1. The amount of the way points are also flexible and selected as 10. As a constraint, the resultant curve structured with the optimal way points must be longer than the *great circle* which is actually defined as the shortest way between two points on the Globe.

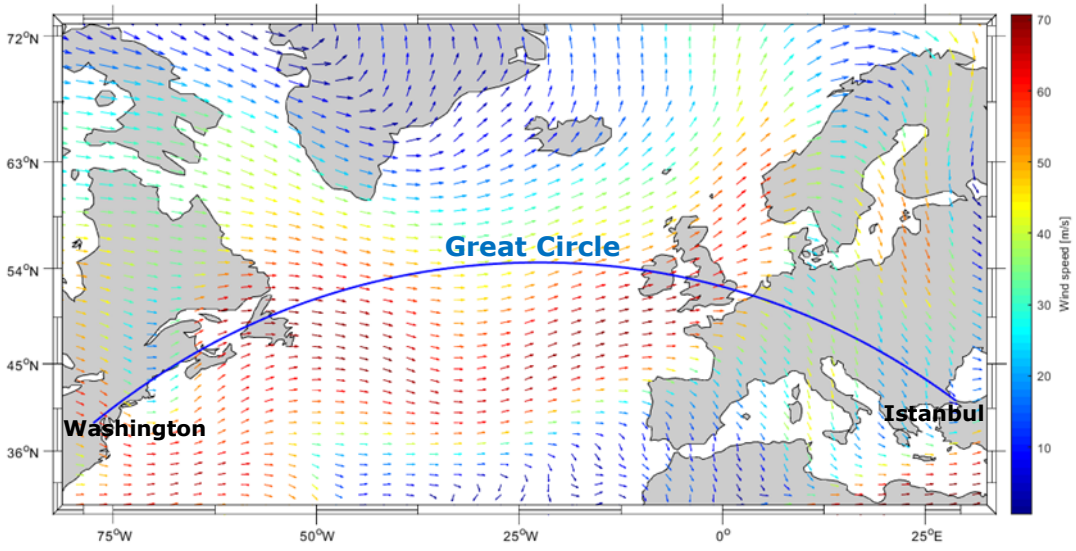


Figure 4.12 Map of the Trajectory Optimization

Since the flight is significantly influenced by the wind and the weather conditions, the flight path becomes longer than the *great circle* and the flight time is increased accordingly [95].

Because of this changing nature, selecting the optimal flight route is a good practice for optimization. Therefore, waypoints are selected as the design variables for the optimization part and the optimal flight time is the objective as a result. The outcomes are given in the Section 5.2

5. Results

5.1.Design Case I (Unmanned Supersonic Aircraft)

An unmanned supersonic aircraft, whose conceptual design model is given in detail in capital 4.1 Aircraft Design, is taken as the design case for a single objective optimization. The objective of the optimization is to reduce the cost. The general mission of an UCAV consists of take-off, climb, cruise, loiter, descent, action, climb, cruise, descent and landing.

The positions of the wings, horizontal and vertical tails, and landing gears with respect to the fuselage are calculated in the Aircraft Design Code prepared. The engine sizing is automatically done in the code which gives its dimensions and its thrust. The calculations are based on a rubber engine with turbofan characteristics which has constant bypass ratio and specific fuel consumption. Landing gears were planned to be tricycle and retractable, and are designed to find the dimensions according to the changing aircraft configurations. The placements of the landing gears are changed for different aircraft configurations with the centre of gravity. A subroutine in the code calculates the total acquisition cost. Proper mathematical models were selected and coded in separate subroutines [68].

Actually, the sized takeoff gross weight is the measure of merit for optimisation and trade studies. However, cost is strongly driven by the weight and it is a final deciding criterion in sale or a design competition. On the other hand, when the alternative technologies, engines, avionics play important role in the design, then it is better to use cost rather than weight on the carpet plot, and same as in multivariable optimizers [2].

Therefore, the objective of this study is to reduce the unit cost of the aircraft. In order to get the feasible aircraft at the end, some constraints should be imposed. The selected constraints for this study and their lower & upper values are listed in Table 5.1. In the code these constraints are normalized with their lower and upper values.

Table 5.1 UCAV Constraints

Constraints	Symbol	Lower Value	Upper Value
Fuselage Length [m]	L_f	5	20
Maximum Structural Load [g]	n_{max}	5	12
Take-off Wing Loading [N/m^2]	$W/S)_{TO}$	2500	7000
Static Margin [%]	Static_margin	-5	+10
Action Time [min]	d	0.7	14
Take-off Distance [m]	s_{TO}	250	700
Landing Distance [m]	s_L	500	1000
Maximum Ceiling [m]	Maximum_ceiling	10000	20000
Range [km]	RANGE	1500	6500
Maximum Endurance [h]	E_{max}	2	6
Min. Sustained Turn Radius [m]	R_{min}	50	250
Max. Sus. Turn Rate [deg/sec]	ω_{max}	8	28
Control Items Deflection Factor	K_P	0.5	1.0

Then, some constants values are read from an input file, which can be changed easily by the user also. How these constants were selected is explained in [68] and they are listed in Table 5.2:

Table 5.2 Constants in UCAV optimizations

Constant	Symbol	Value
Specific Fuel Consumption	SFC	0.64
By-pass Raito	BPR	0.87
Vertical Tail Volume Ratio	V_{VT}	0.40
Horizontal Tail Volume Ratio	V_{HT}	0.07
Vertical Tail Taper Ratio	λ_{VT}	0.3
Horizontal Tail Taper Ratio	λ_{HT}	0.4
Cruise Velocity [km/h]	V_{cruise}	1460
Vertical Tail Aspect Ratio	AR_{VT}	1.4
Horizontal Tail Aspect Ratio	AR_{HT}	3.4
Dash Range [km]	R_d	0.2
Production quantity	Quantity	500

As a single objective optimization problem, the variables are compared with the resultant cost and patterns are selected to decrease this objective function. Accordingly, some parameters were selected as design variables, which are related with the aircraft geometry and mission characteristics. While optimizing, the values are obtained by the computer unsystematicaly with the Latin Hypercube Sampling method in the defined

ranges. These design variables with their boundaries are tabulated in Table 5.3. Wing span, wing sweep angle, vertical tail sweep angle, horizontal and vertical tail volume coefficients, loiter time and payload were selected as the design variables.

Table 5.3 Design variables and their boundaries

	Wing span (m)	Wing sweep angle (deg)	Vertical tail sweep angle (deg)	Horizontal tail volume coefficient	Vertical tail volume coefficient	Loiter time (hour)	Payload (kg)
Upper bound	15	50	55	0.45	0.45	0.75	2500
Lower bound	8	30	35	0.40	0.40	0.10	1500

From the analyses at Sections 3.1 and 3.2, the inequality $(2\,x\,7 \leq n \leq 2^7)$ is used while deciding the number of training points. Thus, the number of training points is selected as $n = 16$ for 7 variables, this lowest value (as well as the power of 2) saves the run time and the memory. Because less number of training points for each iteration means less number of function evaluations in total, as a result. The result is found after 1136 function evaluations. The minimum point is reached at the point where the unit cost is $\$\,19.45$ million.

At first, the Cavus algorithm is compared with the MCMOSA algorithm, because of its proved success to converge the pareto front in studies [68] and [73] for a conceptual air vehicle design. The dependent and the independent variables are listed in Table 5.4 together.

The developed aircraft design code was run with MCMOSA algorithm and it reached the termination criteria after 7786 function evaluations. The minimum point was reached at the point where the unit cost is $\$\,19.62$ million. The detailed values are listed in Table 5.4. From the top and the side views shown at Figure 5.1 and the Table 5.4, it can be seen that the shape as well as the performance is changed slightly and the cost is improved accordingly.

Even though the success of performance parameters slightly differs between these two algorithms; they stay in the given constraints and still meet the requirements for both. On the other hand, the main target was to reduce the number of function evaluations while converging to the optimum, which is much lower than that of the MCMOSA algorithm for the Cavus algorithm, and both are presented in Table 5.8. Regarding that, the single objective convergence, $Cost$, is satisfied slightly better than that of the MCMOSA. Consequently, Cavus algorithm is much faster than MCMOSA at the same level of accuracy.

Table 5.4 Variables and optimization results for UCAV

Algorithm	**Cavus**	**MCMOSA**
Airfoil wing	NACA64A210	NACA64A210
Airfoil horizontal tail	NACA0012	NACA0012
Airfoil vertical tail	NACA0012	NACA0012
Thrust (kN)	64.5	65.1
Cruise altitude (ft)	40000	40000
M_{cruise}	1.37	1.37
C_{D0}	0.0234	0.0232
C_{Lacmax} (1/rad)	2.91	2.90
Wing area (m^2)	16.0	16.2
Aspect ratio	4	4
Taper ratio	0.216	0.215
Horizontal tail sweep (deg)	35.0	35.1
T/W	1.14	1.14
Engine bypass ratio	0.87	0.87
Quantity	500	500
W_0 (kg)	5763	5814
W_e (kg)	3112	3141
W_f (kg)	1151	1173
V_{corner} (km/h)	569	569
Number of turns	3	3
Fuselage length (m)	7.93	7.96
n_{max} (g)	9.0	9.0
$W/S_{takeoff}$	360	359
Static margin %	2.5	2.4
Action Time (min)	1.3	1.1
Take-off Distance (m)	316	315
Landing Distance (m)	610	609
Maximum Ceiling (ft)	57874	57956
Range (km)	2168	2318
$Endurance_{max}$ (hour)	2.4	2.6
Wing span (m)	8.00	8.05
Wing sweep angle (deg)	30.0	30.1
Vertical tail sweep angle (deg)	35.0	44.8
l_{HTco}	0.40	0.40
l_{VTco}	0.40	0.42
$W_{payload}$ (kg)	1500	1500
Loiter (hour)	0.1	0.4
Cost ($ million)	19.45	19.62

The verification of the design is given at the section 4.1.14 and the shapes are presented at Figure 4.10 and Figure 4.11. Likewise, the resultant shapes of Cavus algorithm and MCMOSA are compared in Figure 5.1.

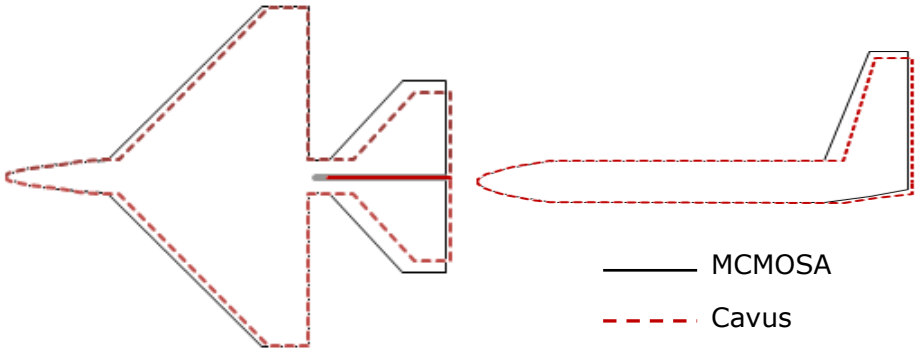


Figure 5.1 Top and side views of the resultant UCAVs

Consequently, within the same range of design variables a lower objective value could be obtained with the new method. It works as a surrogate model of patterns and yields the most promising ones. This saves the number of analyses that is required at each iteration. The computational effort is used for the patterns that have really promising values rather than the unsystematically selected design spaces. Thus, to reach the resultant value less number of function evaluations is needed. With decreasing number of function evaluations, computation time is also saved. This means, if the design problem requires more effort than the optimization algorithm and expands more time, the Cavus algorithm is pretty more successful than this technique. On the other hand, MCMOSA algorithm is a general approach and could be used for a broad application area. Alternatively, the hybrid method may be applied efficiently to the deterministic, observable, convex and non-convex problems that have less number of local minimums. Because Cavus algorithm uses a kind of pattern search technique, very high number of local minimums may mislead the search for problems that are open-end and have stochastic behaviour.

In order to verify the results, the UAVs that have similar mission requirements were searched. The comparison table is shown in Table 5.5.

Because most of them are still under development the exact values could not be listed. Besides that, it should be noted, the mission requirements of these aircraft are not the same. Regarding the resultant design parameters of the Cavus algorithm, the table is sufficient for proving that they fall in the reasonable ranges. Accordingly, the upper and lower bounds of the design variables are selected in respect of the values in Table 5.5, and the resultant cost is crosschecked accordingly.

Indeed, the cost are also difficult to reach and mostly not known. For few aircraft the program cost could be included only.

Table 5.5 Sample UAVs with their Specifications

Aircraft	Specifications											
	Wing span (m)	Length (m)	T (kN)	W_0 (kg)	W_f (kg)	W_e (kg)	W_p (kg)	Max Altitude (m)	Max Endurance (h)	Range (km)	V (Mach)	Cost ($ million)
Cavus	8.00	7.93	64.50	5763	1151	3112	1500	17640	2.4	2168	1.37	19.45
BAE Taranis	10.00	12.43	44.00	7000				11500		3500	0.89	prog: £185
Bayraktar Akıncı	20.00	12.20	1500 hp	5500			1350	12192	24			
Bayraktar TB2	12.00	6.50	100 hp	630	210		55	8239	27	6000	0.20	4.00
Boeing Phantom Ray	15.24	10.97	78.70	16556			>2000	12192			0.80	
Boeing X-45C	14.90	11.90	31.00	16600			2040	12200	7	2222	0.85	
Dassault nEUROn	12.50	9.50	40.00	7000		4900	230	14000			0.80	29.00
EADS Barracuda	7.22	8.25	14.20	3250		2260	300	6096		200	0.85	prog: 40
General Atomics Avenger (Predator C)	20.00	13.00	17.58	8255	3583		2948	15240	18	2897	0.60	12-15
Kratos XQ-58 Valkyrie	8.23	9.14		2722		1134	272	15240		5556	0.94	3.00
Lockheed Martin RQ-170 Sentinel	19.90	9.20	41.26	8242		3544		15250				6.00
MiG Skat	11.50	10.25	49.40	10000			2000	12000		4000	0.80	
Northrop Grumman X-47B	18.93	11.64	71.17	19958			2041	12192	6	3890	0.44	prog: 813
TAI ANKA B	17.50	8.60	170 hp	1700	445		700	9144	24			30.00
Vestel Karayel UCAV	10.50	6.50	97 hp	550			70	6858	20		0.12	

prog: program cost

References: [98], [99], [100], [101], [102], [103], [104], [105], [106], [107], [108], [109], [110], [111], [112], [113], [114], [115], [116], [117], [118], [119], [120], [121], [122], [123], [124], [125], [126] and [127].

The Globex Algorithm [52] is also taken as another reference algorithm for the comparison.

The starting point for the Globex Algorithm is:

Table 5.6 The starting point for Globex Algorithm

Wing span (m)	Wing sweep angle (deg)	Vertical tail sweep angle (deg)	Horizontal tail volume coefficient	Vertical tail volume coefficient	Loiter time (hour)	Payload (kg)
10.00	45.00	50.00	0.41	0.42	0.60	2000

The starting step sizes for the variables are taken as:

Table 5.7 The starting step sizes for Globex Algorithm

dx_1	dx_2	dx_3	dx_4	dx_5	dx_6	dx_7
0.40	1	1	0.0025	0.0025	0.05	50

Taking the constraints as the lower and upper bounds of the variables, the Globex algorithm [52] converges to the unit cost $\$\,19.91$ million after 4321 function evaluations.

On the other hand, when the Genetic Algorithm is used with the same lower and upper bounds as the Cavus algorithm, it converges to the unit cost $\$\,19.45$ million after 10400 function evaluations and 51 generations. Besides the fact that, the Genetic Algorithm could converge to the same unit cost with the Cavus algorithm it has the highest number of function evaluations.

The results of Cavus, Globex and Genetic Algorithm are seen on the Table 5.8. In order to reach the lowest cost value, the optimum point is expected to have the lowest values for the structure related variables (the wing span and the sweep angle) and also for the mission related variables (the loiter time and the payload). The Cavus algorithm has slightly lower value than the Globex algorithm for the cost. Meanwhile, the number of function evaluations is almost 4 times better than the Globex algorithm.

Table 5.8 The Results for the Aircraft Design Case

Algorithm	Wing span (m)	Wing sweep angle (deg)	Vertical tail sweep angle (deg)	Horizontal tail volume coefficient	Vertical tail volume coefficient	Loiter time (hour)	Payload (kg)	Cost ($ million)	Function evaluations
Cavus	8.00	30.00	35.00	0.40	0.40	0.10	1500	19.45	1136
Globex	8.01	33.49	36.13	0.43	0.40	0.40	1508	19.91	4321
MCMOSA	8.05	30.10	44.80	0.40	0.42	0.40	1500	19.62	7786
Genetic	8.00	30.00	35.00	0.40	0.40	0.10	1500	19.45	10400

When the time for evaluating one design point is considered, these reduced function evaluations serve real benefit for the aircraft designer.

5.2.Design Case II (Trajectory Optimization)

The optimization results for the trajectory calculation which is described in chapter 4.2 Trajectory Optimization is illustrated in Figure 5.2.

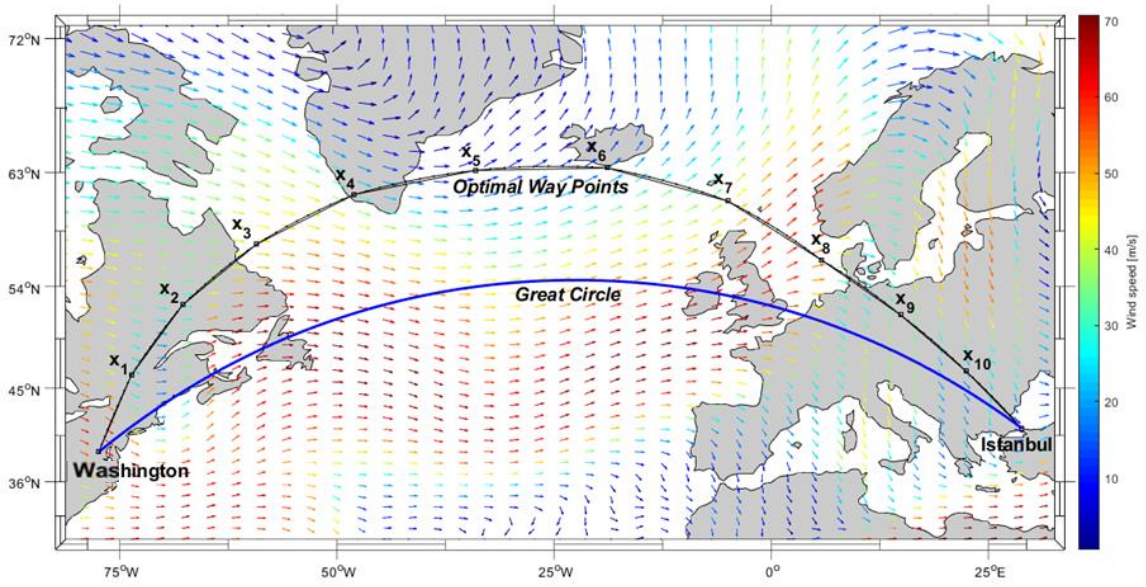


Figure 5.2 Optimal Way Points

The lower and upper limits of the waypoints for all of the algorithms used here:

Table 5.9 Lower and Upper Bounds of the Waypoints

Lower Bounds	Upper Bounds
-1.0	1.0

The result for the Cavus algorithm:

Table 5.10 The result of the Cavus algorithm

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	Optimal Flight Time (s)
0.16	0.34	0.53	0.82	0.98	1.00	1.00	0.88	0.70	0.42	36783

The starting point for the Fmincon algorithm:

Table 5.11 The starting point for the Trajectory Calculation

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

The result for the Fmincon algorithm:

Table 5.12 The result of the Fmincon algorithm for the Trajectory optimization

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	Optimal Flight Time (s)
0.16	0.33	0.53	0.82	0.98	1.00	1.00	0.88	0.70	0.42	36783

For this design case the Globex Algorithm [52] has the starting point:

Table 5.13 The starting point of the Globex Algorithm for the Trajectory optimization

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

The starting step sizes for the Globex Algorithm:

Table 5.14 The starting step sizes of the Globex Algorithm for the Trajectory optimization

dx_1	dx_2	dx_3	dx_4	dx_5	dx_6	dx_7	dx_1	dx_2	dx_3
0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

The result for the Globex algorithm:

Table 5.15 The result of the Globex algorithm

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	Optimal Flight Time (s)
0.16	0.34	0.53	0.82	0.98	1.00	1.00	0.88	0.71	0.42	36783

The result for the Genetic Algorithm:

Table 5.16 The result of the Genetic Algorithm

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	Optimal Flight Time (s)
0.16	0.33	0.52	0.80	0.97	1.00	1.00	0.88	0.71	0.42	36785

The total results are listed and ranked by the numbers of function evaluations as:

Table 5.17 Comparison of the results for the Trajectory Optimization

Algorithm	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	Optimal Flight Time (s)	Function evaluations
Fmincon	0.16	0.33	0.53	0.82	0.98	1.00	1.00	0.88	0.70	0.42	36783	314
Cavus	0.16	0.34	0.53	0.82	0.98	1.00	1.00	0.88	0.70	0.42	36783	416
Globex	0.16	0.34	0.53	0.82	0.98	1.00	1.00	0.88	0.71	0.42	36783	19309
Genetic	0.16	0.33	0.52	0.80	0.97	1.00	1.00	0.88	0.71	0.42	36785	21400

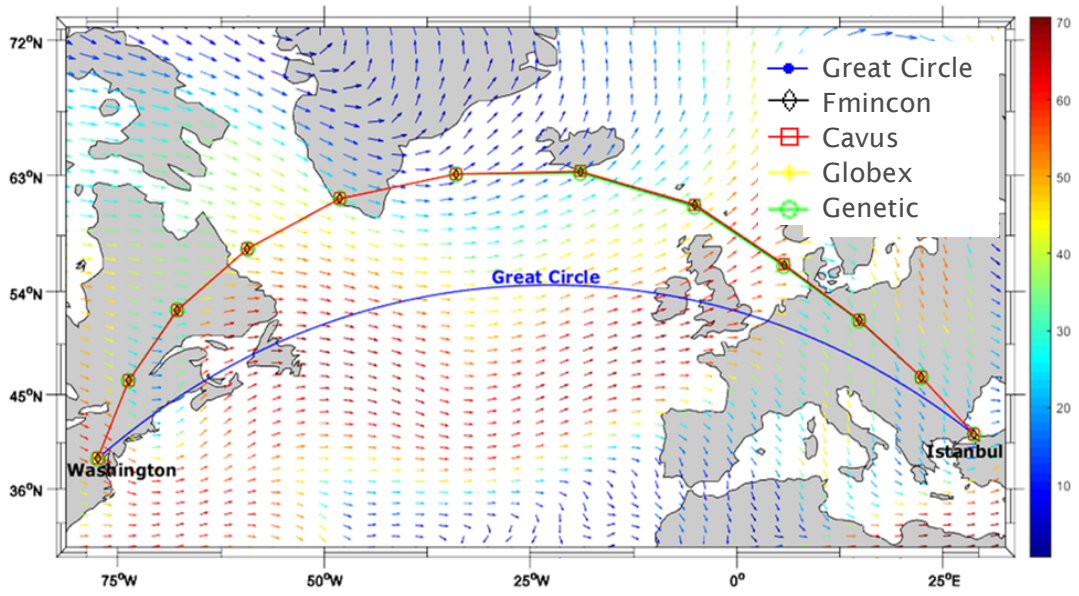


Figure 5.3 Resultant Optimal Way Points

To clarify, when the function evaluations are concerned the Cavus algorithm performs better than the Genetic Algorithm and the Globex algorithm. Especially, if the design problem requires more effort than the optimization algorithm and expands more time, the Cavus algorithm is pretty more successful than these techniques. On the other hand another method which is known as Fmincon function in MATLAB is also used here. This method is applied especially to the nonlinear and not much complex problems and as being known as a problem specific function, it shows having an advantage over the Cavus algorithm for that trajectory optimization. In fact, the calculation time increases rationally with the number of function evaluations; more number of function evaluations means more calculation time spent.

6. Conclusion

The new hybrid technique is studied here to have a specific optimization algorithm for the problems that have a complex structure and high number of design variables whereas having low number of local minimums. In fact, the starting point of this algorithm was actually from the need of aerospace domain. The success of the evolutionary algorithms that are mostly used in aerospace would be considered as becoming cumbersome when we consider the developments in other domains recently. Nevertheless, aerospace design cases inherently have their advantages parallel to the recent developments. First, there is a high number of related data which can be collected from the real life or produced with using the domain equations. Second, the aerospace problems are closed loop processes; simply the chemical energy (the fuel) is converted to the potential and kinetic energy (the range, etc.), nothing is disappeared or changed extraordinarily, which means a consistent and a sustainable structure. Third, aerospace problems are really challenging cases when high fidelity results are required, on the contrary to that the intermediate stages can be estimated with low fidelity methods plus probabilistic approximations due to the stated two facts above.

In this study, depending on the mentioned three properties a novel algorithm is developed. The original contributions of the thesis are:

1. Instead of using interpolations between the design points, gradients derived from these points are used at each stage, and presented as patterns
2. These patterns are classified in each other, and also on top they are clustered as tried and untried patterns
3. Using the probability of success and neighbouring, the scores are assigned and systematical improvements are gained
4. Since each design space component is presented with a pattern, at the end there exists no design space that is left as untraced.
5. To decrease the memory usage and improve the computational performance, a case specific dimensional reduction is applied. The number of training points is reduced accordingly, especially for the problems that have high number of variables.

The main target of the novel algorithm is to reduce the function evaluations while converging to the global optimum with higher accuracy.

The hybrid algorithm is composed of pattern generation, harmonic distribution with dimension reduction model, PNN and selection module. At an intermediate step, one of the ANN techniques (PNN) is integrated to train the algorithm and find the probability of the success of each neighbouring patterns. Besides, the burden of the PNN with increasing number of design variables is alleviated with the dimension reduction model. Thus, the number of trial points is reduced with less compromise, while the algorithm becomes faster and less power consuming.

The trend analysis was done to present the effect of the number of training points with changing variable numbers. The results show that when the number of training points is kept constant, with increasing variable numbers the success to find the

promising patterns is reduced as expected. Besides that, when the number of training points are increased proportionally, with the increasing variable numbers the success is improved until a level after which the system is defined as overtrained.

The algorithm is tested on the Rosenbrock function to demonstrate the efficiency of the algorithm for 2, 7, 10 and 14 variables. The results are compared with those from the literature, Globex and Genetic Algorithm, and presented in Table 3.26, Figure 3.24 and Figure 3.25. In summary, with the lower function evaluations the accuracies of the results are better for the Cavus algorithm. Although, the increasing inclination of the Function Evaluations vs. Variable numbers line should be examined and could be improved as a future work.

The algorithm is developed to optimize single objective for two design cases in this study. However, with the same modules of the code the number of objectives can be increased without much effort. Only the objective function should be adjusted for multiobjective optimization case. As such, normalized change on the objectives may be handled with digits and integrated to a string like objective patterns. Then, they may be grouped in 2^{x-1} objective patterns, where x is the number of objectives. Then, they may be assigned as successful or unsuccessful patterns. Moreover, proportional with the normalized values, these objective patterns may be sequenced and fed by penalty coefficients accordingly. Similarly, they may be classified as successful, less successful or unsuccessful patterns, and then these complex relations may also be examined. Multiobjective optimization case is to be an interesting future work.

The Cavus algorithm is applied on a supersonic aircraft discrete mission to minimize the unit cost. The values of the constraints and design variables are slightly differ from the other algorithms, but the target for the objective function and function evaluations are reached for this case also, in other words better results are obtained.

Another aerospace issue, the trajectory optimization is also studied. Again the hybrid algorithm performs better than Globex and Genetic Algorithm. However, the problem specific algorithm, Fmincon, that is known and used especially this kind of less complex and nonlinear problems, performed slightly better than the Cavus algorithm when the number of function evaluations is concerned, but the accuracy is similar.

Indeed, it is proved that the used hybrid method increases the efficiency of the optimization and improves the design task; and it is seen to be competitive to the other well-known optimization techniques.

In addition, the number of training points is found out as the determining parameter on the efficiency of the Cavus algorithm. While increasing the values of this parameter the convergence is improved but the cycle time and the memory usage are also increased.

It is shown that the Probabilistic Neural Networks with the combination of rule based agent systems the optimization of a design is possible and has advantages on different type of problems in aerospace area. With this approach the dimension reduction is also possible, which saves the memory and improves the efficiency. Although, searching the whole design area with promising patterns expends the memory, it also helps to diminish the number of function evaluations used for calculating the poor design points, which are experienced and defined by the previous trials. Finally, the hybrid algorithm saves the computational effort and time.

Bibliography

[1] in *The Holy Qur'an*, Translator: Elmalılı Muhammed Hamdi Yazır, Press: T.C. Diyanet İşleri Reisliği, Translation Title: Hak Dini Kur'an Dili, Translation Publication Year: 1936, p. 2:255.

[2] D. P. Raymer, Aircraft Design: A Conceptual Approach, 5th ed., Virginia: AIAA Education Series, American Institute of Aeronautics and Astronautics Inc., 2012.

[3] J. S. Saggu, "Distributed Artificial Intelligence Applied to Design of Aircraft Fuselage and Wings," in *Proceedings of the 19th Congress of the International Council of the Aeronautical Sciences*, Anaheim, California, U.S.A., 18-23 September 1994.

[4] T. Pfeiffer, B. Nagel, D. Böhnke, A. Rizzi and M. Voskuijl, "Implementation of a Heterogeneous, Variable-Fidelity Framework for Flight Mechanics Analysis in Preliminary Aircraft Design," in *Proceedings of the 60 Deutscher Luft- und Raumfahrtkongress*, Bremen, Germany, 27-29 September 2011.

[5] N. V. Nguyen, S. M. Choi, W. S. Kim, K. S. Jeon, J. W. Lee and Y. H. Byun, "Multidisciplinary Unmanned Combat Air Vehicle-UCAV Design Optimization Using Variable Complexity Modelling," in *Proceedings of the 9th AIAA Aviation Technology, Integration and Operations Conference (ATIO)*, South Carolina, 21-23 September 2009.

[6] J. Sobieski-Sobiezczanski and R. T. Haftka, "Multidisciplinary Aerospace Design Optimization: survey of Recent Developments," in *Proceedings of the 34th Aerospace Science Meeting*, Reno, NV, January 1996.

[7] K. Amadori, C. Jouannet and P. Krus, "Aircraft Conceptual Design Optimization," in *Proceedings of the 26th International Congress of the Aeronautical Sciences*, Stockholm, Sweden, 2008.

[8] T. Zill, Model Hierarchy Exploitation for Efficient Multidisciplinary Design Optimization in a Distributed Aircraft Design Environment, Hamburg: Ph.D. Thesis, Hamburg University of Technology, 2013.

[9] I. Staack, R. Munjulury, T. Melin, A. Abdalla and P. Krus, "Conceptual Aircraft Design Model Management Demonstrated on a 4th Generation Fighter," in *Proceedings of the 29th Congress of the International Council of the Aeronautical Sciences*, St. Petersburg, Russia, 7-12 September 2014.

[10] A. Gelsey, M. Schwabacher and D. Smith, "Using Modeling Knowledge to Guide Design Space Search," *Artificial Intelligence,* vol. 101, pp. 35-62, 1998.

[11] I. P. Sobieski and I. M. Kroo, "Collaborative Optimization Using Response Surface Estimation," *AIAA Journal,* vol. 38, no. 10, pp. 1931-1938, 2000.

[12] D. Tejtel, D. N. Mavris and M. Hale, "Conceptual Aircraft Design Environment: Case study evaluation of Computing Architecture Technologies," in *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, MO, U.S.A., 1998.

[13] L. Z. Bashir and R. S. M. Hasan, "Solving Banana (Rosenbrock) Function Based on Fitness Function," *World Scientific News,* vol. 6, pp. 41-56, 2015.

[14] D. F. Specht, "Probabilistic Neural Networks for Classification, Mapping, or Associative Memory," in *Proceedings of IEEE International Conference on Neural Networks*, San Diego, CA, U.S.A., 1988.

[15] G. N. Vanderplaatz, "Approximation Concepts for Numerical Airfoil Optimization," National Aeronautics and Space Administration, Ames Research Center, Moffett Field, California, 1979.

[16] H. G. Jacob, "An Engineering Optimization Method with Application to STOL-Aircraft Approach and Landing Trajectories," National Aeronautics and Space Administration, Washington, D.C., 1972.

[17] V. Torczon, "On the Convergence of Pattern Search Algorithms," *SIAM Journal on Optimization,* vol. 7, no. 1, pp. 1-25, 1997.

[18] A. A. Hopgood, "The State of Artificial Intelligence," in *Advances in Computers*, vol. 65, M. V. Zelkowitz, Ed., Amsterdam, Elsevier Academic Press, 2005, pp. 1-75.

[19] G. La Rocca and M. J. L. van Tooren, "Knowledge-Based Engineering Approach to Support Aircraft Multidisciplinary Design and Optimization," *Journal of Aircraft,* vol. 46, no. 6, pp. 1875-1885, 2009.

[20] G. Gursoy and I. Yavrucuk, "Direct Adaptive Limit and Control Margin Estimation with Concurrent Learning," *Journal of Guidance, Control, and Dynamics,* vol. 39, no. 6, p. 1356–1373, 2016.

[21] M. A. A. Oroumieh, S. M. B. Malaek, M. Ashrafizaadeh. and S. M. Taheri, "Aircraft Design Cycle Time Reduction using Artificial Intelligence," *Aerospace Science and Technology,* vol. 26, pp. 244-258, 2013.

[22] N. Ernest, D. Carroll, C. Schumacher, M. Clark, K. Cohen and G. Lee, "Genetic Fuzzy based Artificial Intelligence for Unmanned Combat Aerial Vehicle Control in Simulated Air Combat Missions," *Journal of Defense Management,* vol. 6, no. 1, pp. 1-7, 2016.

[23] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, Adaptive Computation and Machine Learning Series, Cambridge, MA: The MIT Press, 2016.

[24] D. L. Poole and A. K. Mackworth, Artificial Intelligence: Foundations of Computational Agents, First ed., New York: Cambridge University Press, 2010.

[25] A. L. Chandra, "McCulloch-Pitts Neuron _ Mankind's First Mathematical Model of a Biological Neuron," 24 July 2018. [Online]. Available: https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1. [Accessed 7 November 2021].

[26] A. M. Turing, "Computing Machinery and Intelligence," *Mind,* vol. 59, no. 236, pp. 433-460, 1950.

[27] M. Negnevitsky, Artificial Intelligence: A Guide to Intelligent Systems, 2nd ed., England: Pearson Education, 2005.

[28] J. McCarthy, "Programs with Common Sense," in *Proceedings of the Symposium by H. M. Stationary Office (Paper presented at the Symposium on the Mechanization of Thought Processes)*, National Physical Laboratory, Teddington, England, 1958.

[29] J. McCarthy, "Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I," *Communications of the ACM,* vol. 3, no. 4, pp. 184-195, 1960.

[30] D. Bobrow, Natural Language Input for a Computer Problem Solving System, Cambridge: Ph.D. Thesis, Department of Mathematics, Massachusetts Institute of Technology, 1964.

[31] Tutorials Point (I) Pvt. Ltd., "Artificial Intelligence," 2015. [Online]. Available: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_tutorial.pdf. [Accessed 7 August 2019].

[32] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, Global Edition, 3. ed., Saffron House, London: Pearson Education, 2016.

[33] P. D. Gallaher, R. A. Hunt and R. C. Williges, "A Regression Approach to Generate Aircraft Predictor Information," *Human Factors,* vol. 19, no. 6, pp. 549-555, 1977.

[34] I. Kroo and M. Takai, "A Quasi-Procedural, Knowledge-based System for Aircraft Design," in *Proceedings of AIAA/AHS/ASEE Aircraft Design, Systems and Operations Meeting*, Atlanta, Georgia, 7-9 September 1988.

[35] A. Takasu, Y. Itoh, S. Futatsugi and S. Ohsuga, "Intelligent Wing Design Support System," in *Proceedings of Scandinavian Conference on Artificial Intelligence*, Tampere, Finland, 13-15 June 1989.

[36] S. Ohsuga, "A New Method of Model Description: Use of Knowledge Base and Inference," in *the Proceedings of Working Conference on CAD system framework*, North-Holland, 1983.

[37] S. Ohsuga, "Predicate Logic Involving Data Structure as a Knowledge Representation Language," in *the Proceedings of the 8th International Joint Conference on Artificial Intelligence*, Karlsruhe, Germany, 1983.

[38] H. Yamauchi and S. Ohsuga, "KAUS as a Tool for Model Building and Evaluation," in *the Proceedings of the 5th International Workshop on Expert Systems and Their Applications*, Avigon, France, 1985.

[39] A. R. Dovi and G. A. Wrenn, "Aircraft Design for Mission Performance Using Nonlinear Multiobjective Optimization Methods," NASA Langley Research Center, Hampton, VA, U.S.A., October 1990.

[40] G. A. Wrenn, "An Indirect Method for Numerical Optimization Using the Kreisselmeier-Steinhauser Function," NASA Langley Research Center, Hampton, VA, U.S.A., March 1989.

[41] J. Sobieski-Sobieszczanski, A. R. Dovi and G. A. Wrenn, "A New Algorithm for General Multiobjective Optimization," National Aeronautics and Space Administration, Washington, DC, March 1988.

[42] A. Faghri and J. Hua, "Evaluation of Artificial Neural Network Applications in Transportation Engineering," National Research Council, Washington, DC, 1992.

[43] S. S. Rao, Engineering Optimization: Theory and practice, 4. ed., New Jersey: John Wiley & Sons Inc., 2009.

[44] P. Venkataraman, Applied Optimization with Matlab Programming, 2. ed., New Jersey: John Wiley & Sons Inc., 2009.

[45] J. C. Spall, Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control, New Jersey: John Wiley & Sons Inc., 2003.

[46] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, International edition, 3 ed., New Jersey: Pearson Education, Inc., 2010.

[47] L. E. Sucar, Probabilistic Graphical Models: Principles and Applications, In: Advances in Computer Vision and Pattern Recognition, London: Springer Verlag, 2015.

[48] R. M. Lewis and V. Torczon, "Pattern Search Methods for Linearly Constrained Minimization," Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, January 1998.

[49] D. G. Krige, A Statistical Approach to Some Mine Valuation and Allied Problems on the Witwatersrand, Witwatersrand: M.Sc. Thesis, University of the Witwatersrand, 1951.

[50] J. Sacks, W. J. Welch, T. J. Mitchell and H. P. Wynn, "Design and Analysis of Computer Experiments," *Statistical Science,* vol. 4, pp. 409-435, 1989.

[51] G. Matheron, "Principles of Geostatistics," *Economic Geology,* vol. 58, pp. 1246-1266, 1963.

[52] H. G. Jacob, Rechnergestützte Optimierung statischer und dynamischer Systeme : Beispiele mit FORTRAN-Programmen, Berlin: Springer Verlag, 1982.

[53] D. R. Tveter, The Pattern Recognition Basis of Artificial Intelligence, Los Alamitos, California: IEEE Computer Society Press, 1998.

[54] C. M. Bishop, Pattern Recognition and Machine Learning, New York: Springer, 2006.

[55] K. P. Murphy, Machine Learning: A Probabilistic Perspective, Cambridge: The MIT Press, 2012.

[56] D. F. Specht, "Probabilistic Neural Networks," *Neural Networks,* vol. 3, pp. 109-118, 1990.

[57] E. Parzen, "On Estimation of a Probability Density Function and Mode," *Annals of Mathematical Statistics,* vol. 33, pp. 1065-1076, 1962.

[58] A. Lotfi and A. Benyettou, "Using Probabilistic Neural Networks for Handwritten Digit Recognition," *Journal of Artificial Intelligence,* vol. 4, no. 4, pp. 288-294, 2011.

[59] D. F. Specht, "Generation of Polynomial Discriminant Functions for Pattern Recognition," *IEEE Transactions on Electronic Computers,* Vols. EC-16, no. 3, pp. 308-319, 1967.

[60] T. M. Cover and P. E. Hart, "Nearest Neighbour Pattern Classification," *IEEE Transactions on Information Theory,* Vols. IT-13, no. 1, pp. 21-27, January 1967.

[61] R. Hecht-Nielsen, "Nearest Matched Filter Classification of Spatiotemporal Patterns," *Applied Optics,* vol. 26, no. 10, pp. 1892-1899, May 1987.

[62] D. F. Specht, "Generation of Polynomial Discriminant Functions for Pattern Recognition," Stanford Electronics Labs., Stanford, CA, May 1966.

[63] S. S. Sawant and P. S. Topannavar, "Introduction to Probabilistic Neural Network - Used for Image Classifications," *International Journal of Advanced Research in Computer Science and Software Engineering,* vol. 5, no. 4, pp. 279-283, 2015.

[64] D. P. Solomatine, "Genetic and Other Global Optimization Algorithms - Comparison and Use in Calibration Problems," in *the Proceedings of the 3rd International Conference on Hydroinformatics*, Copenhagen, Denmark, 1998.

[65] N. Cavus, "Multidisciplinary Design Optimization using Artificial Intelligence for Aircraft Control," in *8th Ankara International Aerospace Conference*, Ankara, Turkey, 10-12 September 2015.

[66] N. Cavus, "Artificial Intelligence in Aircraft Conceptual Design Optimization," *International Journal of Sustainable Aviation,* vol. 2, no. 2, pp. 119-127, 2016.

[67] N. Cavus, "Aircraft Optimization at the Early Stages of Design with a Hybrid Technique," in *6th Council of European Aerospace Societies (CEAS) Air & Space Conference Proceedings*, Bucharest, Romania, 16-20 October 2017.

[68] N. Cavus, Multidisciplinary and Multiobjective Design Optimization of an Unmanned Combat Aerial Vehicle, Ankara: M.Sc. Thesis, Aerospace Engineering Department, Middle East Technical University, 2009.

[69] O. Tekinalp and N. Cavus, "Multiobjective Conceptual Design of an Unmanned Combat Air Vehicle," in *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Indianapolis, Indiana, 17-19 September 2012.

[70] N. Cavus, "Multidisciplinary Design Optimization with using Artificial Intelligence for Aircraft Control, 1st International Symposium on Sustainable Aviation," Istanbul, Turkey, 31 May - 3 June 2015.

[71] N. Cavus, "Aircraft Design Optimization with Artificial Intelligence," in *54th AIAA Aerospace Sciences Meeting, AIAA SciTech*, San Diego, California, USA, 3-8 January 2016.

[72] O. Tekinalp and G. Karslı, "A new Multiobjective Simulated Annealing Algorithm," *Journal of Global Optimization,* vol. 39, pp. 49-77, 2007.

[73] S. Özdemir, Multi Objective Conceptual Design Optimization of an Agricultural Aerial Robot (AAR), Ankara: M.Sc. Thesis, Aerospace Engineering Department, Middle East Technical University, 2005.

[74] J. C. Helton and D. E. Burmaster, "Treatment of Aleatory and Epistemic Uncertainty in Performance Assessments for Complex Systems," *Reliability Engineering & System Safety,* vol. 54, no. 2, pp. 91-94, 1996.

[75] F. O. Hoffman and J. S. Hammonds, "Propagation of Uncertainty in Risk Assessments: The Need to Distinguish Between Uncertainty Due to Lack of Knowledge and Uncertainty Due to Variability," *Risk Analysis,* vol. 14, no. 5, pp. 707-712, 1994.

[76] W. Yao, X. Chen, Q. Ouyang and M. van Tooren, "A Reliability-based Multidisciplinary Design Optimization Procedure Based on Combined Probability and Evidence Theory," *Structural and Multidisciplinary Optimization,* vol. 48, no. 2, p. 339–354, 2013.

[77] V. K. Murthy, "Estimation of a Probability Density," *Annals of Mathematical Statistics,* vol. 36, pp. 1027-1031, 1965.

[78] V. K. Murthy, "Nonparametric Estimation of Multivariate Densities with Applications," in *Multivariate Analysis*, P. R. Krishnaiah, Ed., New York, Academic Press, 1966, pp. 43-58.

[79] T. Cacoullos, "Estimation of a Multivariate Density," in *Annals of the Institute of Statistical Mathematics*, vol. 18, Tokyo, Springer, December 1966, pp. 179-189.

[80] D. E. Rumelhart, J. L. McClelland and the PDP Research Group, Parallel Distributed Processing, Volume I Explorations in the Microstructure of Cognition: Foundations, Cambridge, MA: The MIT Press, 1986.

[81] M. D. McKay, R. J. Beckman and W. J. Conover, "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics,* vol. 21, no. 2, pp. 239-245, 1979.

[82] R. L. Iman, J. C. Helton and J. E. Campbell, "An Approach to Sensitivity Analysis of Computer Models: Part I - Introduction, Input Variable Selection and Preliminary Variable Assessment," *Journal of Quality Technology,* vol. 13, no. 3, pp. 174-183, 1981.

[83] D. Böhnke, A Multi-Fidelity Workflow to Drive Physics-Based Conceptual Design Methods, Hamburg: Ph.D. Thesis, Hamburg University of Technology, 2015.

[84] J. Roskam, Airplane Design Part I-VII, Lawrence, Kansas: DAR Corporation, 2000.

[85] A. Abdullah, M. N. S. M. Jafri and M. F. Zulkafli, "Numerical Study of Military Airfoils Desgin for Compressible Flow," *ARPN Journal of Engineering and Applied Sciences,* vol. 12, no. 24, pp. 7129-7133, December 2017.

[86] E. Torenbeek, Synthesis of Subsonic Airplane Design, Holland: Kluwer Academic Publishers, 1982.

[87] J. D. Anderson, Aircraft Performance and Design, New York: McGraw-Hill, 1999.

[88] B. L. Stevens and F. L. Lewis, Aircraft Control and Simulation, 2 ed., New Jersey: John Wiley & Sons, Inc., 2003.

[89] D. P. Raymer, Aircraft Design: A Conceptual Approach, 3. ed., Virginia: American Institute of Aeronautics and Astronautics Education Series, 1999.

[90] Bureau of Labor Statistics, "CPI Inflation Calculator," U.S. Department of Labor, [Online]. Available: https://data.bls.gov/cgi-bin/cpicalc.pl. [Accessed 7 August 2019].

[91] Bureau of Labor Statistics, "Consumer Price Index," U.S. Department of Labour, [Online]. Available: https://www.bls.gov/CPI/#data. [Accessed 7 August 2019].

[92] Aeroflight, "Lockheed Martin F-16 Fighting Falcon," 27 June 2016. [Online]. Available: http://www.aeroflight.co.uk/aircraft/types/type-details/lockheed-martin-f-16-fighting-falcon.htm. [Accessed 26 December 2019].

[93] E. J. Saltzman and J. W. Hicks, "In-Flight Lift-Drag Characteristics for a Forward-Swept Wing Aircraft (and Comparisions with Contemporary Aircraft)," National Aeronautics and Space Administration, California, 1994.

[94] M. C. Fox and D. K. Forrest, "Supersonic Aerodynamic Characteristics of an Advanced F-16 Derivative Aircraft Configuration," NASA Langley Research Center, Hampton, VA, June 1993.

[95] B. Lührs, F. Linke und V. Gollnick, „Erweiterung eines Trajektorienrechners zur Nutzung Meteorologischer Daten für die Optimierung von Flugzeugtrajektorien," in *63. Deutscher Luft- und Raumfahrtkongress (DLRK)*, Augsburg, Deutschland, 16-18 September 2014.

[96] K. Becker, I. Terekhov, M. Niklaß and V. Gollnick, "A Global Gravity Model for Air Passenger Demand between City Pairs and Future Interurban Air Mobility Markets Identification," in *2018 Aviation Technology, Integration, and Operations Conference, AIAA Aviation Forum*, Atlanta, Georgia, 25-29 June 2018.

[97] I. Terekhov, Forecasting Air Passenger Demand between Settlements Worldwide Based on Socio-Economic Scenarios, Hamburg: Ph.D. Thesis, Hamburg University of Technology, 2017.

[98] G. Bakır, "İnsansız Hava Araçlarının Savunma Sanayi Harcamasında Yeri ve Önemi," *Eurasian Journal of Researches in Social and Economics (EJRSE),* vol. 6, no. 2, pp. 127-134, 2019.

[99] J. Gertler, "U.S. Unmanned Aerial Systems," Congressional Research Service, Washington, DC, 2012.

[100] B. Yenne, Drone Strike!: UCAVs and Aerial Warfare in the 21st Century, 1st ed., Forest Lake, MN: Specialty Press, 2017, pp. 90, 185.

[101] "BAE Systems Taranis," Avia.Pro, 28 October 2016. [Online]. Available: http://avia-pro.net/blog/bae-systems-taranis-tehnicheskie-harakteristiki-foto. [Accessed 4 January 2020].

[102] A. Parsch, "Boeing X-45 / X-46," Designation-Systems.Net, 2 September 2007. [Online]. Available: http://www.designation-systems.net/dusrm/app4/x-45.html. [Accessed 4 January 2020].

[103] "Predator C Avenger," General Atomics Aeronautical Systems, Inc., 2015. [Online]. Available: http://www.ga-asi.com/Websites/gaasi/images/products/aircraft_systems/pdf/Predator_C021915.pdf. [Accessed 4 January 2020].

[104] "Tactical UAVs," Kratos Defense, 2019. [Online]. Available: http://www.kratosdefense.com/systems-and-platforms/unmanned-systems/aerial/tactical-uavs#XQ58A. [Accessed 4 January 2020].

[105] "Karayel Tactical UAV," VESTEL Defence Industry, 2017. [Online]. Available: http://www.vestelsavunma.com/views/web/vestelsavunma/downloads/tr/urunler/KARAYELTacticalUAVeng-tr.pdf. [Accessed 4 January 2020].

[106] C. Haddox, "Boeing 'Phantom Eye' Hydrogen Powered Vehicle Takes Shape," Boeing, 8 March 2010. [Online]. Available: https://boeing.mediaroom.com/2010-03-08-Boeing-Phantom-Eye-Hydrogen-Powered-Vehicle-Takes-Shape. [Accessed 4 January 2020].

[107] C. Ritsick, "Top 35 Most Expensive Military Drones," Military Machine, 26 July 2019. [Online]. Available: https://militarymachine.com/top-35-most-expensive-military-drones/. [Accessed 4 January 2020].

[108] M. Cassese, "The RQ-170 Sentinel Spy Drone: The Beast of Kandahar," Postmedia Network Inc., 7 December 2011. [Online]. Available: https://nationalpost.com/news/rq-170-sentinel-spy-drone-the-beast-of-kandahar. [Accessed 4 January 2020].

[109] M. Grimson and M. Corcoran, "Taranis drone: Britain's $336m supersonic unmanned aircraft launched over Woomera," ABC News, 7 February 2014. [Online]. Available: https://www.abc.net.au/news/2014-02-06/taranis-drone-uk-mod-bae-systems-woomera-south-australia/5242636. [Accessed 4 January 2020].

[110] "Barracuda Demonstrator Unmanned Air Vehicle Developed by EADS Military Air Systems," Army Technology, [Online]. Available: https://www.army-technology.com/projects/barracuda-demonstrator-uav/. [Accessed 4 January 2020].

[111] "Taranis," BAE Systems, [Online]. Available: https://www.baesystems.com/en/product/taranis. [Accessed 4 January 2020].

[112] "Bayraktar Akıncı," BAYKAR Savunma, [Online]. Available: https://www.baykarsavunma.com/iha-14.html. [Accessed 4 January 2020].

[113] "Bayraktar TB2," BAYKAR Savunma, [Online]. Available: https://www.baykarsavunma.com/iha-15.html. [Accessed 4 January 2020].

[114] "Anka Block-B MALE UAV System Starts Serving Turkish Naval Forces Command," Defence Turkey, 2018. [Online]. Available: https://www.defenceturkey.com/tr/icerik/anka-block-b-male-uav-system-starts-serving-turkish-naval-forces-command-3035. [Accessed 4 January 2020].

[115] B. Stevenson, "Upgraded Anka carries out maiden flight," FlightGlobal, 2 February 2015. [Online]. Available: https://www.flightglobal.com/civil-uavs/upgraded-anka-carries-out-maiden-flight/115789.article. [Accessed 4 January 2020].

[116] J. Pike, "Phantom Ray," GlobalSecurity.org, 16 November 2017. [Online]. Available: https://www.globalsecurity.org/military/systems/aircraft/phantom-ray.htm. [Accessed 4 January 2020].

[117] J. Pike, "MiG Skat UAV," GlobalSecurity.org, 17 May 2019. [Online]. Available: https://www.globalsecurity.org/military/world/russia/mig-skat.htm. [Accessed 4 January 2020].

[118] "Dassault nEUROn," MilitaryFactory.com, 24 January 2019. [Online]. Available: https://www.militaryfactory.com/aircraft/detail.asp?aircraft_id=987. [Accessed 4 January 2020].

[119] D. Alex and J. R. Potts, "Northrop Grumman X-47B," MilitaryFactory.com, 24 January 2019. [Online]. Available: https://www.militaryfactory.com/aircraft/detail.asp?aircraft_id=1015. [Accessed 4 January 2020].

[120] "Bayraktar TB2," MilitaryFactory.com, 13 May 2019. [Online]. Available: https://www.militaryfactory.com/aircraft/detail.asp?aircraft_id=1679. [Accessed 4 January 2020].

[121] "Kratos XQ-58 Valkyrie (XQ-222)," MilitaryFactory.com, 8 March 2019. [Online]. Available: https://www.militaryfactory.com/aircraft/detail.asp?aircraft_id=1755. [Accessed 4 January 2020].

[122] "X-47B Unmanned Combat Air System (UCAS)," Verdict Media Limited, [Online]. Available: https://www.naval-technology.com/projects/x-47b-unmanned-combat-air-system-carrier-ucas/. [Accessed 4 January 2020].

[123] B. Wang, "Russia's Unmanned Next Generation Fighter is being based off the MIG Skat Stealth UCAV Prototype," Next Big Future Inc., 16 January 2015. [Online]. Available: https://www.nextbigfuture.com/2015/01/russias-unmanned-next-generation.html. [Accessed 4 January 2020].

[124] "X-47B UCAS," Northrop Grumman Corporation, 2015. [Online]. Available: https://www.northropgrumman.com/Capabilities/X47BUCAS/Documents/UCAS-D_Data_Sheet.pdf. [Accessed 4 January 2020].

[125] B. McKinney, "Unmanned Combat Air System Carrier Demonstration," Northrop Grumman Corporation, 19 December 2012. [Online]. Available: https://www.northropgrumman.com/Capabilities/X47BUCAS/Documents/X-47B_Navy_UCAS_FactSheet.pdf. [Accessed 4 January 2020].

[126] G. Robbins, "More Drones, Smaller Navy," The San Diego Union-Tribune, 4 May 2010. [Online]. Available: https://www.sandiegouniontribune.com/sdut-more-drones-smaller-navy-2010may04-htmlstory.html. [Accessed 4 January 2020].

[127] "Lockheed Martin RQ-170 Sentinel UCAV," SimplePlanes.com, 25 April 2018. [Online]. Available: https://www.simpleplanes.com/a/41Dajr/Lockheed-Martin-RQ-170-Sentinel-UCAV. [Accessed 4 January 2020].

Appendix

PNN Classification

Deep Learning Toolbox of Matlab R2018b is used for Probabilistic Neural Networks. The function is called as *newpnn* in Matlab and used as following:

```
%input patterns
P = [successful_patterns,unsuccessful_patterns];
%classes of input patterns
Tc = [successful_set,unsuccessful_set];
%Target class indices are converted to vectors
T = ind2vec(Tc);
%apply PNN
net = newpnn(P,T);
%apply trained network to guess the classes of the untried_patterns
Y = sim(net,untried_patterns);
%classes of untried_patterns are found as:
Yc = vec2ind(Y);
```